Kingdom of Saudi Arabia
King Abdulaziz City For
Science and Technology
General Directorate of
Research Grants Programs

AT – 20 - 80

FINAL REPORT

DESIGN OF AN INTELLIGENT TELEROBOTIC SYSTEM

Dr. Mayez Al-Mouhamed
Dr. Onur Toker
Dr. Nesar Merah

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

2005G.

Contents

List of Figures

List of Tables

Acknowledgement

The project team thanks His Excellency Professor Khaled S. Al-Sultan, Rector of King Fahd University of Petroleum and Minerals, for his support of KACST project AT-20-80. We also thank Professor Mohammed O. Budair, Vice Rector for Graduate Studies and Scientific Research, King Fahd University of Petroleum and Minerals, for his guidance and support of the above project.

The project team thankfully acknowledges the General Directorate of Research Grants Programs at King Abdulaziz City for Science and Technology (KACST) for funding and supporting the research project AT-20-80 entitled "Designing of an Intelligent Telerobotic System".

The project team deeply thanks

1. Dr. Jaralla Al-Ghamdi, Dean of the College of Computer Science and Engineering, KFUPM, for his support of the project,

2. Dr. Sadiq Sait Mohammad, chairman of the Computer Engineering Department, College of Computer Science and Engineering, KFUPM, for his support of the project,

3. Dr. Omar Al-Turki, chairman of the Systems Engineering Department, College of Computer Science and Engineering, KFUPM, for his support of the project,

4. Dr. Faleh A. Al-Sulaiman, Chairman of the Mechanical Engineering Department, College of Engineering, KFUPM, for allowing the project team to use the Mechanical Engineering Workshop for the manufacturing tasks,

5. Mr. Abdul-Khalik Al-Harthy and Mr. Asif Iqbal, Systems Engineering Department, College of Computer Science and Engineering, KFUPM, for their contribution to the above project,

6. Mr. Kamal Abdul-Ghani and Mr. Romeo Agua, Mechanical Engineering Workshop, KFUPM, for their work in achieving the manufacturing tasks,

7. Mr. Ahmad Abdul-Mutaleb, KACST, for his administrative support and his effort with KACST in all related aspects of the project.

8. Mr. Ibrahim Al-Bathi, Mr. Sami Al-Subhi, Mr. Rakan Al-Otaishan, and Mr. Hisham Al-Garni, Computer Engineering department, College of Computer Science and Engineering, KFUPM, for their participation and contribution to this project as part of their Computer Engineering Senior Design Projects.

9. Mr. Mohammad Hafeez and Mr. Mohammad Khourshid, Computer Engineering department, KFUPM, for their continuous support.

1 Project Summary (Arabic)

تصميم نظام ذكي للتواجد عن بعد بواسطة الروبوت

ملخص :

يتناول المشروع تصميم وتقييم نظام الأذرعة الآلية التي تتألف من ذراع آمر (زبون) وذراع منفذ (خادم) متصلين بواسطة شبكة محلية للاتصالات. جرى تصميم نظام زبون – خادم متعدد البرمجيات باستعمال المعالجة الموزعة مع لغة سي المنظورة وتكنولوجيا (.NET)، ولتوفير الرؤية ثلاثية الأبعاد جرى تصميم نظام موزع لنقل الصور بواسطة الشبكة كما تم بنجاح توصيل هذا النظام إلى نظارة ثلاثية الأبعاد في محطة الزبون. وجرى تصميم وتصنيع ذراع آمر ذو ست وحدات حرية بجامعة الملك فهد للبترول والمعادن، كما جرى تصميم نظام استشعار للقوى يركب على رسغ ذراع منفذ من نوع البوما (PUMA 560). صمم الذراع الآمر بطريقة تعزز استقلالية الحركة الخطية والدورانية، بحيث أن العامل الإنساني الذي يمسك بالذراع الآمر يشعر بتساوي المقاومة الميكانيكية نظراً لالتقاء محاور الدوران الثلاثة النهائية بالذراع. وتم بنجاح تطوير مجموعة من المساعدات البرمجية لعملية العمل عن بعد بواسطة الأذرعة الآلية كوظيفة التبديل، وتوسعة المجال، والتماثل المنطقي للحركة بين يد الإنسان والغرض المعالج، والتحكم بالذراع المنفذ بتخفيف المقاومة الميكانيكية، وأتمتة المهام، والتحكم بتركيز الكاميرات. جرى استعمال النظام المصمم بنجاح في ظروف الزمن الحقيقي أثناء النقل المتواصل للمعلومات مع صور الرؤية ثلاثية الأبعاد وأوامر الحركة والقوى المنعكسة. وتبين أن كفاءة الرؤية ثلاثية الأبعاد ممتازة من حيث إمكانية تقدير العمق من خلال الصور المنقولة. وجرى تقييم التأخير والتغيير في التأخير الزمني للصور، وأوامر الحركة، والقوى المنعكسة، وللتقليل من تأثيرات التأخير استعملت هندسة الأنظمة البرمجية بهيكلة النظام على أساس البرمجيات المتوازية، وبالنتيجة يمكن نقل (١) أوامر الحركة بتراتب ٥٠هرتز، (٢) القوى المنعكسة بتراتب ٧٦ هرتز، (٣) صور الرؤية ثلاثية الأبعاد بتراتب ١٧ صورة في الثانية باستعمال شبكة اتصالات محلية (100 Mbps)، وقمنا بنجاح بمهام عن بعد كصب السوائل، إدخال عامود في فراغ ضيق، وتجميع مضخة مياه صغيرة، وعمليات توصيل الأسلاك بالدوائر الإلكترونية الصغيرة، مما يشير إلى أن النظام المقترح يوسع من التنسيق الحركي بين العين واليد لدى ا لإنسان بواسطه شبكا ت الاتصالات وهو أداة نافعة للعمل عن بعد في المجالات المهددة التي لا تسمح للإنسان بالتواجد فيها.

ملخص موسع :

يتناول هذا المشروع تصميم وتقييم نظام الأذرعة الآلية التي تتألف من ذراع آمر وذراع منفذ متصلين بواسطة شبكة اتصالات محلية (LAN)، والذراع الآمر يستعمل كأداة كي يمسك به الإنسان لإملاء الحركة بحيث يتمكن الحاسب الآلي المتصل بالذراع الآمر من تحديد الحركة ونقلها عبر شبكة الاتصالات إلى الذراع المنفذ الذي يقوم بإعادة تمثيلها، وهذه الفكرة تسمح بنقل الحركة من مكان إلى آخر وذلك بتخطي محدودية المكان الجغرافي، ونظراً لأهمية الرؤية والإحساس بالقوى الخارجية فإن النظام الذي جرى تطويره يقوم أيضاً بنقل الصورة ثلاثية الأبعاد والقوى المنعكسة من مكان العمل (الخادم)، حيث يتواجد الذراع المنفذ، إلى مكان التحكم (الزبون) حيث يتواجد الذراع الآمر. والنظام المذكور يسمح للإنسان بنقل حركة ذراعه بكفاءة بالإضافة إلى توفير الرؤية ثلاثية الأبعاد والقوى الخارجية المنبعثة من محيط العمل إلى الإنسان بحيث يرى ويشعر بنتائج عمله ويمسك الإنسان بالذراع الآمر ويحركه من خلال قيامه بعمل ما ونظراً لوجود ست درجات حرية بالذراع الآمر لذلك فيمكن لنهايته أن تأخذ أي موضع وأي اتجاه حسب توجيهات الإنسان، ويقوم الحاسب المتصل بالذراع الآمر بقراءة إحداثياته أثناء الحركة مما يسمح بقراءة حركة ذراع الإنسان بصورة متواصلة، كما يمكن نقل هذه الإحداثيات إلى ذراع منفذ يقوم بنفس الحركات ولكن في مكان آخر، ولكي يتمكن الإنسان من القيام بعمل عن بعد بكفاءة عالية فمن الضروري أن يتمكن من رؤية نتائج عمله باستعمال وحدتي كاميرا موجهة إلى محيط العمل حول الذراع المنفذ بحيث تكون المسافة بين الكاميرات حوالي ٦ سنتيمترات ، وهي تقريباً المسافة بين عيني الإنسان، وتنقل الصورة من كل كاميرا إلى الإنسان عبر الشبكة وتعرض على شاشتي نظارة خاصة يستعملها الإنسان المتواجد في محيط الذراع الآمر. وهذا يمكن للإنسان من الرؤية الطبيعية (ثلاثية الأبعاد) لأن الصورتين اللتين يراهما منبعثتان من وحدتي كاميرات كعيني الإنسان وفي هذه الحالة يقوم الدماغ بتداخل الصورتين مما يؤدي إلى رؤية العمق في الصورة لذلك تدعى هذه الخاصية بالرؤية ثلاثية الأبعاد. وقد أثبتت التجارب أن الرؤية ثلاثية الأبعاد نافعة جداً في العمل عن بعد إلا أنها غير كافية لمماثلة العمل التعاوني بين اليد والعين والإحساس بالقوى أو

اللمس. لذلك فمن الضروري نقل القوى التي تنجم عن تلامس الغرض الذي يجري التعامل به مع محيطه ونقله إلى الذراع الآمر بحيث ينتج نفس القوى المتجهة كي يشعر بها الإنسان، ويهدف هذا المشروع إلى تصميم نظام للتواصل عبر شبكات الاتصالات المحلية بحيث يتمكن الإنسان الممسك بالذراع الآمر من القيام ببعض الأعمال عن بعد.

تم تصميم نظام زبون – خادم متعدد البرمجيات الموزعة باستعمال المعالجة المتوازية التوجهية الفرضية ولغة (C) المنظورة وتكنولوجيا (.NET) للاتصال عبر الشبكات التي طورتها شركة ما يكروسوفت لتعمل بكفاءة على الحاسب الذي يستعمل ويندوز كنظام للتشغيل، وتسمح البرمجيات الموزعة ببناء الأنظمة البرمجية بكفاءة نظراً إلى المشاركة في استعمال البرمجيات في نظام الزبون والخادم على السواء، كما تسمح البرمجيات المتوازية بهندسة تزامن العمليات لتحسين الكفاءة وبصورة خاصة تقليل زمن المعالجة، ونقل الصورة من الكاميرا إلى الحاسب الشخصي، وأخيراً إلى الشبكة بتفادي التسلسل الزمني المعيق. وتسمح تكنولوجيا (.NET) بالتركيز على التطبيقات دون الاهتمام بكيفية نقل المعلومات بواسطة الشبكة. ذلك أن كل البرمجيات الأساسية المتوفرة في الطرف الآمر (الخادم أو الزبون) يمكن استدعائها وتنفيذها دون الحاجة إلى الاهتمام بكيفية نقلها عبر الشبكة بالزمن الحقيقي وتضمن الشركات المنتجة للبرمجيات المذكورة أعلاه أن تتم عملية الاتصال المتواصل بين الزبون والخادم بأقل تأخير وبأقل جهد ممكن. وقد جرت عملية تصميم واختبار هذا النظام بنجاح بنقل أوامر الحركة والصورة ثلاثية الأبعاد والقوى المنعكسة بكفاءة عالية، كما جرى تصميم لوحة استعمال في كلٍ من محطتي الزبون والخادم بهدف صيانة النظام واختباره وإعادة تشغيله بسرعة في حالة الخلل. وبصورة مبسطة ليعمل النظام بصورة دورية فبعد قراءة إحداثيات الذراع الآمر يقوم الحاسب بحساب إحداثيات يد الإنسان متمثلة بست عناصر مستقلة، ثلاثة منها تعين إحداثيات اليد والثلاثة الأخرى تعين الاتجاه العام وتنقل هذه المعلومات إلى محطة الخادم حيث يتم إضافة متغيرات الحركة إلى إحداثيات الجزء النهائي للذراع المنفذ بناءً على الناتج، ثم ترسل إلى الذراع المنفذ ليقوم بالحركة المناسبة لتنفيذها.

تم تصميم عدد من البرمجيات التي تدعم العمل عن بعد بواسطة الأذرعة الآلية، كوظيفة التبديل التي تسمح للإنسان بتوقيف الاتصال بين الذراعين مؤقتاً ريثما يعيد الإنسان ذراعه إلى مجال عمله الحيوي حيث يتمكن من العمل بكفاءة أكبر ومتى تم ذلك تتم إعادة التوصيل بين الآمر والمنفذ من حيث توقف المنفذ، وهذه الوظيفة نافعة جداً في العمل عن بعد ولا يمكن الاستغناء عنها، ومن الوظائف النافعة جداً وظيفة توسعة المجال بين الذراعين ذلك أنه تلزم دقة عالية في بعض مراحل العمل وعندها يمكن للإنسان زيادة الدقة بالحركة باستعمال مفتاح خطي متوفر عند قبضة الذراع الآمر، ويمكن أيضاً تشغيل مهام خاصة ضمن النظام الخادم كالتحكم بالقوى المحلية لتقليل القوى المنبعثة خلال الاحتكاك أو الاصطدام وهذه المهمة نافعة جداً أثناء عملية إدخال جسم في جسم آخر بحيث نتفادى الانسداد، وتساند هذه المهمة أيضاً في تفادي التأخير الذي تعود أسبابه إلى بطئ برمجيات الاتصالات وانتقال الإشارات على الشبكة، وذلك أن التحكم المحلي هو نوع من التحكم المستقل المساند للإنسان البعيد، ومن المهام الأساسية هي تماثل الحركة بين تلك التي تقوم بها يد الإنسان وتلك التي تنفذ على الغرض المحمول من قبل الذراع المنفذ، وهذا التماثل يسمح بتكافئ الحركة وتناسبها على أسس رياضية بنيت على التماثل الهندسي للذراعين المذكورين أعلاه. وهذا التماثل يسمح للإنسان بتركيز نظره أثناء العمل على الغرض المعالج بغض النظر عما تفعله يد الإنسان أو ما ينتج عن ذلك في حركة الذراع الآمر، وتثبت التجارب التي قمنا بها أن التماثل المذكور أعلاه ذو كفاءة عالية ويظهر هذا بشكل واضح من خلال المهام المصورة كمهمة صب الماء من وعاء إلى آخر كما يظهر خلال عملية تركيب مضخة الماء.

تم تصميم نظام زبون – خادم لنقل الصورة ثلاثية الأبعاد بناءً على استعمال البرمجيات المتوازية بحيث يمكن أحد البرامج بنقل الصورة من إحدى الكاميرات إلى ذاكرة الحاسب الشخصي بينما ينقل برنامج آخر صورة سابقة عبر الشبكة إلى نظام الزبون حيث تعرض على الإنسان، وتسمح التقنية المذكورة أعلاه بتفادي التأخير والتوازي في عمليات الحاسب الشخصي. وجرى تصميم وتصنيع ذراع آلي آمر ذو ست درجات حرية تسلسلية، وهو ذراع خفيف نظراً لاستعمال الأسلاك الفولاذية المرنة في نقل الجهد من محركاته إلى أضلاعه وأخيراً إلى قبضته حيث يمسك بها الإنسان عندما يعمل عن بعد. وهندسة الذراع الآمر مبينة

على فكره وضع يد الإنسان في نقطة تقاطع محاور الدوران الثلاث الأخيرة في الذراع المذكور أعلاه، وهذا يسمح بفصل الحركات الانتقالية عن الحركات الدورانية كما يسمح للإنسان بالشعور بوجود مقاومة ميكانيكية ثابتة في مختلف اتجاهات دوران يد الإنسان، إن هندسة هذا الذراع الآمر أثبتت جدواها نظراً للكفاءة العالية التي اكتشفناها أثناء القيام بالأعمال عن بعد، وأهم هذه الخصائص التوصيل المنطقي للحركة من يد الإنسان، حيث يمسك بالذراع الآمر، إلى الغرض الذي يجري التعامل به في قبضة الذراع المنفذ. وجرى تصميم البرمجيات الرياضية التي تنقل التغيير الحركي من الذراع الآمر إلى الذراع المنفذ، وبشكل دقيق من قبضة الذراع الآمر إلى الغرض المعالج في قبضة الذراع المنفذ. وبالإضافة إلى متابعة حركة الذراع الآمر بواسطة الحاسب الشخصي يسمح الذراع الآمر نقل الشعور بالقوى التي تنبعث من محيط العمل إلى الإنسان وذلك بتوصيل تلك القوى إلى المحركات الكهربائية داخل الذراع الآمر ونظراً لتوصيل تلك المحركات بيد الإنسان القابض على الذراع الآمر فإن هذه القوى يشعر بها الإنسان كما لو كانت منبعثة من محيطه المباشر، والمماثلة الرياضية تسمح بتحديد كمية واتجاه القوى التي تنبعث على يد الإنسان، والجدير بالذكر أن تلك القوى يمكن أن تكون غير حقيقية ومنبعثة عن برمجيات افتراضية تحاكي عمليات محددة كميكانيكية النابض على سبيل المثال، ومن السلبيات التي جرى ملاحظتها على الذراع المذكور هي خاصية المطاطية في توصيلات نقل الجهد حيث توجد فروق بين زوايا المحركات الكهربائية وزوايا قبضة الإنسان نظراً لمطاطية التوصيلات مما يؤدي إلى الإخلال بعملية متابعة حركة يد الإنسان ونقل الجهد من المحرك إلى يد الإنسان. وينصح باستعمال أذرعة أصغر طولاً أو أسلاك فولاذية أقل مطاطية لتحديد المعوقات المذكورة أعلاه، ويعتبر هذا الذراع أحد أهم عناصر المشروع وذلك لأن أذرعة مشابهة جرى استعمالها في جامعات عالمية في عمليات الجراحة الروبوتية بنجاح مما يفسح للجامعة فرصة طيبة لتطوير أنظمة ميكاترونيه للمساندة في عمليات الجراحة على الإنسان.

وجرى اختبار الذراع المذكور بنجاح في عمليات ميكانيكية تتم عن بعد وتبين بشكل واضح نجاح فكرة التوصيل المنطقي للحركة كما سمح ذلك بتقييم الوظائف المساندة التي قمنا ببرمجتها كوسائط مساندة العمل عن بعد بواسطة الأذرعة الآلية. وقد أثبتت التجارب أن

الرؤية ثلاثية الأبعاد تعمل بكفاءة عالية عند استعمال النظارات الخاصة وتسمح هذه التكنولوجيا المتقدمة رؤية العمق أو البعد الثالث في الصورة ثنائية الأبعاد، وهذا يسمح للإنسان برؤية العمق بكفاءة تمكنه من تقدير الأوضاع الهندسية للأغراض التي يجري التعامل معها عن بعد، كما يمكن تكبير أحجام هذه الأغراض ورؤيتها بوضوح تام أفضل من كفاءة العين المجردة، أن التنسيق بين العين يأخذ بعداً جديداً باستعمال نظام الأذرعة الآلية، ومن المعروف أن الإنسان طور كفاءة عمله وبصورة خاصة كفاءة التنسيق بين العين واليد من خلال وجوده على الأرض لملايين السنين والنتيجة هي نظام عصبي عالي الكفاءة للتنسيق بين الفعل اليدوي ورد الفعل بقياسه بالنظر أو بالشعور بقوى تفاعله، ويمكن لنظام الأذرعة الآلية أن يسمح بامتداد التنسيق بين العين واليد لدى الإنسان من مكان تواجد الإنسان إلى أبعاد أخرى قد تكون متناهية في الكبر أو متناهية في الصغر، مما يسمح للإنسان بتخطي حدوده الطبيعية كالحجم والطول بحيث يمكنه العمل في حيز متناهي الصغر يتفاعل معه بتحويل قوانينه الدقيقة إلى قوانين حيز الإنسان مما يسمح له بالعمل في أوساط تختلف ميكانيكيتها عن تلك المتوفرة في محيطها الطبيعي، ولتقييم كفاءة نظام الأذرعة الآلية قمنا بتقييم اعتما دية الاتصال بين الخادم والزبون وخصوصاً التأخير والتغير في التأخير وأعدنا صياغة العديد من البرمجيات لجعلها أكثر توازياً متى أمكن ذلك بحيث تنفذ بصورة تسابقية، وقمنا بتقييم الكفاءة النوعية أثناء القيام ببعض الأعمال الميكانيكية مما أدى إلى استخلاص الكثير من العبر على كل المستويات، كالنواحي الميكانيكية التي تتعلق بتصميم الأذرعة الآلية ونقل حركتها وأبعادها وطريقة تصميمها بحيث تكون خفيفة الوزن لتنقل بكفاءة القوى الخارجية إلى يد الإنسان، وجرى تقييم التأخير في نقل الصورة، وأوامر الحركة، والقوى المنعكسة، وبالنتيجة يمكن نقل أوامر الحركة بتراتب ٥٠ هرتز، والقوى المنعكسة بتراتب ٧٦ هرتز، والصور ثلاثية الأبعاد بتراتب ١٧ صورة في الثانية باستعمال شبكة اتصالات محلية ١٠٠ ميغابت في الثانية. ولتطوير الأذرعة الآلية في المستقبل، نوصي بتصميم أذرعة آلية آمرة عامة التوصيل والتلائم بحيث تجهز بوسائط متطورة لدعم العمل عن بعد بواسطة الحاسب الآلي، كما نوصي بتصميم أذرعة آلية منفذه خفيفة ومتحركة تتناسب مع محيط العمل.

وبصورة عامة يشكل نظام الأذرعة الآلية أداة نافعة للعمل عن بعد في المجالات التي لا يمكن للإنسان التواجد فيها لمختلف الأسباب كالأوساط المهددة. ويشكل هذا النظام أداة استراتيجية للبحث والتطوير لمختلف احتياجات المملكة وخصوصاً في تطوير أنظمة عالية التقنية لدعم العمليات كظروف السلامة في الأوساط المهددة وأعمال الصيانة في الأعماق البحرية والجراحة باستعمال الأذرعة الآلية.

2 Project Summary (English)

The project is on the design and evaluation of a telerobotic system that consists of a master arm (client) interconnected to a slave arm (server) by means of a local area network. A Multithreaded Distributed Components Client-Server interface was designed and implemented using object-oriented programming, Visual C#, and .NET technology. For the stereo visualization a distributed framework was designed for relaying stereo vision over the network and successfully interfaced it to a head-mounted display (HMD) at the client site. A 6 DOF light telerobotic master arm and a wrist force sensor were designed and manufactured at KFUPM. The master arm structure uncouple motion translation from change in orientation. The iso-impedance feature of master arm and its concurrent rotation axes of last three dof makes it transparent to the operator. A library of computer aided telerobotic functions was successfully implemented. It includes functions like the master shift, space scalability, mapping of operator hand to a floating tool, compliance loop at slave arm, automatic tasks such as the camera zooming control, etc. The proposed telerobotic system system was successfully operated under real-time multi-streaming of stereo information, motion commands, and force feedback. The real-time delays and jitter were evaluated in the case of end-to-end multistreaming of video data, force information, and motion control over a LAN. To minimize delays the telerobotic client-server was engineered with respect to its motion coordination system and stereo vision server using concurrent programming. The optimized telerobotic system has refreshing rates of 50 Hz in real-time transfer of commands, 76 Hz for reflected force feedback, and 17 fps for stereo vision over a 100 Mbps LAN. The stereo visualization provides excellent depth perception. Using the master arm, HMD, and reflected force feedback we successfully carried out a number of remote tasks like pouring of water, peg-in-hole insertion, assembly of a small water pump, and wire-wrapping. Video clips on the above experiments are available at [1]. The proposed telerobotic system allows extending natural eye-hand motion coordination and human's arm manipulative capabilities and dexterity through standard computer network even in the case of scaled-down operations performed in small spaces. The proposed telerobotic system is a useful tool to perform remote work in hazardous, hostile, and small environments.

3 Introduction

The aim of this project is to design and evaluate a telerobotic system to allow human operators to perform working tasks in hazardous and hostile environments of interest to Saudi Arabia. The targeted telerobotic system must have the following functions:

1. Perform remote manipulative tasks by using a master workstation that is interconnected to a slave robotics workstation through a local area computer network.

2. Interface master and slave workstations to allow the operator to "feel" the reflected force feedback and to see the slave scene (stereo-vision) that are continuously sent from the slave workstation through a computer network. Using the above retroceptive information, the operator can remotely manipulate the slave robot arm to achieve some task whose timely accomplishment requires high quality visual information and force feeling.

3. A library of intelligent telerobotic services is used to support the operator. Some of the functionalities of the library are:

   (a) ability of the operator to manually control the trajectory (using a master arm) of the slave arm through a computer network in real-time.

   (b) ability to shift the master arm position and orientation to more dexterous configurations without causing change to the positioning of slave arm,

   (c) ability to activate slave automatic tasks,

   (d) ability to remotely activate local compliance loop for the slave arm by using force sensing to ensure that no excessive force is exerted, and

   (e) ability to remote control the orientation and zooming of the cameras in the slave arm environment,

4. Investigate the problems of time delay and delay jitter caused by network latency and their effects on telerobotics. It also includes the evaluation of the performance of networked telerobotics and its operability in the presence of network delays.

This final report is organized as a set of seven sections. The first and second sections are dedicated to Arabic summary, English summary, and an introduction, respectively. Section 4 presents the literature surveys for the various aspects of the project. Section 5 describes the research methodology which includes the description of the original tasks and states the way they were addressed during the project. Section 6 describes the results obtained from the present work. A discussion of these results is also included in this section. Finally Section 7 presents some conclusions and recommendations concerning the discussed tasks and the overall research project. Sections 8 and 9 are dedicated to Appendix A *Assembly drawing of the master arm* and Appendix B *Detailed drawing of the master arm*, respectively.

4 Literature surveys

The literature surveys is divided into four parts which are (1) dexterous master arms, (2) the master arms and haptic devices, (3) the compliant end effectors and force sensors, (4) the stereoscopic visualization, (5) networked Internet telerobotics and delays, and (6) network protocols for telerobotics.

In sub-section 1, the "dexterous master arms" presents a taxonomy of some representative commercial and research master arms including serial and parallel structures. In sub-section 2, the "compliant end effectors and force sensors" presents representative approaches for sensing and measuring external forces in a robot arm. In sub-section 3, the "stereoscopic visualization" presents the technology, tools, and bandwidth needed to implement networked stereoscopic visualization. In sub-section 4, the "networked Internet telerobotics and delays" reports on methods that address the effects of delays in telerobotics. In sub-section 5, the "dexterous master arms" presents the need for advanced robotic structure to support telesurgery. In sub-section 5, the "network protocols for telerobotics" presents end-to-end real-time client-server software and quality of service requirements for telerobotics.

4.1 Dexterous master arms in telerobotics

Robotic technology [2, 3, 4, 5] is enhancing surgery through improved precision, stability, and dexterity. In image-guided procedures, robots use magnetic resonance and computed tomography image data to guide instruments to the treatment site. This requires new algorithms and user interfaces for planning procedures; it also requires sensors for registering the patient's anatomy with the preoperative image data. Minimally invasive (MI) procedures use remotely-controlled robots that allow the surgeon to work inside the patient's body without making large incisions. Specialized mechanical designs and sensing technologies are needed to maximize dexterity [6] under these access constraints. Robots have applications in many surgical specialties. In neurosurgery, image-guided robots can biopsy brain lesions with minimal damage to adjacent tissue. In orthopedic surgery [2] , robots are routinely used to shape the femur to precisely fit prosthetic hip joint replacements. Robotic systems are also under development for closed-chest heart bypass, for microsurgical procedures in ophthalmology, and for surgical training and simulation. Although results from initial clinical experience is positive, issues of clinician acceptance, high capital costs, performance validation, and safety remain to be addressed.

Robotic manipulators promise to solve many of these problems. The challenge is to design devices with good dexterity [6] and intuitive control that can be inserted through small incisions. One focus is the development of general purpose systems that can execute a range of procedures in general, thoracic, and gynecological surgery. These systems are often configured so that the surgeon sits at a console in the operating room and uses a *master control manipulator* that sends commands to the robots performing the surgical procedure. *Video images*, and sometimes *force sensations* [7, 8, 9], are reproduced at the surgeon's console. Other systems under development are aimed at specific access modalities such as percutaneous needle puncture and transurethral prostate resection. There are also systems that take advantage of robotic ability to perform stable and untiring holding tasks, such as

endoscope pointing and organ retraction, and to work at microscopic scales.

Other procedures are performed interactively or assistively, meaning the surgeon and robot share control. One example is a robotic system for bone cutting in knee joint-replacement procedures. The surgeon grasps the cutting tool at the end of a *low-impedance robot* [10, 11] manipulator and moves the tool to reshape the bone to fit the prosthetic joint. The robot monitors the surgeon's actions and permits free motion in the appropriate cutting region but applies *forces* [8] to prevent motion into regions where bone should not be removed. This allows the surgeon to supervise and control the robot, using innate human sensing and judgment, while it also provides active constraints that increase safety and accuracy of the cutting process. This approach may also improve acceptance of robotic systems by surgeons and patients, as the surgeon remains in control of the procedure.

Robots for assistive control applications may require new *manipulator designs*; most robots are designed for *high stiffness* to ensure geometric accuracy at the tip in the presence of variable task loads. This makes it difficult to design a sensing and control scheme that allows the robot to follow the surgeon's hand without the application of *large forces* or significant *time delays*. At the other extreme of the autonomy scale, the minimally invasive surgical robot systems are often controlled explicitly by the surgeon. Each motion the surgeon makes with the master manipulator at the control console is transmitted to the robot working inside the patient's body. The surgeon formulates all motion commands on the basis of sensory information returned from the surgical site, which usually consists of video images. Because the master manipulator is physically separate from the surgical robot, this control mode falls under the category of *teleoperation*, even though the surgeon is usually located in the operating room with the surgical robot. Researchers have proposed that this technology will allow surgeons to treat patients from a considerable distance. This could reduce the need to transport patients to highly specialized surgeons and avoid exposing surgical personnel to hazardous conditions in wartime or following natural disasters. A central problem is communication delays: Satellite links, for example, often have round-trip delays that last from a fraction of a second to several seconds. This can greatly slow task execution, as the surgeon must pace the procedure to wait to see the effects of commanded motions. In the case of force feedback [12, 13], it has been known for decades that delays of this magnitude can cause instability of the robot control system, although various techniques can help to minimize this problem. A less ambitious application is telementoring, where an experienced surgical specialist can observe and advise surgeons performing a procedure in a distant location. Robotics permits new forms of interaction in telementoring, such as giving the mentor control of the endoscopic camera. It remains be seen whether the benefits of long-distance telerobotic surgical applications will outweigh technical hurdles, acceptance barriers, and attendant costs.

Currently, many image-guided surgical [2] applications use off-the-shelf industrial robot manipulators. This speeds development and reduces costs, but these devices have not been optimized for the characteristics of specific surgical tasks. For example, most industrial robots are designed for good repeatability but may lack sufficient positional accuracy. Similarly, assistive systems that share control between the robot and human surgeon would benefit from the development of low impedance manipulators [12, 13] in place of highly geared, stiff industrial arms.

Other advantages that can accrue to specialized designs include improved sterility and compatibility with imaging systems (e.g. transparency to Xrays). In teleoperated systems, access constraints have always necessitated the development of new manipulator configurations, but kinematic structures and actuator technologies are far from perfected. These technologies also limit the development of microrobots for medical applications.

In teleoperated systems for MI or microsurgical procedures, there is substantial room for improvement of control and sensory feedback interfaces. In general, the human factors aspects of these systems have been little studied. Research questions [2, 3, 4, 5] include master manipulator configuration, mapping between master and remote robot coordinate systems, scaling laws for micromanipulation systems, and video, force, and tactile feedback fidelity and bandwidth requirements.

Telerobotic MI surgery reduces patient pain and trauma, leads to a fewer complications, and shorten convalescent periods. However, the shift from open surgery to telrobotic MI surgery was accompanied with motion constraints and pivot geometry. An effective cognitive mapping [14] is needed between visual and motor frames. Two methods used: (1) mapping from surgeon motion at master arm and motion of instrument tip within the body of the patient, and (2) screen-mapping in which the body-frame at master is mapped to image based frame at slave. It reveals that mapping of the instrument tip frame to the surgeon handle provides the best results with the lowest error rates.

The development of a master arm with minimal impedance for gentle manipulation [11] of known or unknown objects in unstructured environments. Similar to a human, The master arm should minimize the contact forces by a combined approach based on minimal mechanical friction and active impedance control. To achieve low contact forces in telemanipulation tasks a model of the master arm is used to control the impedance which enables the use of low gain controllers so that to significantly reduce frictions sensed by the operator.

Minimally invasive surgery (MIS) is based on a inserting a surgical tool through a fixed pivot point. In telerobotic MIS six independent forces and moments are measured at the tool tip. Force is sampled at about 1 KHz. The main issue is to provide a dexterous master arm that allows the surgeon to safely tele-operate soft tissue. The difficulties are the lack of depth perception and the single incision point that constrain the surgeon. The instrument shaft acts as a perfect lever and force is a scaled and reproduced by the surgeon master arm which provides the feeling of forces that would experience if the surgeon was directly handling the instrument. In robotic MIS force feedback (FF) enhances performance [15] of robotic surgery. In a blunt dissection where artery is stiffer that surrounding tissue task performance with laparoscaopic video produced a gain of 75% and 150%. The transmitted forces is between 0.1-1 N. Here it is critical to preserve surrounding structures and avoid tissue trauma. The absence of FF increased the average force applied on tissue by at least 50% and peak force was doubled. The number of errors that cause damage to tissue increased by a factor of three. However, FF does not significantly improve the rate and task precision. FF helped minimizing contact forces between the tool and the working environment. High fidelity FF appeared not to be an essential requirement in all operations.

The use of FF in telerobotic MI surgery enhances operation dexterity:

1. reduces the number of accidental incursions into sensitive structures because it reduces exerted forces.

2. increases operation safety by setting of force safety barrier to avoid damaging tissue.

3. used to drag and guide the instrument using the concept of force avoidance and force display

4. introduced in a natural way without planned training.

5. lead to designing of instruments with force sensing capability.

Modelling the master arm with feed-forward impedance control can result in low interaction forces. This gives the ability to vary the impedance of the robotic arm from being very gentle to very stiff, thereby accomplishing a variable dynamic range of impedances similar to a human arm. There is need to perfect the performance of a robotic arm for a wide variety of tasks commonly carried out by humans from simple gentle probing to complex manipulation and reaching tasks. The objective [2, 3] is to create robotic systems that can perform simple grasping and manipulation operations on a range of objects in unstructured environments. The main component to achieve this objective are the variable manipulator impedance and simple visual and contact sensing.

There is pressing need to explore the control of a manipulator to minimize interaction forces in the initial groping phase of acquiring unknown objects. This requires the ability to move the manipulator through space with low impedance but with reasonable positional accuracy. This precludes conventional position control methods where high feedback gains are used to minimize position errors, because these high gains also produce high impedance. One approach is to use a model-based approach, where the model predicts the manipulator torques required to follow the desired trajectory, so that low feedback gains produce low impedance in the event of unanticipated contact.

In addition to a low impedance position controller, the master arm mechanism must have intrinsically *low impedance*. Highly geared manipulators cannot be back-driven by contact forces, so they must rely on feedback from sensor signals to drive the motors to emulate low interaction impedance. The use of a intrinsically *compliant manipulator* eliminates the need for contact sensing and guarantees good force display. While dynamic models are aimed at improving precision and speed the dynamic model used for impedance control allows controlling the apparent impedance of the manipulator in the contact phase. Thus the dynamic model [10, 11] is used here to:

1. hide the imperfection of the mechanical system from being sensed by the operator,

2. improve the quality of interaction between operator, master arm, and slave arm, and

3. minimizing overall contact impedance and limiting the need for sensing.

This approach is a straightforward extension of model-based impedance control with, however, a different focus. The role of dynamic models in controlling position in free space has largely been aimed at improving precision and speed. In impedance control, dynamic models have often been used to modify the apparent impedance of the manipulator in the contact phase. The aim is to use the dynamic model to achieve good position control in free space while at the same time minimizing the contact impedance and limiting the need for sensing. Research on controlling the forces generated upon contact has often been termed impact control. In these studies, the focus is often on switching between different controllers for the contact phase, and the use of proximity and/or force sensors to anticipate and control the impact. While these approaches have achieved good results, the goal here is to create a controller for use in the free motion phase that does not use sensor feedback to minimize forces in the event of unanticipated contact. This approach [11] improves robustness by eliminating sensors from the control loop, but still results in low interaction impedance through the use of a *back-drivable manipulator*. A low-impedance manipulator is just one component of a system that can grasp and manipulate unknown objects in unstructured environments.

Other mechanisms, such as soft contact surface coverings and reactive control methods that alter the controller and commanded trajectory in response to contact are also essential. Minimizing the impedance of the manipulator, however, is a robust strategy for avoiding large forces in the inevitable unexpected contacts that could damage the object or the manipulator.

This work is part of an effort to develop a robot that can gently interact with objects in unstructured environments, where object size and position may be poorly known. This requires the ability to move the robot to the object with moderate precision but minimal impedance, so that unanticipated contacts do not produce large forces that might disturb the object or damage the robot.

The approach implemented here uses a model of the robot's dynamics to predict the joint torques that are needed to follow the trajectory. If the model is accurate, then only small feedback gains are needed to correct residual errors in the model and compensate for disturbances. For example the WAM robot arm [11] is a highly back-drivable, low impedance device, the resulting system has low impedance even without external sensors. In terms of tuning the controller to particular task requirements, the central tradeoff is between position error and contact force magnitude. As stiffness increases, the error drops and the contact impulse increases, so appropriate performance may be selected. In unstructured environments where object position is uncertain anyway, there may be no reason to use high gains that might engender high forces. One interesting question in this context is what limits the minimum practical impedance? There appear to be two factors unrelated to the modelling issues described above. First, external disturbances can apply forces to the arm that by definition cannot be modelled. Feedback is thus required for correction. This is probably not, however, a significant problem in many situations of interest (e.g. indoor settings, passive loads). The second factor that limits impedance is the intrinsic impedance of the manipulator. High impedance devices such as geared robots must use a force sensor to detect contact, and then actively change trajectory to avoid applying large forces. While these systems have met with some success, issues such as servo delays and sensor noise can produce undesirable force transients.

The goal is to develop a system that is intrinsically as low impedance as possible.

External force sensors [2, 3] and active control mechanisms must be added to respond to contact, and even proximity or visual sensing is needed to respond before contact. But because the manipulator mechanism and control algorithm have intrinsically low impedance, performance limitations or even failure of these sensors and reactive controllers will not generate high contact forces. Another passive mechanism that can limit force magnitudes is covering the contact surfaces on the robot arm with soft material. The initial force transients are due to the dissipation of the robot's kinetic energy in the collision. Generally the terminal part of the robot arm and the manipulated object are hard surfaces, so the contact force rose quickly to a high peak. A layer of rubber would greatly lower the peak magnitude. However, the important point is that the impedance of the manipulator (i.e. its inertia) is a key factor in limiting this force, as the kinetic energy is proportional to the inertia.

4.2 Master arms and haptic devices

The increasing need for enhanced man-machine interaction [7, 16, 17, 18] is pushing for new interfaces that allow humans and robots to exchange a wider range of information. As a consequence, applications involving new interaction modalities such as vision display techniques [19] and virtual reality are being developed. Among these new interfaces, haptic devices [20, 21] or (master arms) are promised a place of choice. Not only does haptic make difficult manipulation tasks possible or easier, it also opens the door to a wide range of new applications in the fields of simulation and assistance to human operators.

There are four key requirements for a haptic device [22] (1) allowing the user to move his hand freely on the desktop, (2) providing a sense of force feeling, (3) possessing six active degrees of freedom, and (4) a large user workspace. In addition there is need to minimize backdrive friction, moving inertia, and backlash. An ideal interface should provide an easy way to control the position and orientation of a virtual object in 3D space. Most commercially available devices have small user workspace.

Some definitions are useful. Haptic relates to the sense of touch and force feeling. Proprioceptive Relates to sensory information about the state of the body (including cutaneous, kinesthetic, and vestibular sensations). Vestibular pertains to the perception of head position, acceleration, and deceleration. Kinesthetic means the feeling of motion related to sensations originating in muscles, tendons and joints. Cutaneous pertains to the skin itself or the skin as a sense organ, sensation of pressure, temperature, and pain. Tactile pertains to the cutaneous sense but more specifically the sensation of pressure rather than temperature or pain. Force feedback relates to the mechanical production of information sensed by the human kinesthetic system.

An ideal interface should provide an easy way to control the position and orientation of a virtual object in 3D space. Most commercially available devices are (1) small and the user cannot map hand movements freely to object movement, and (2) do not provide the sense of force feeling. Numerous applications can benefit from haptic technology, ranging from teleoperation [22] to nanomanipulation, to medical simulators and surgical aids [1]. Moreover, force-feedback devices are moving to the consumer market, and are invading the gaming industry as well as unexpected other

areas such as automobile industry. Haptic devices allow the operator to feel surface constraints and contact forces as he rotates and manipulates a part into place. There are many applications in desktop virtual prototyping which refer to the process by which a new design can be evaluated on a computer without the need to create a physical prototype. Maintenance analysis (aircraft engines and automobiles) is concerned with component accessibility for speedy repair, i.e. component can be inserted into and extracted from its target environment.

On the hardware side, illustrative examples of haptic devices include the Rutgers dextrous hand master [6], the Immersion Feelit Mouse [11], and the PHANTOM [18] [24] master arm. Force feedback joysticks and steering wheels abound in the home PC market, such as the SideWinder Force Feedback Pro Joystick [20]. Specialized haptic devices have been built for the medical market, such as the AccuTouch Tactile Feedback Device for catheter insertion simulation [12]. These devices do not provide 6 dof force feedback. The devices that do provide 6 dof force feedback are not optimized in general for desktop use. The Argonne Arm [3], arguably the first haptic display to be developed, is a human scale robot arm. The Salisbury/JPL Arm [4] is another example. The SpacePen [8] has 20 cubic meters of viable workspace. These devices are not appropriate for desktop use. Agronin [1] describes a 6 dof joystick actuated by strings. Lawrence [16] has a device that can provide up to 6 dof force feedback. Berkelman [5] describes a floor mounted magnetic levitation haptic device with a small range of motion. Salcudean [22] describes a similar interface with an even smaller range of motion. The Freedom 6 [21] has low peak force and a small range of motion in two of the three rotational freedoms. While these devices can deliver excellent force feedback, they all suffer from workspace limitations, especially in rotation, which reduce their usability in a maintenance analysis environment.

Representative structures from the major three classes of master arm devices: (1) commercial haptic devices, (2) parallel arm structures, and (3) exotic arm structures are discussed below. Illustrative examples of haptic devices with serial structures are presented in the following subsections.

4.2.1 Commercial haptic devices

In the following illustrative examples of commercial haptic devices (serial structures) are presented.

1. The Phantom Haptic Device The Phantom Haptic Device [22] (Sensable Technologies, Inc., USA) is the standard device used by most. It is a 6 dof input but only a 3 dof output. Various sizes of the device are available. Their small desktop has a 16cm x 13cm x 13cm workspace. A max force of 6.4N and can be driven with nominal position resolution of 0.02mm. The device has a stiffness of 3.16N/mm with very low inertia of under 75g. Their largest device, the Phantom 3.0, has a workspace of 42cm x 59cm x 82cm, maximum force of 22N, nominal resolution of 0.02mm, stiffness of 1N/mm, and an inertia of under 150g. The device produces very smooth and crisp results.

   The PHANTOM Premium 6 dof prototype is a desk mounted force feedback system that provides six degree-of-freedom force and torque feedback. To provide force feedback, the base of a standard PHANTOM Premium 3.0 force

feedback device is used. The 3.0 is a large haptic device that accommodates free arm and hand motions about the shoulder as the center of rotation. The 3.0 base uses cable-capstan drive trains. Its motors are equipped with co-located position sensors. The base motor (primarily responsible for left-right motions, or x motions) is grounded. A cable transmission converts motor rotations to the rotation of a large disk which carries the rest of the mechanism. The second and third motors (primarily responsible for up-down and in-out motions, or y and z motions) ride on a ring mounted on the large disk, using a cable-capstan drive. The second and third motors control the position of two linkages in a four-bar parallel linkage design. The sensor-motor packages measure the endpoint position and provide force feedback in three translational degrees of freedom. To provide torque feedback, a proprietary instrumented gimbal is attached to the last link of the PHANTOM (the link is called the shin). This instrumented gimbal encapsulates three additional sensors and actuators in a compact package. The gimbal provides torque feedback in three orthogonal rotational degrees of freedom. It allows largely unencumbered movements of the hand with the wrist as a center of rotation.

The end effector takes the shape of a handle and measures less than an inch in diameter. It is roughly the size and shape of a large permanent felt tip marker. It sports one switch which can be programmed to produce visual and/or haptic effects. The 6 dof device is driven by a 6-axis power amplifier box and interfaces to a Pentium based Windows NT computer via a PCI controller card. Low level communications between the PC and the PCI card are handled by the PHANTOM Device drivers (PDD). The PDD maintains a 1 kHz servo update rate to ensure stable closed loop control of the device. The device kinematics and other robotic calculations are handled within the PDD. High level force and position calculations are provided by the the GHOST Software Developer's Kit (SDK). The SDK provides a high level C++ programming interface for generating haptic effects. Haptic effects handled by the SDK can be based on geometry (such as point haptic exploration), or on force-time profiles (such as sinusoidal vibrations and jolts), or the user can define their own custom force fields [24]. Currently the SDK only supports 3 dof forces based on point haptic exploration. Using newly developed extensions for the 6 dof system, custom torques can be superimposed onto these 3 dof forces. These extensions allow a 6 dof application to read a 4x4 homogeneous transform describing the global position and orientation of the end effector, calculate 6 dof forces and torques in the application, and command these 6 dof global forces and torques to the device. Force calculations are based on applying a simple repelling force at any point that is 1 voxel away from contact with objects in the environment.

It became clear that a large workspace directly translates into large arm movements, which can become tiring after a short time. Another observation is that torques in the roll direction are more readily felt than torques in the pitch and yaw directions because of excitation of tactile as well as kinesthetic sensors in the hand in the roll direction as opposed to kinesthetic sensors in the other directions. It seems that there are two distinct markets that require different workspace sizes: a training and ergonomic analysis market, where a big

workspace is required, and a design analysis market, where a desktop based workspace is more appropriate.

2. The Freedom 6S Device [20] (MPB Technologies, Canada) is a competitor to the Phantom with a workspace of 22cm x 24cm x 22cm. Unlike the Phantom this is a 6 dof device, both input and output. It doesn't map to one model of the Phantom though. It has a maximum force of 2.5N and 125mNm torque. Its resolution is the same as the Phantom at 0.02mm. The stiffness is around 1.3N/mm and the inertia of the tip is 150g. Like the Phantom it has a 1kHz update rate.

   A device of this nature inherited from telerobotic master arms and its design was much inspired by early systems such as CEA's MA-23 force reflecting manipulator, a project that was headed by J. Vertut and JPL's FRHC designed by J. K. Salisbury. A worry was the limitations of electromagnetic actuators. Again it was decided to initially use existing core-less DC motors although they were clearly sub-optimal for this task. Consequently, accurate static balancing was required for all six degrees of freedom. distal stage. Taking advantage of the geometric properties of a four bar linkage, proper dimensioning allowed to locate an invariant center of mass on the axis of Motor-1, thus realizing static balancing. The mechanical requirements are (1) static balancing, (2) uniform dynamic response both inertially and structurally, (3) wide dynamic range. (4) work-area compatible with a resting elbow posture, and (5) low visual intrusion. Item 4 meant that the design had to adopt a wrist partitioned arrangement which in turn implied remotization of actuation. Thus partitioned into a grounded positioning stage and a distal orienting stage , the design could be better analyzed. For the positioning stage, a pivoting four-bar linkage was found which could achieve both static balancing and uniform inertial properties. It was first observed that mass concentration would occur at the actuators and at the wrist. Static balancing required that the center of mass remained invariant under any movement of the device. Uniform and minimal inertial properties was achieved by placing each actuator such that each would experience an inertia dominated by just one single actuator plus that of the distal orientation stage, in all three directions. Ignoring the links, the moment of inertia experienced by Motor-1 is due mostly to Motor-2 and to the distal stage. Motor-2 experienced a moment of inertia due mostly to Motor-3 and the distal stage, and Motor-3 experienced the operator inertia and that of the distal stage. Taking advantage of the geometric properties of a four bar linkage, proper dimensioning allowed to locate an invariant center of mass on the axis of Motor-1, thus realizing static balancing. The design was also considered from the view point of its structural response. This follows from the observation that temporal resolution of the sense of touch, while not being as keen as that of hearing, is nevertheless sensitive to the high frequency details of the force applied. This is exacerbated by the fact that force feedback devices face conflicting requirements.

   The commercial version differed from the laboratory prototype in a number of ways. The use of the box design for the links was generalized resulting in a much cleaner feel. It integrated position sensing in the driven joints,

making it easier to achieve high control stiffness. In the commercial version, this parallel orienting design was found to be too costly to manufacture, given other requirements such as resistance to abuse and was replaced by a more sturdy serial structure. Instrumentation was also much improved. Since by static balancing, the device operated equally well under any orientation with respect to gravity, it was possible think of attaching it to a isometric orientation stage that would provide an infinite orientation range, so-to-speak, by servoing orientation so as to always keep the distal stage within its range of motion. The workvolume is 120 x 180 x 160mm; the range of the orientation is 90 x 100 x 120 degrees; the peak Force and Torque are 5N and 300 Nmm; the displacements resolution is 0.02 mm; the angular resolution is 20 seconds of degrees; the resolvable forces and torques at the handle are 0.01 N and 0.7 mNm; the electro-mechanical bandwidth is 200 Hz in all directions; and the inertia perceived by the user is less than 100g.

3. The Palmtop Display for Dexterous Manipulation [23] (PDDM) (ATR Communication Systems, Japan) is a haptic device for use in Virtual Space Teleconferencing system. The device itself is a linkage system much like the phantom. The end-effector is a palmtop display device. Manipulation of the device takes both hands, one on either side, with the visual display showing a local view of the scene to allow the user to look where he is working instead of looking at the large projection. Ideally, this allows for less of a need to eye-hand coordination and the awkwardness of manipulating an object in space from a distance. The disadvantage of the local display is associated with depth conflict. This can ruin the stereo sensation and cause the user to drop out of immersion.

   Palmtop displays have been extensively studied, but most of them simply refocus information in the real or virtual world. The palmtop display for dextrous manipulation (PDDM) allows the users to manipulate a remote object as if they were holding it in their hands. The PDDM system has a small LCD, a 3D mouse and a mechanical linkage (force display). When the user locks onto an object in the center of the palmtop display, he can manipulate the object through motion input on the palmtop display with haptic sensation. The goal is to obtain an intuitive interface for manipulation in the VSTC. The PDDM is proposed as one solution. A trial PDDM was built with an Ultra Sonic Motor (USM) force display. The palmtop display of the PDDM is a 4-inch LCD (resolution 220x480 dots, weight 350 gf) and has two push-button on the back. The user holds both sides of the display and presses the buttons with the left fingertips. Here, only one push-button is used to shift between the observing and the handling phase. A magnetic position sensor is installed on the side of the display. This sensor provides the orientation and position of the palmtop display. The display is connected to a 6 dof USM force display. 3 of these are actuated with USM and generate a force in any direction at the display. The others are an unactuated 3-axis universal joint at the end of the arm. Thus, the system can't restrict rotating torque. Each arm's length is 30 cm, and the workable volume is a half sphere with a radius of 60 cm. In this configuration, the force display can generate a maximum 5 N reacting

and sustaining force in the volume. A USM works with frictional force, which completely differs from an electric magnetic motor (EMM). It can generate a high sustaining and rotating torque in one component and the output can be controlled from a full sustaining torque to a full rotating torque continuously. The rotating torque per unit weight of our USM is almost twofold that of a typical EMM. In addition, a USM can output maximum torque at lower speed (1.3 Nm at 60 rpm), so it does not need reduction gears, which reduce back driveability. Furthermore, since a USM makes no magnetic noise at all, it does not affect the results from the magnetic position sensor introduced in many VR systems. An electrical touch sensor is installed on the back of the palmtop display. When the user holding the display, all joints can be freely rotated. When the user releases it, the system detects it and immediately locks all joints. It is an important feature that the PDDM keeps the same position as that where the user released it. The system does not support this now, but the weight and the inertia of both the display and the mechanical linkage can be reduced to a level sufficiently small with dynamic impedance control. It is expected that fatigue in the user's arm through extended use can also be reduced. Both the PDDM and the large screen display the same virtual world. The SGI workstations contain a master server process, a drawing process and communication processes. A PC is used to control the force display and communicate with the server process through the ethernet. The drawing process can update an image in real.

4.2.2 Parallel arm structures

Parallel robots, which are closed kinematic chain mechanical systems, promise advantages over serial robots (open chains) in terms of less moving inertia, faster and stiffer motion, and much better force-to-weight ratio. While control of serial robots is a well established field, the derivation of dynamic equations of motion for parallel robots suitable for model-based controller design is still an open research topic because of the complexity of the kinematics, dynamics, and control analyses. Illustrative examples of haptic devices based on parallel architectures include the following:

1. The Haptic Interfaces (University of Colorado) is an interesting device because it is configurable from 3 dof to 6 dof. When at 3 dof it resembles a pyramid with three rods connecting at a point where the user holds. When configured for 6 dof there are three rods attached to each end of a stylus, similar to a Stewart Platform. They allow the device to have an accuracy similar to that of the Phantom with comparable maximum force and force resolution. The authors state that "Motor and rod are smoothly coupled by preloaded rolling elements, eliminating backlash and cogging, yet providing an extremely stiff link which enables high bandwidth transmission of forces to the user's fingers."

2. The Delta Haptic Device [24] (Forcedimension Co, Switzerland) is a force-feedback system that meets the high standards required for industrial applications by combining high strength, high stiffness and high sensitivity. This system is based on the patented Delta robotic structure, which provides three

translational dof. A dedicated mechanical wrist plug-in provides 3 rotational dofs and is based on the Paramat structure. All of the dofs are fully active and generate forces and torques that are way beyond the ability of currently available devices on the market. Dedicated electronics and signal processing provide the high-quality control required for credible force rendering.

3. A 6 dof joystick [25] is proposed as a haptic device based on the bilateral Stewart platform (BSP). This desktop joystick was developed based on the generalized end-point force reflection pioneered by Bejzy [26]. The device receives feedback from a force/torque sensor (F/T) mounted at the wrist of a slave arm. The teleoperation system is configured such that the BSP accepts position demand signals and performs two kinematic conversions before transmitting the required slave arm joint angles to the robot joint servos. In the reverse direction, the BSP controller receives end-point F/T signals from a slave arm wrist mounted sensor.

   The BSP is designed such that the feedback is not distorted by the mechanical characteristics of the mechanism. The input device mechanical structure is built on a six-leg parallel link structure of the Stewart platform. The platform was selected for its simplicity and robustness. The Stewart platform was implemented as a hinged link structure with drive motors connected directly to the base of each link. The top of each link connects to the platform upon which the joystick handle is mounted. The measurement of the joystick position is provided through optical shaft encoders. The whole mechanism is covered by a flexible material. The operator acts on the stick which can be operated in position, rate and preferred axis modes using outboard switches. The operation volume which is rather small allows 150 mm in the X and Y directions and 80 mm in the Z direction.

   The authors report that the BSP has been integrated into a telerobotic control system that provides a number of operating modes ranging from bilateral teleoperator control through shared control to robotic control. The BSP has been used to control a number of slave arms.

4. The Haptic Master [9] (Iwata Lab, Japan) is a 6 dof desktop device based on the Stewart Platform. The simplicity of the structure, basically built with three pantagraphs in parallel, give the device the advantage of being inexpensive. Other advantages of the design are associated with its ability to carry large payloads and its compactness. The drawbacks are its limited working volume and its inability to backdrive.

5. A modified delta mechanism [27] designed as a novel 6-DOF haptic interface characterized by compactness, quick motion and relatively large workspace. The authors adopted a hybrid parallel mechanism with uncoupled substructures for position and orientation. Six motors (3 at the base and 3 at the top) with harmonic gears are utilized to drive the different links. A compact 6-DOF force/torque sensor is mounted with the joystick. The modified delta mechanism serves as a positioning subsystem at the base. The mechanism allows for 3-DOF. On top of this substructure a five-bar 2-DOF orientation

subsystem is employed. Both these mechanisms have no singularities in their workspaces. Finally, on top of the five-bar mechanism, is mounted a 1-DOF joystick with a force/torque sensor. This structure can be used to realize a wide workspace (the positioning range is a sphere 75 mm in radius; the orientation range is 120-160 degrees for each axis). The authors emphasize the fact that the parallel structure ensures also a quick ability motion.

4.2.3 Exotic robotic structures

The third class consists of traditional heavy master arms with anthropomorphic, exoskeleton, or humanoid [28] architecture. Illustrative examples of this class include the following:

1. The whole arm manipulator [11] (WAM) is an impressive humanoid arm using 4-dof. In particular, the authors present a model that is used to control a manipulator to minimize interaction forces in the initial phase of acquiring unknown objects in unstructured environments. Using modeling and parameter estimation, the authors were able to vary the impedance of the robotic arm from being very gentle to very stiff just like human fingers. This was achieved by employing the model based approach plus the intrinsically low impedance of the WAM reaching impedances similar to a human arm. The authors describe the design and implementation of a low impedance controller and examine its performance in an exploration task that requires contact with an unknown surface. The control law used computes the Joint Torque with the Contact Torque as function of the dynamic model of the arm, the friction, and the characteristics of parameters of the controller. The dynamic model includes gravity and dynamic terms, the controller represents a set of error-based feedback terms, and contact torque represents the torque at the joints created by the forces at the tip. Each term is developed to arrive at an equation which allows the computation of the joint torques needed to follow a trajectory. To experiment their model, the authors have mounted an ATI force sensor at the end of the WAM arm covered with a hemispheric cap. Thus the effects of impedance on position accuracy are quantified.

2. A simple platform type master arm [29] based on the kinematic of a human arm and a master hand. In both systems, force reflection is achieved via the use of pneumatic actuators. These actuators are used to generate necessary force for feedback. The device is integrated with Korea Institute of Science and Technology (KIST) humanoid robot as well as with a graphic simulator to teach the robot. The master arm is warn by a human that employs it for teaching the KIST robot. The investigators' goal was deriving joint angle commands from human's posture so that the robot follows human motion exactly. Arms joint angles, waist joint angles as well as neck joint angles are calculated using twenty two analytically derived equations for the robot posture. These equations use input from seven sensors; two FasTrack sensors attached at each shoulder and the other sensors are attached at each arm's elbow, hand and at the back of the head. To control the KIST robot, visual

information is provided by a CCD camera attached to the robot head to the human through HMD.

The structure of the master arm is similar to a Stewart Platform with linear actuators and 6-DOF. One platform is for the shoulders and another for the wrist. The two platforms are connected by two links, which represent the elbow joint of the slave robot. The master hand is composed of four fingers which have 2-DOF each controlled by two pneumatic actuators. The details of the design of the master arm and master arm hand are included in the paper by Lee et al. [30]. A brief description of the KIST humanoid robot is given in the paper. The authors report that the KIST design is one of the closest to human with 9 DOF for each arm, 10-DOF for each hand, 3-DOF for the waist and 2-DOF for the neck. It is shown that the KIST has an automatic path planner for which the computation load is too long. Another difficulty resides in the fact that only the end effector location is the same as the masters hand location. Because of these two difficulties two ways of teleoperation were developed by the authors: 1) matching the end effectors location of the robot with the masters' by solving inverse kinematics, 2) matching the KIST robot with the master's.

3. The WYSIWYF Display [31] (Carnegie Mellon University) consists of a Puma robot used as a haptic device. A portable TFT display is placed in between the user and the device. On the backplane of the display there is a color CCD camera used to track the devices location which has several markers placed on it to aid this process. The objects have grab-points attached to them in order to match the physical grab-point that is the devices end-effector. This allows the models to be manipulated through these artificial grasp areas. One point of interest is that the device, and the background, are all painted blue so that the users hand can be extracted using Chroma Keying techniques and placed in the visual display instead of a graphical version being generated. Among the experimental applications of the above described haptic device is that where it was used as a robotic master arm to control a dual-arm space robot. The set-up is used as the controlling system (master-arm) on the ground part of the experiment. The ground part has also two PCs, one for controlling the haptic interface device and the other is used for simulating the 3-D graphics model of the space system. The part of the space system is composed of a dual arm manipulator system and two controlling computers. The manipulator system has two slave arms with 7-DOF in each arm. A 6-axis F/T sensor is mounted at the tip of each arm (at the wrist) of a multipurpose BarrettHand.

   The teleoperation environment is based on virtual reality that uses the haptic interface and the 3-D graphics. To avoid the collision of the slave system with obstacles in the dead space the concept of virtual radar [32] is used to transpose the obstacle avoidance information to the end-effectors. The authors have also implemented the concepts of "virtual grip" and "virtual ball" in their system to help the operator teleoperate the slave system comfortably [33].

4. The Sarcos Dexterous master [6] (Sarcos Inc.) is a high inertia device with 10 dof and a complex dynamics structure. Eventhough it is worn around the

users arm, and is anthropomorphic, it is not an exoskeleton. It is a grounded device that is capable of producing external forces.

4.3 Compliant end effectors and force Sensors

H. Kazerooni et al. presented the design, construction and control of an active end effector [34] which can be attached to the endpoint of a commercial robot manipulator. The development of an impedance control device causes the end effector to behave dynamically as 2-D remote center compliance where the compliancy can be modulated by an on-line computer. The active end effector is a five-bar linkage system with 2-DOF powered by two dc actuators. The links are made aluminum 6061 and the motors are dc-brushless with direct drive. The direct drive system is rigid and allows for a wide control bandwidth. A wide bandwidth piezo-electric based force sensor is located between the endpoint of mechanism and the end-effector gripper. The force sensor is preloaded by a clamping bolt and measures the force in two dimensions. The end-effector is attached to the robot by a simple.

Of the applications of such a device is robotic assembly and manufacturing. The author details the use of the end-effector in a pneumatic grinder, used for deburring a surface, to show the importance of having the modulation of the compliance in the system. He explains that it is necessary in such an operation for the end-effector to accommodate the interaction forces along the tangential direction (i.e. small impedance value). The author gives also the details of the dynamic model of the effector as well as the results of the testing of the system.

S.E Salcudean et al. presented a new teleoperation system based on parallel actuation [13, 12]. It is proposed that the teleoperation slave be a coarse-fine manipulator with a fine-motion wrist identical to the teleoperation master. The fine motion technology chosen for this work is Lorentz magnetic levitation. The wrist employed was designed and built at the University of British Columbia (UBC) along the principles described in [12]. The issues of coordination, sensing and control that arise in the operation of the proposed force-reflecting teleoperation are also described in this paper.

A wrist-level coordinated force approach emulating a massless rigid link between the master and the slave wrists is employed. Feed forward of sensed hand forces to the slave and environment forces to the master is used to improve transparency. It is shown by simulations and experiments that such feed forward leads to poor stability margins in contact with stiff environments, unless substantial damping is added to the system. It is also shown by simulations and experiments that the coordination error can be substantially reduced by such sensed forces feed forward. If a simple adaptation of damping to sensed environment forces is applied, free-motion tracking performance and stiff contact stability do not have to be traded against each other. This approach is similar to the estimated impedance feedback but does not require environment identification. Coarse positioning motion is achieved by using rate control to move the transport (coarse) robot in the direction pointed at by the master, whenever the teleoperation master is outside a pre-determined subset, or rate-control dead-band, of its workspace.

The wrist is an in-parallel actuated device in which six Lorentz forces are generated between two rigid elements a stator and a "flotor". Only wires for power and

sensor signals connect the two, the lighter flotor being actively levitated. The maglev wrist has 120(symmetry. Three horizontal and three vertical coils are imbedded in the flotor. Each coil fits within the gap of a matching magnetic assembly attached to the stator. The flotor's horizontal plate has holes to allow supporting posts to hold the stator and the magnetic assemblies attached to it. The UBC wrist fits within a cylinder with diameter of 132 mm and height of 110 mm.

The paper describes also the details of the uncoupled coarse-fine control and the experimental validation of the system showing excellent performance in free-motion tracking as well as in contact tasks.

In his work Sherry Draisey [35] presented a 6 DOF force sensor concept with an initial view of a sensor that has been designed to function as a vibration control transducer for the deploying and deployed, low frequency NGST sunshield. The sunshield on-orbit dynamic and control characteristics will have an impact on mission planning and life of the spacecraft. The sunshield deployment (particularly if it is an inflatable structure) will potentially need a control philosophy. A force sensor as the measurement device for the control system provides a weight and location effective solution. It can be located at the spacecraft end of the sunshield, and does not have to form part of the deployment philosophy. This paper includes initial and conservative assumptions about the NGST system, to derive force sensor requirements for both launch and operation use. The sensor design has been in support of development of a sunshield vibration control system. It assumes the sunshield will have the capability to be repositioned, in the interests of saving overall fuel consumption. The justification for the sunshield vibration control system itself is the significant reduction in settling time which will result from these repositioning slew maneuvers. This savings will be used to increase operational data gathering time.

The patented sensor concept uses non-linearity to create measurable mechanical system frequency changes for use in sensing the DC (and very low frequency loads). The use of a piezoceramic element as the transduction unit allows for the wide dynamic range of the sensor, as well as providing the oscillator forcing element. Both operational and launch configurations have been evaluated in the force sensor concept of the study. The finite element analysis, and preliminary prototype testing suggest a minimum measurable moment of 10-3 ft-lb can be achieved, with a maximum measurement value of 100 ft-lb. Survival loads of up to 660 ft-lb of launch moment can be survived. The effects of thermal gradients on the sensor have been carefully considered in the design configuration, but only cursory analysis on this has been done to date. The design description is given below.

The design description of the above sensor is based on mechanical configuration non-linearity; stiffening under load. The non-linear effect is created by radially arranged, pin ended struts.

Non-linearity Implications: (1) allows for use of frequency shift measurement as low frequency range measurement method, (2) complicates overall control system design, (3) provides more stability when used in large load handling situation, and (4) provides shock/overload protection for transducer. The mechanical design has been kept flexible to allow for the use of an overload stop, without unreasonably high precision tolerances (.01" clearance to stop). The mechanical design of the sensor shows

- 6” OD steel cylinder, wall thickness .25”
- one end of cylinder closed by a 0.06” bottom steel plate
- other end partially covered by upper loading plate, .04” thick, 4.5” diameter
- upper plate held in place via preloaded bolt, passing through, the piezo stack, into the bottom plate
- outer edge of loading plate connected to outer housing via 8 pin ended struts
- sensing transducer/oscillator is a piezoceramic cylinder 1” high, 0.5” diameter
- structural weight (exclusive of electronics) 1.6 lb.

There are two electronic modes of operation for the sensor, each overlapping in frequency, allowing for cross-correlation accuracy. The low frequency mode (10 Hz) relies on measurement of system frequency shift (nominal system frequency is 200 Hz), resulting from applied load. The piezoceramic element acts as both an exciter and transducer, in this mode. The high frequency mode (¿10 Hz) uses conventional strain (amplitude) sensing to determine applied load level.

William A. Lorenz [36] describes the design and construction of a new class of force sensors. The sensor allows larger displacements in response to forces than present commercial sensors, and orthogonalizes the axes to be measured by a combination of the properties of the flexure and the displacement sensing mechanism. Two methods of measuring the displacement, optoelectronic and electromagnetic are discussed. The first, is optoelectronic in nature where infrared LED/photodiode pairs are used as the sensor element in place of strain gauges. Deflections of the order of millimeters are measured as compared to strain gauges where microns can be measured. The author reports that photosensors suffer from the same problems as strain gages including nonlinearity and temperature sensitivity. The second displacement sensing device is electromagnetic in nature. The design utilizes 2 zigzags of conducting material which side by each other. An AC current is supplied to one zigzag,inducing current in the other. It allows large displacement measurement and is easy to construct. Criteria for material selection and dimensioning are given, and experimental results are reported in this thesis.

4.4 Stereoscopic Visualization

Stereoscopic photography is a technique, which is known for a quite long time, and is commonly used by amateur and professional photographers to capture 3D information of a scene. With the wide recognition of importance of virtual reality tools in telerobotics, teleoperation, telesurgery, and telepresence, more researchers turned their attention to 3D video generation and visualization techniques. In the following we present a classification of visualization systems based on (1) used equipment, and (2) implied delay and bandwidth.

4.4.1 Classification of visualization systems based on used equipment

There are a variety of 3D-video formats, interlaced, page flipping, sync doubling, and line blanking. Each format requires different techniques and/or equipment for generation and visualization. Furthermore, they have different robustness characteristics under MPEG compression, and image/video resizing. For a detailed and comparative discussion on these modes, see the online document, Eye3D Manual [37], There are a variety of different ways to generate 3D video content which are the following:

1. Parallel camera configurations [38], can be used to observe with high accuracy a 3D object under magnification and depth. This is a very commonly used technique for 3D video generation. Computational aspects are simpler than the tilted case. However, it has problems especially with the near stereoscopic viewing. Most of the time, some sort of video mixer may be required to convert two video streams into a single synchronized stream.

2. Tilted camera configurations [39, 40] produce more accuracy in the horizontal direction than in the vertical direction compared to the case of parallel camera configuration [41]. However, this problem can be overcome by using different horizontal and vertical scaling factors. Furthermore, this configuration provides a larger area of stereoscopic vision, such that the total area for 3D display is more, the depth resolution is enhanced, and near stereoscopic viewing is better than the parallel configuration. On the other hand, computational aspects are more complicated and demanding compared to the parallel case. Again, most of the time some sort of video mixer may be required to convert two video streams into a synchronized single stream.

3. NuView 3D adapter consists of two LCD-shutters, a prismatic beam splitter and an adjustable mirror. Watching through the Nu-View, while it is switched off, one will see two images. The mirror/prism system puts the camera lens into the center of the light rays of a left and a right eye view. The shutters allow the camera lens to get only one of the views at a time. The adaptor is connected to the video-out port of the camcorder. This way the shutter can sync to the recording (50 or 60 Hz). The drawbacks of this approach are: (1) when zooming to the widest angle parts of the NuView adapter may appear in the frame, producing a dark border and (2) it produces some ghosting in hi-contrast scenes. However, besides these side effects, it is a simple and practical solution to 3D video generation. See the online documentation at [42] for further details.

There are basically two major classes of 3D visualization techniques. These are shuttering glasses and head mounted displays which are described as follows.

- Shuttering glasses enable to view stereoscopic images. The glasses alternately "shutter," i.e. block, the viewer's left, then right, eyes from seeing an image. The stereoscopic image is alternatively shown in sequence left-image, right-image in sympathy with the shuttering of the glasses. At low refresh frequencies, the user can experience the annoying phenomena of flickering which

can effect his or her ability to control the robotic arm. However, most of the available monitors and display adapters can support refresh frequencies equal or above 120 Hz at resolutions of 1024x768 or above. Therefore, 3D visualization with very high detail is possible with most shuttering glasses. There are indeed numerous such papers [43, 44], which demonstrate the effectiveness of shuttering glasses in 3D visualization.

Just for illustrative purposes, the "Eye3D Premium" shuttering glasses can support resolutions (in pixels) up to 2048 x 1538 at 120 Hz, and 1856 x 1392 at 140 Hz. These specs are only found in high-end monitors, namely for high resolution and high refresh rate 3D visualization; the existing shuttering glasses technology is more than enough.

- Head mounted displays [45, 42] provide a much larger virtual monitor size for the user, usually in the range of 2 meters large. However, their main disadvantage is that their resolutions are either VGA or SVGA (at least the ones which are commercially available during this period of time). They are more comfortable to work with, forces to use to see the 3D object and nothing else, and there is no problem of flickering. Most of them support the INTERLACED 3D video format, but not the so called ABOVE/BELOW format which is robust under video compression and resizing. Most HMDs also support page flipping, but this requires special drivers for each display adapter/chipset.

  Some HMDs are also equipped with ear-phones and head trackers, like the "hiRes-900 + InterTrax2" set available from Cybermind Interactive, the Netherlands. But compared to shuttering glasses, they are a factor of 10-20 or more times expensive, yet they are limited to SVGA resolutions.

4.4.2 Classification of visualization systems based on delay and bandwidth

Dealing with network transmission delays and limited network bandwidth is a fundamental research problem in telerobotics. Introduction of time delays into a general control system poses problems related to stability and performance. This is also true for a telerobotics system. It has been reported that operators confronted with time delay had a tendency to move by small increments and wait to see the results of their motion, i.e. using the "Move and wait" strategy. This approach considerably reduces the overall system performance.

In [46], a telerobot at Jet Propulsion Labs (JPL) is described. It has been reported that a 5 milliseconds delay is too small for the operator to notice. This is called as the "normal" mode and it provides high fidelity and stable performance. However, as the delay is increased up to 1/4 seconds, it starts to be noticeable by the operator, and this starts to affect his cognitive task and motion planning. Delays as small as 1 second, considerably degrade the operator's performance.

For some space applications it is desirable to control the space manipulator from Earth. This introduces unavoidable time delays in data links between the master and slave systems. Round-trip communication times can be as large as 6 seconds. When faced with such a large delay, the operator needs some support to overcome

the lack of frequent interaction with the remote site in an attempt to improve the timing and correctness of the task execution.

In the following, we briefly describe three visualization approaches. They indeed differ on the way that they address the issues of delay and bandwidth.

1. One way Image/Video Transmission Based Methods [47, 48, 49, 50, 51, 52, 53, 54] consists of sending static images or live video from the slave robot location to the display(s) at the master arm location. In this simplistic approach, the only effort done to reduce transmission delays is to compress the static images or use some video compression techniques. In any case, there will be a long delay between the actual slave scene and what is seen at the master station. This is a feasible solution only if the master station can issue high-level operator commands and there is a local controller at the remote slave location to interpret these commands. A typical command might look like "Move 5cm in the North direction", "Open the gripper", etc. The operator interaction in such systems is usually minimized by the use of short actions that automatically executes at the slave site without involvement of the remote operator.

2. The Model-Based Methods [55, 56] consists of using graphical tools to superimpose a picture of the slave robot scene with a generated background image at the display of the master robot site. The ARGOS (Augmented Reality through Graphic Overlays on Stereo-video) project is one example for this approach. Transmission of static images and/or live video generally introduces delay and consumes a significant portion of the available bandwidth. This is also the case even if advanced image and video compression techniques are used to overcome the effects of delay and low bandwidth. The model based methods use Augmented Reality (AR) or Virtual Reality (VR) tools to draw the slave robot arm picture on a real or computer generated background image at the master stations display unit. For this a complete and accurate model of the slave robot arm is used at the master station. The slave station is supposed to send position and orientation parameters of the slave robot arm to the master station in a continuous manner. Based on these received parameters, the master station can draw an artificial image (graphically computed) of the master robot arm based on the available model. Compared to sending the whole image, sending a couple of position and orientation parameters is more economical, which reduces delay and doesn't consume large bandwidth. On the extreme side, one can even model the whole robotics scene, and regenerate the same scene artificially at the master station's display unit. Since reducing delay means better performance and more realistic operation, the operator works in a highly interactive model-based environment. In other words, the operator is not forced to issue only high level commands to be able to operate the slave robot arm.

3. The Predictive Methods [57] consists of using a predictive model to overcome the effects of delays. It is no surprise that predictive methods are also utilized in telerobotics systems to reduce the effects of transmission delays. In some model based applications, even transmission of the model parameters over

the communication channel may take long time. Consider for example, the operation of a telerobot in Australia from a master arm located in Saudi Arabia. In this case, the master station needs a prediction filter for the slave robot arm parameters. The prediction filter will continuously receive delayed slave arm model parameters, and generate predicted actual model parameters. Then either using Augmented Reality or Virtual Reality tools, the slave robot arm picture can be drawn on a real or computer generated background image at the master station's display unit by using the parameters output from the prediction filter, not by using the received delayed ones. As in the model based case, predictive methods increase level of operator interaction and gives a more realistic sense of teleoperation.

4.5 Networked and Internet telerobotics

Paolucci et. al [58] discussed teleoperation on packet switched networks. The experiments are carried out with varying values of data loss, delay and jitter to evaluate the performance of teleoperation system. It is shown that when the packet size is increased from 64 to 1024 bytes, the network delay is also increased from a mean value of 5.6 ms to 13.4 ms with a minimum value of delay equal to 5.4 ms always present due to computational overheads. LAN performs well even in the presence of traffic caused by other users until the total network congestion, which, of course, causes the system to be completely unpredictable. The performance is more degraded with added delays and jitter on MAN (Metropolitan Area Networks) possibly due to the presence of different routers and queuing algorithms.
Teleoperation performance tests are carried out on a network simulator. An important result is that the operator performance is quite insensitive to a fairly small data loss. Also if the same quantity of data is supplied but spaced at regular intervals, an increase in the operator performance is observed.
Introduction of delay causes a decrease in operator performance almost linearly. Jitter produces a disturbance in velocity due to varying interval between samples. Introduction of a buffer can decrease the jitter but at the cost of increased delay. A tradeoff can be negotiated between the two parameters.
A predictive algorithm utilizing the model of the actuator is applied to get better performance out of the telerobotic system. The model is located at both master and slave sites. Master site model gives immediate visual feedback to operator enhancing his performance while the slave side model is used to periodically update the parameters of the actuator by comparing the predictive and actual outputs. Actuator dynamic model is obtained using least square recursive estimator with an exponential forgetting factor.
Component-based distributed control for tele-operations using DCOM and JAVA is discussed by Yeuk et. al. in [59]. A model-based supervisory control is proposed at the remote site that is the foundation preparation project at the foot of a volcano in Japan. In order to fulfill certain requirements such as high level of robustness, flexibility in deployment of the complete system and ease in upgrading the system, a component based distributed control of the system is used. A supervisory control is implemented at the remote site which is based upon the remote environment model. Internet is used as communication backbone and JAVA/DCOM are employed to

realize component infrastructure. Complete isolation from the network protocol is obtained using components. JAVA and DCOM are used for component development, each one having some unique characteristics. JAVA is basically an operating system transparent software language but the use of Virtual Machine makes it a bit slow than OS optimized compiled DCOM objects. So DCOM is used in all interface components except Path Planner GUI and Database interface which are written in JAVA due to simplicity with no hard real-time constraint. DCOM/ActiveX Supervisory Control Server is the heart of supervisory system and it maintains communication with vehicle objects, direct manual control as well as sensor integration server components.
Video stream as well as a 3D graphical model of the current remote environment are provided to the operator. Generally there should not be much difference between the two, but if there is, the Supervisory Control will transfer the control to the operator to initiate necessary actions.
They have further extended their work in [60]. Here the feedback is provided by two paths, one from the GPS (Global Positioning System) data and the second one from the visual feedback. The visual feedback is generated by the images from a camera at the slave site. Here the images are snapped and from the remote environment models which are identified by Visual Enhancements(VE) , the position $\underline{X}$ of the vehicle is determined by minimizing the following error function based on the difference between the vehicle position coordinates obtained from GPS and the visual feedback.

$$E = \sum_{i,j} E_{ij}^2 = \sum_{i,j} K_{ij}[X_p(i,j) - P_i Tcw_i Twf_i(\underline{X})]^2 \quad (1)$$

Here P_i, Tcw_i, $Twf_i(.)$ are coordinate transformation operators and K_{ij} is determined from the reliability of the measurement. This information is used in supervisory control and is also sent to the master site to invoke operator intervention if any critical error occurs. The system is stated to be sufficiently robust against the addition of white noise in both robot actuator and camera planes.
A collision-free Multi Operator Multi Robot (MOMR) teleoperation scheme is proposed by Chong et. al. in [61]. Effect of time-delay will cause more severe problems in MOMR systems than in Single Operator Single Robot (SOSR) systems due to unpredictable nature, in local display, of the slave arm under the control of remote operator. Due to the presence of long distance in the positions of operators, one can not get immediately the command issued by the other operator as a result posing the danger of collision in slave arms. This type of collaboration, known as unconstrained collaboration, in which each operator has the freedom to control his/her slave arm independently from the other slave arm, is very sensitive to time delays. Operator usually commits to a *wait-and-move* strategy in order to avoid collisions thus decreasing the productivity considerably.
Simulation experiments conducted using OpenGL and network delay simulator showed the occurrences of collisions even when a virtual thickness corresponding to the time delay is added to the slave manipulator model in local display. There were collisions even when there was no network delay because of human error. Authors suggest a new approach using velocity rate control that scales the velocity commands issued by the operator considering the relative positions of the slave arms. If they are too

near, the velocity commands will be scaled down, otherwise they will be sent as they are. However if the distance is too small, the velocity commands will become zero neglecting the operator completely. This approach avoids the collisions completely but ruins the maneuverability of the joystick because of scaling effect.
Martin Jagersand in [62] proposes an image based predictive display for tele-manipulation. To obtain a predictive display, normally system modeling is the primal part but in this method, there is no need of a-priori modeling, instead an image model is generated from the delayed feedback signals and the command sequence from the operator. Operator controls the robot by command signals $(x_1, x_2, \ldots,)$. After some time the real image stream $(I_1, I_2, \ldots)$ arrives. Due to delay, the operator is seeing image I_k at time $k+d$, where d is the delay. Author possibility is to estimate a function $\phi(x)$ online which approximates each image I_i on the trajectory such that

$$I_i \approx \phi_k(x_i),\ i \in \{1, \ldots, k\} \tag{2}$$

First the image is compressed and is represented in a lower dimensional space of appearance vectors using an approximately invertible image appearance function g such that $I = g(y)$. To obtain this compression KL(Karhunen-Loeve) basis compression is used. Then a function f is learnt such that $y \approx f(x)$. This function is initially unknown and can vary during manipulation due to unmodeled kinematics so it is desirable to continuously estimate f. A recursive Jacobian estimator is used to train f.
Once a reasonable number of images (100-1000) have been obtained, f is sufficiently trained. So the increment in x, the command signal, is used to estimate the change in visual appearance. This change is then used to predict the display by forming an image using the inverse KL approach. The method is suited for applications where the workspace is small and there are only few changes while moving the manipulator. It is particularly useful where we don't have geometrical models of the objects. It is argued that a table lookup approach is not suited for interpolating images in a real-time application as it will introduce jitter and will require more computation for the same dimension of the system. The algorithm requires large spatial data to generate good quality images so it may not be efficient in situations having greater spatial details.

An effective way of overcoming the varying time delays in bilateral feedback systems is discussed by Kazuhiro et. al. in [63]. It is already proven in literature that passivity can be assured in communication block by using scattering transformation. But variable time delay destabilizes bilateral master-slave manipulator even with scattering transformation and authors propose virtual time delay method to keep the time delay constant.
To understand the influence of variable time delay suppose that a communication block has a delay which changes from T to $T - dT$. If we transmit a sinusoidal signal via this block, the signal received after the change in time delay will be changed abruptly with shift of δu in the amplitude. This makes the communication block a time-varying system, the passivity of which we can't guarantee only with scattering transformation.

In the method proposed by authors, the network traffic is observed to get an estimate of maximum delay, T_c between master and slave. Velocity and force information are sampled at constant rate and are transmitted to the other side along

with the sampling time t_{send}. On the other end, the data is received after the time delay $t - t_{send}$ which may be shorter than the maximum time delay T_c. If this is true then these received values are held in a receive buffer until the delay reaches T_c. At this time, the received values are released from the buffer for manipulation. In this way we can make the apparent time delay of the system equal to virtual time delay which is constant thus guaranteeing the passivity of the system. Virtual time delay can be made to adapt to the traffic condition of the network so that it remains appropriate in all practical situations.

4.6 Real-time network protocols for telerobotics

In the presence of large transmission delays the slave arm is supported with a shared-autonomy under a sensor-based motion planning algorithm [64]. A sequence of fine motion is selected by operator watching its virtual motion implemented using OpenGL 3D graphics. The sequence is then transmitted to slave which executes its under sensor-based impedance control which provide robustness against position and orientation errors. The impedance control aims at making the closed loop slave control a linear, second order, position error, to compensate for the sensed force and torque at the slave wrist. The VxWorks real-time OS is used.

Networked telerobotic systems are subject to random delays in the closed loop control system which affect the stability and performance [65]. Robot control input and sensor data have strict real-time requirements and visual feedback requires high bandwidth. A robust control using μ-synthesis is proposed to handle worst case time delay. Evaluation is carried out for telerobot implemented over two LANs interconnected by a campus ATM backbone. Specification of Quality-of-Service (QoS) includes application timing, criticality, clock synchronization, and reliability. This is accomplished by using a constant bit rate (CBR) ATM connection allowing a tightly constrained transmission delay which is suitable for real-time applications. The OMEGA architecture with ATM networks are used to guarantee time service frequency at the computing nodes (QoS brokerage). The sampling intervals reported are 0.4, 0.3, and 0.2 for TCP/IP, raw ATM, and TCP/IP over ATM, respectively. It is reported that TCP/IP over ATM performs better than raw ATM. However ATM has guaranteed performance and better jitter control.

In a computer network, the communication delays and traffic capacity vary with flow direction and irregularly change with traffic conditions [66]. To avoid destabilization of a bilateral master-slave telerobotic system. A discrete time domain scattering transformation is proposed to guarantee the stability is based on transmitting control sampled at a constant rate together with its sampling time to account for the apparent delay. Here transmission rate is much lower control rate. Evaluation is done using nodes running VxWorks OS and interconnected by a LAN. Each packet has a payload of 10 bytes and sent every 10 ms. Overall delay was up to 0.94s and irregularly changes with flow direction (0.15 s and 0.45s). Here the apparent time delay is kept constant by the local controllers.

A supervisor control is proposed to reduce operator attendance. The implementation uses distributed software component design using DECOM [60]. Integration of web technology and the telerobot is addressed together with environment constraints for diverting lava flaws near a volcano in Japan. Slave robots is like a

bulldozer. This requires autonomous to reduce human attendance, robustness control, and flexibility to ease the task of upgrading the system. Here the operator infrequently and indirectly interact using a finite set of high level commands such as specification of destination to be used by a remote controller. A model-based vision system with GPS sensing are used. An OpenGL simulator was used for testing. A DECOM/ActiveX supervisory control server is developed using DECOM low level feature extraction components like segmentation, pre-filtering, measurement, verification, and feature matching with predicted parameters. The supervisor control uses a DECOM sensor integration server to assess prediction errors. The communication architecture uses the Internet as backbone because of its universal availability. Java and DECOM provided the needed flexibility. The distributed components were connected using the remote object invocation which allowed full isolation of component design from the network protocol. Java and DECOM provide a tool for building network-centric software with ease of remote incremental maintenance and updating. Object technology ease the mapping between local and remote object management. Java virtual machine provide an execution system that is isolated form the OS to achieve cross-plateform compatibility. A central supervisory control component interacts with the sensors, IP, onboard video camera components, and other service modules. Operator views 3-D model, control paths, and issue commands through the supervisor central control.

The impact of random delays in Internet communication [67] may have reduced impact on teleoperation performance and stability if a sensor-based autonomous loop is used to set up a stable contact and to maintain compliance with the contact surface. Under contact with environment the slave controller becomes independent from the remote operator. Local compliance at the slave robot contributes in reducing fine interactions while the operator controlling the motion using visual feedback.

Machine intelligence is useful at low level sensory control functions, precision, reliability and computationally intensive tasks [68]. While human are more suited for higher level perception and reasoning, task conceptualization, understanding the environment and dealing with unusual circumstances. Handling of unstructured environment lead telerobotic to be built on the top of tasks that are assistive, low-level, and manual to directly support the human operator in harmony with more independent tasks that only constrained by the limitation of machine intelligence, a system with an ecological control that is consistent with the director/agent metaphor. In the proposed ARGOS system intelligence is relevant to human component that interacts by using a partial knowledge of the remote site in a semi-automated control of the telerobot.

The most standardized approach for sending information over the Internet is through the use of the highly reliable TCP/IP sockets [69]. It is based on a connection-oriented between a client and a server stations. The drawback is the delay jitter in the arrival rate of packets that were generated in a synchronous way which is caused by network capacity and traffic intensity. Inter-arrival packet delay varies from a few ms to a few seconds, where the last delay is rather rare. An average delay for a small packet ranges from 50 ms to 100 ms is common case over the US. Asynchronous packets do not preserve their chronological order. TCP/IP can be reasonably used for packet with 256 bytes with a 10Hz sampling rate. Telerobotic control is implemented using a client computer running under Unix and connected

a server computer running under VxWorks through an ethernet network. The real-time multitasking system allow spawning and running many tasks. By prioritizing the tasks the user can control the order of task execution and the amount of CPU time allocated to each task. Example of prioritized tasks are joint control, communication between client and server, socket communication, and communication with landmark system.

Telerobotics uses highly demanding media data such as tactile, proprioceptive (muscle tension), and kinesthetic (joint angle and velocity) information [58]. These have different sampling rates which are increasing with advances in technology requiring tradeoff between quality and frequency as well as new compression methods. Internet and LANs produces random transmission delays due to the lack of quality of service which causes real-time processes to go unstable when the delay exceeds some limit. The delays cause proportional degradation in operator performance and jitter disturbs the velocity due to time varying intervals. Predictive models are proposed so that visual feedback is produced at master site and a slave model is used periodically to update the actuator parameters.

An Internet based Distributed Component Object Model(DCOM) design for telerobotics is proposed by [60] to integrate web technologies and telerobotics together with environmental constraints. This provides autonomous mechanism to reduce human attendance, robustness control, and flexibility to ease the task of upgrading the system. A model-based vision system with GPS sensing is used. OpenGL simulator is used for testing. A DCOM/ActiveX supervisory control server is developed using DCOM for low level feature extraction components like segmentation, pre-filtering, measurement, verification, and feature matching with predicted parameters.

In this project report we propose a reliable, real-time, telerobotic framework interfacing the master and slave stations using advanced object-oriented distributed components technology. For this we designed various telerobotic components, interaction methods, and secure communication support while isolating the components from network protocols. Data and object remoting support provides the client station with a real-time update of sensor data and process statuses without the need to register the slave arm components on the server. Telerobotic stereo vision greatly enhance the operator's efficiency but imposes severe requirements in terms of bandwidth to transfer real-time stream of video data in a client-server environment. Using advanced software and hardware technologies we present a highly optimized client-server framework for grabbing and relaying of a stereo video stream. Performance of force feedback and stereo vision multi-streaming delays, bandwidth, and teleoperation with kinesthetic force feedback are presented.

5 Research methodology

This section presents the specification of the project tasks and the way each task was addressed during the course of the project. Each task is followed by a short presentation and discussion of the methodology used in addressing it. Generally short presentations refer to the detailed analysis in the subsequent sections of this report.

5.1 Task M-1

Original Task M-1: Literature review of the architectures of master-slave robot arms in the research and commercial domains, flexibility, ability to communicate sensorial information, and man-machine interfacing. Selection of a commercial arm as a master arm having dc-motorization, reversibility of transmissions, and flexibility. At the end of this task the needed equipment for the master arm and the force sensor including PCs and electronic accessories (interfacing cards, components, etc.)will be ordered. The completion of this task will be measured by the selection of the master arm and associated electronic interfaces together with full justifications. The equipment will be available three months after making the orders. The duration of this task is two months.

5.1.1 Status of M-1 for the first year

Following the literature surveys two options for the selection of the master arm were identified: (1) adapting a light robot arm with back-drivable capability for use as master arm, or (2) adopt a haptic device with 6 dof for position and force sensing. At the end of November 2001, the general specifications for the above options were established and the KFUPM Purchasing Department was requested to forward the purchase requisition to a set that consists of a dozen of potential international companies. Two months later only a small number of requested items with inappropriate bids were received. Thus, the process had to be repeated by targeting selected providers with a detailed description of the needed items.

In the following are presented decisions based on analysis of proposed equipment and the literature surveys. Adapting a sophisticated wire-controlled arm for use as master arm was found to be too expensive (more than USD 200 K) to implement due to the high cost of wire-controlled arms like the WAM, Sarcos master arm, etc. The second solution is to adopt a light master arm with force feedback by using a commercial haptic device such as: (1) Phantom Premium 1.5 and 3.0 (Sensable Technologies), (3) the Freedom 6S Device (MPB Technologies, Canada), and (4) The Delta Haptic Device (Forcedimension Co, Switzerland) were excluded because of very high cost. Haptic devices like (1) the Palmtop display (ATR Communication Systems, Japan), and (2) the Haptic Master (Iwata Lab, Japan) were also excluded because they cannot be exported outside of their original countries. The only devices which meet our requirements (measuring operator motion and providing force feedbacK) including the constraints on cost are (1) Phantom Premium 1.0 (Sensable Technologies) with 6 dof position sensing and 3 dof force sensing, and (2) an older version called Phantom Desktop device having similar features. The bid is USD 22K

or SR 82,5K. Both have a PCI electronic interface card (PC) and software drivers for their interfacing. The detailed specifications of these devices are given in the literature review. Several companies that provide the above systems were contacted. A large number of these providers did not respond. Three companies were identified and finalized the vendor specifications and associated quotations.

The process of acquiring the equipment is very slow and requires continuous intervention of the project team. As per KACST regulation three companies were contacted in February 2002 and a set of three quotations were obtained for the following haptic device:

1. Item A: Phantom 1.0A haptic master arm including a boxed Phantom arm, amplifier box (110v/220v), a PCI card, cables and drivers.

2. Item B: Phantom encoder stylus (1.0A)

The quotations received were form the following companies:

1. SensAble Technologies, Inc., USA. This company quoted only Item A for a total of USD 21.667 (SR 81.251) after applying a discount of 10% for educational institutions.

2. Sim Team, France. This company quoted only Item A for EUR 25.460 and Item B for 3.800. The total for both items is 29.260. They will apply a discount of 8% which makes their total EUR 26.919 or USD 23,958 (SR 98.842).

3. Engineering Systems Technologies (EST), Germany. This company quoted only Item A for EUR 19.718 and Item B for 2.941 after applying a discount of 20% for educational institutions. The total for both items is 22.659. Since 1 EUR is about USD 0.89, then the total for both items is USD 20.166 (75.624).

The first company (SensAble Technologies) was excluded because of (1) high bid for Item A, and providing no quote for Item B. The second company (Sim Team) was also excluded because of its high bid for Items A and B.

It is recommended to purchass *Item A (Phantom 1.0A, boxed Phantom arm, amplifier box (110v/220v), a PCI card, cables and drivers) and Item B (Phantom encoder stylus (1.0A))* from the third company (Engineering Systems Technologies (EST)) because:

1. its quoted equipment matches our specifications (see above), and

2. has the lowest bid for Item A and Item B.

The above quotations and analysis (together with other equipment quotations) were forwarded to KACST on March 4, 2002. The PI was calling KACST every two weeks to follow up on the acquisition process. The PI was asked by KACST to carry out the following requests: (1) provide quotations in Euro not in USD, and (2) make sure the companies accept KACST method of payment (Letter of Credit LC). In each case the PI contacted the companies and did his best to meet KACST requirements. As by May 2002, KACST obtained from the third company (Engineering Systems

Technologies (EST)) the needed approval for the LC and shipment fees. As by September 2002, KACST informed the PI (by Phone) that an LC was opened for the above company and that the equipment (Phantom 1.0A master arm) will be available within one month.

5.1.2 Status of M-1 for the second year

An unsurmountable number of difficulties were encountered in purchasing the Phantom Master Arm.

For the above mentioned reasons we were thinking since the first year of the project about locally designing and manufacturing a master arm for telerobotics. In the first year we designed a prototype 1 DOF master arm based on multiple closed-loop wire transmission and impedance-zeroing controller. We successfully experimented the above arm. The prototype 1 DOF master arm proved the ability to design and manufacture a high-quality master arm with the following features:

1. A flexible wire-based transmission system,
2. A maintainable mechanism based on multiple closed-loop wire transmission,
3. A lightweight structure made of light Aluminum alloy,
4. Force display and impedance reduction controller,
5. A standard PC-based commodity controller.

We feel confident that we can design and manufacture a full 6 DOF master arm with the above features by using the facilities available at the Mechanical Engineering Workshop of the Mechanical Engineering Department at KFUPM. This approach has a number of advantages such as the development of local expertise in the design of a dexterous master arm to support a variety of applications in the Kingdom.

To complete the project we propose:

- extending the 2-year project AR-20-80 by a duration of eight months, and
- reallocating the budget to accommodate the design and manufacturing of the master arm.

The design and manufacturing of the master arm will not require any increase in the project budget. The proposal for the design and manufacturing of the master arm has been submitted for approval by KACST on November 23, 2002. The Arabic translation was sent on January 26, 2003.

The preliminary design of the needed Master Arm has been completed. The team members are awaiting KACST approval and budget reallocation to launch the detailed design and manufacturing at the University.

5.2 Task M-2

Original Task M-2: Adaptation of a light master robot arm having six degrees of freedom with motorization and flexible gear system that is capable of transmitting reflected force feedback to the operator hand. In this task we will interface the master arm to the master workstation computer. The completion of this task will be measured by the accomplishment of the operability method for: (1) the master arm to computer interfacing, (2) the man-machine interfacing, and (3) the testing of the master arm interface, control, and operability. When the whole system becomes operational, this task will be re-visited again for fine tuning of its various components and interfacing. The duration of this task is eight months.

This task is 8 months, of which 5 are in the first year of the project and the other 3 are in the second year.

5.2.1 Status of M-2 for the first year

The main task is the interfacing of the Master arm to the Slave Robot. The completion is measured by: (1) the master and slave arms are interfaced to a LAN, (2) testing of interchanging of data and synchronization using communication commands, and (3) proper operations with reliable connection. This task has interfacing issues and equipment related issues. The interfacing issues have been accomplished. However, the equipment related issues cannot complete before we complete the design and manufacturing of the master arm.

We implemented the interface between a PC used a server station and the PUMA 560 slave robot by connecting the PUMA serial link to a PC and developing a driver program written in Visual Basic (VB). Thus the server PC allows us to issue commands to activate PUMA motion control and well carrying out synchronization operations because the PUMA system operates in the "move-and-wait" mode. For this the server PC monitor a specific control following each motion before being able to issue the next motion command. The current server PC interface enables us to issue motion and status commands allowing us to gain full control of the PUMA from the server PC. In turn, the server PC is interfaced to a 100 Mbps LAN by using TCP/IP socket programming which is controlled from Visual Basic programming environment. This allows to issue motion and status commands allowing from a remote PC (client) which provide a mean of gaining full control of the PUMA from a remote client PC (connected to the LAN) through the server PC and the above TCP/IP socket programming interface.

In this case, the server system consists of (1) the slave robot that is interfaced to a server PC, and (2) the master arm that is interfaced to a client PC. To synchronize the above system and to provide a reliable communication among the (1) slave arm, (2) the server PC, and (3) the client PC we have developed [70] a reliable client-server interface for telerobotics. The client-server system was also written in VB and augmented with reliability communication functions based on stop-and-wait protocol to monitor the status of the three major components: (1) the slave PUMA arm, (2) the server PC, and (3) the client PC.

5.2.2 Status of M-2 for the second year

The Visual Basic (VB) software implementation of the client-server telerobotic system proved to be ineffective software solution to implement real-time data streaming between the slave robot (server) and a master station (client). We specifically had to handle the reliability issues of the communication protocol at the application level. In addition we need an end-to-end software solution that allows good control of CPU processing time, creation of effective copying of data from the server devices (like robot, cameras, and force sonsors), creation of reliable communication sockets, streaming of data transfer in a synchronous way (master-slave control, video, and force), and providing a simple software solution for mirroring of status information at the client site. For the above we decided to use a radically different approach that employs one of the most advanced software solutions for effective control multistreaming of real-time processes in a client-server environment. In the following we present the new client-server implementation.

Using the new distributed components and Multithreaded implementation of the client server system we also re-implemented the direct/inverse geometric transforms (kinematic) of the PUMA slave arm. We tested the correctness of these motion functions in the local and networked modes. The user interface windows showing the status of the slave robot was mirrored to the client machine using the advanced .NET software technology which takes care of the communication and refreshing details. Windows 2000 provides powerful tools to control the timing of real-time processes as well as CPU sharing by setting up process priority of corresponding multithreaded processes (see Section 6.4 and 6.5 for more details).

5.3 Task M-3

Original Task M-3: Design of force sensor capable of providing electrical measurements of forces and torques that are exerted on the manipulated objects. The completion of this task will be measured by the accomplishment of: (1) design and implementation of the force sensor, (2) interfacing of the sensor, and (3) testing the force sensor responses. When the whole system becomes operational, this task will be visited again for fine tuning of its various components and interfacing. The duration of this task is six months.

5.3.1 Status of M-3 for the first year

Task M-3 deals with the design of a force sensor. This task started on November 18, 2001 and will end on April 18, 2002. The literature surveys revealed that there are mainly two commercial force/torque (F/T) sensors with a rigid structure: (1) the ATI F/T transducers (ATI Ind. Aut. see www.ati.com), and (2) the JR3 F/T sensor (JR3 see www.jr3.com). Other devices are based on passive compliance with no sensing like the Remote Compliance Center device (ATI Ind. Aut.). The internal structure of sensors (1) and (2) is made of two parallel plates that are interconnected by using links on which strain gages are installed for force measurement. We were designing two similar force sensors: (1) a rigid structure, and (2) a slightly compliant structure. Both solutions will be evaluated and the one that is capable of effectively sensing fine forces and torques will be adopted. Since no master arm was not yet

available, a mechanical system was designed and built to study the implementation of force-feedback strategies. The system is a wire-based 1 dof master arm with force feedback comprising a 1 dof link with a gear ratio of 10. The force control strategy of the system was also under study. An old motor was used to test the functionality of the system. The required DC motors to support the above experiments with force sensing and force control will be ordered shortly.

We designed and manufactured the rigid and compliant force sensors. Both solutions have been evaluated and tested in the presence of external forces and moments. The signals generated from each of the above sensors consists of six voltages, i.e. 3 forces and 3 moments. Therefore one single I/O card will be used for reading of the sensor signals from the slave PC.

The rigid sensor was tested in the Lab by using some Lab instrumentation electronics. Please see Report 2 for the detailed presentation and signal analysis. The experiments consisted of applying specific external forces and moments and plotting the sensor responses. Analysis of the response proved to be consistent with the applied force or moment. This is useful for calibrating the sensor. Testing the sensor results is as follows:

1. A linear variation of the strain (μ/m) with the applied load, F.

2. All tested strain gages yield similar readings, with a small increase in the difference between the strain indications at higher loads.

3. The current sensor structure may not be adequate for sensing torques around the Z-axis.

These preliminary experimental results will be analyzed to improve the design of the sensor. The finite element analysis was helpful to optimize the sensor design and to select the location and proper size of the strain gages in the final sensor design. The integration of some dedicated instrumentation electronics for this sensor allowed testing it as a wrist sensor.

The compliant force sensor was tested as a wrist sensor on the PUMA 560 robot arm by implementing its own simple electronics and interfacing the sensor to a PC card providing the needed A/D conversion and data acquisition functions. Please see Section 6.3 for the detailed presentation and signal analysis. The testing experiments are based on selective application of force and torque and measurements of the corresponding output signal from the compliant force sensor. Specifically, the experiments consist of (1) applying specific external forces and moments, (2) reading by the PC of the sensor signals, and (3) plotting the six sensor responses. We studied each setting of individual force and torque with both positive and negative magnitude. Analysis of the output signal proved to be consistent with the applied force or moment. The conclusion from the testing are the following:

1. The attachment of the rubber blocks to the disks is not reliable and a simple mechanical attachment was used to avoid this problem.

2. The use of rubber blocks to provide some compliance is subject to the traditional hysteresis problem. As a result the sensor signals do not return to their previous values when the external force is removed.

3. Due to the use of optical devices for the detection of small motion the setting of the zero reference for all the sensors is difficult.

4. There is a clear linear variation of sensor signal with the applied load.

Further improvements and analysis were undertaken arriving at final design for both force sensors. Specifically, some improvements and revisions will be needed when the above sensors are used to sense contact forces occurring during telerobotic tasks and transmitted as force feedback to the operator site.

5.3.2 Status of M-3 for the second year

The compliant sensor has been revised to improve its stability by implementing fixtures for the rubber blocks. This ensures that external forces exerted will be directed to the sensors. In addition the fixtures will provide a better return to previous state after removal of external forces. The electrical interfacing is complete. We also completed sensor interfacing with the computer by acquiring a Window 2000 Eagle card with 32 analog inputs and 4 analog outputs. The next step is to integrate the sensor in the main loop of the telerobotic system and study the force information generated to the client side using the new client-server system.

As mentioned earlier in Report 2 the rigid sensor was unable to provide clear feeling of applied torque (Z axis). To overcome this problem a new force sensor was designed and manufactured. The strain gages are mounted on a tetra-hedral structure that connects two circular plates. This structure is a parallel arrangement of six active links similar to those used to control giant mirror in space telescope. The sensor was evaluated using specially designed test stands. The general trend in the results is encouraging. However, problems were encountered in strain gage sensitivity. It is found that the placement of strain gages is defective because it does not produce similar sensitivity and resulted in an unreliable sensing. The idea of developing force sensors based on confining six strain gages in a small area was abandoned.

The compliant force sensor was successfully used in master-slave interconnection through a LAN. The experiment reports a telerobotic system in which the operator moves down a 1 DOF master arm along the vertical direction which causes similar motion in the slave arm tool. A real-time force data stream is transmitted from the server (slave robot) to the client (master arm). The study dealt with the force interaction occurring during contact between the slave arm tool and (1) a rigid body (case (a)), (2) a spring (case (b)), and (3) a human muscle tissue (case (c)). Following the contact, the operator was asked to maintain a constant force on the target. For a total duration of about 20 s the task of the operator is to bring the slave arm tool into contact with the target and to maintain on it a constant force of 0.75N. Each contact operation has 5 phases which are (1) contact-free, (2) pre-contact, (3) contact, (4) pre-release, and (5) release. For each given instance the operator receives no force feedback as far as the tool is still in free space. The pre-contact phase starts when the tool hits the target in the remote environment. It is noted that in both pre-contact and pre-release phases the teleoperation system is subject to vibrations which are displayed to the operator through the master arm after scaling it by some factor. The vibration frequency depends on the combination

of two factors which are the stiffness of the target and the value of force feedback gain (FFG). Stiff targets produce prompt bouncing contact forces and therefore produce higher vibration frequency. The vibrations for the rigid object are greater and faster than those of the spring or the tissue. The experiment is fully described in the subsection entitled "Performance evaluation" of Section 6.4.

5.4 Task V-1

Original Task V-1: Literature review of available stereo vision, light cameras, camera arrangements, stereo/3D visualization technology and techniques, and associated PCs, CRTs, and cabling. Commercial Stereo and 3D visualization tool kits will be investigated. Ordering of the selected Stereo or 3D visualization tool kit including its hardware (interfacing boards, image compression cards, stereo-vision display kit) and software components. The equipment will be available three months after making the orders. The duration of this task is two months.

5.4.1 Status of V-1 for the first year

Literature review on 3D video generation, 3D visualization, and associated hardware for PCs and CRTs is completed. We completed the literature surveys of Video and 3D visualization approaches as presented in many recent (95-2002) research papers. We also classified the visualization methods used with respect to (1) the techniques used, local or remote (Coaxial cable, LAN network, Internet, etc), and (2) the way it is used with respect to the robotic applications (See Section 4.4).

The beam splitter approach, NuView 3D video adapter approach, and two-parallel/tilted cameras approach for 3D video generation were considered in detail. Because beam splitters are known to have some artifacts, they have been ruled out. Two parallel/tilted cameras approach seems to be the best method because it allows the control of many parameters like eye separation distance, tilt angle, etc.

For 3D visualization, both shuttering glasses to be used with a PC and Monitor, and head mounted displays have been reviewed. Most of the commercially available monitors can support quite high resolutions and refresh frequencies, and when used with shuttering glasses can produce high-resolution 3D images/video. However, for user concentration, head mounted displays are superior compared to shuttering glasses, and they give a view equivalent to a very large size monitor. On the other hand, their disadvantages are: (i) they are expensive, and (ii) Head Mounted Displays which are available in the market support either VGA and/or SVGA resolutions only. Therefore, both options will be tried to see which one works better. Based on the above requirements and the allocated budget we need the following equipment: (1) four digital cameras with their tripods and accessories, (2) two high-resolution Head Mounted Displays and accessories, 3) ten high-quality Eye Shuttering Glasses with accessories, and (4) motor components to control the camera orientation.

The PI was asked by KACST to carry out the following requests: (1) providing quotations in Euro not in USD, and (2) make sure the companies accept KACST method of payment (Letter of Credit). In each case the PI contacted the companies and did his best to meet KACST requirements. As by September 2002, KACST

informed the PI (Phone) that the above equipment arrived at Riyadh Airport and will be delivered at KFUPM.

5.4.2 Status of V-1 for the second year

Reference to the purchasing order, dated 4/5/2002, forwarded to KACST for the acquisition of the following equipment:

1. Two (2) units of SVGA resolution Head-Mounted Display (Cy-visor DH-4400VP 3D).

2. Ten (10) units of LCD Shutter Glasses (H 3D PC glasses, wireless version).

The formal PR (May 4, 2002) requested for CIF King Fahd International Airport, City of Dammam, and delivery at the address at KFUPM. Due to a mistake, the company sent the equipment to KACST during summer. In the beginning of September 2002 we have been contacted by the purchasing department of KACST to inform us of the arrival of the above mentioned equipment at King Khalid Airport in Riyadh. On September 20, 2002, the purchasing department of KACST forwarded the received equipment to KFUPM.

After inspection of the main box we found that one Head-Mounted Display (cost USD 2000) is missing, its box contains only the reference manual. We contacted by phone the purchasing department of KACST, to inform them about this problem, and request blocking of the payment. Later we learned that the payment cannot be blocked at that stage. We also sent a memo to the company requesting replacement of the missing HMD.

On September 25, 2002 we filled up the Testing Report for the equipment. The result is as follows:

1. One Head-Mounted Display is properly working,

2. Ten LCD Shutter Glasses (cost USD 1100) are not working.

We also sent a second memo to the company asking for their advice regarding the equipment that is not working.

KACST administration was immediately informed of the problem and requesting them to pursue the matter with the company and with the insurance firm. No formal reply has been sent to us concerning the above problem.

After long discussion with the company (two months) the PI coordinated the return of the 10 LCD Shutter Glasses back to the company. In the beginning of February 2003, the PI was informed by the company that the above LCD belongs to a faulty batch and they assume full responsibility for them. After consultation with the co-investigator Dr. Onur Toker we decided to request an Head Mounted Displays HMD in replacement of the faulty LCDs. The company (Cybermind, Netherland) agreed to the request and the above HMD was received. On testing we found than the newly acquired HMD has a serious defect and we concluded that the company is dealing with us in an unethical way.

5.5 Task V-2

Original Task V-2: Install the visualization equipment, implement, and ensure operability of a local direct connection (through a computer) between the mobile camera system and the visualization display kit. Implement the networked connectivity between the vision capturing system at the slave site and the visualization system at the master site. Testing of the remote connectivity. When the whole system becomes operational, this task will be visited again for fine tuning of its various components and interfacing. The duration of this task is five months.

5.5.1 Status of V-2 for the first year

Because of the difficulty encountered in collecting quotations from local and international vendors, we couldn't finalize the purchase of all visualization equipment and complete all of the experiments. However, we have used an existing IEEE-1394 card which belongs to a faculty member, two digital video cameras of different optical characteristics purchased for other projects, and again a faculty member's PC which has a display adapter that supports analog video in/out, and we were able to generate 3D video signal. The format was the so-called ABOVE/BELOW format, and is supported by many 3D shuttering glasses as SYNC DOUBLING mode. As a side result, we have also observed that a fast PC with an IEEE-1394 card is able to "synchronize" two digital video cameras. Furthermore, some initial studies have been made towards interlacing two video streams by using Windows based programs, and generating the INTERLACED format 3D video. The INTERLACED format is also commonly used, and is supported by many shuttering glasses as well as Head Mounted Displays. However it is fragile and right/left information can easily be corrupted if resized or compressed improperly.

Existing analog video cards, which are again personal belongings of certain faculty members, are used to send 3D video content, output from the above mentioned PC, to computers on the university network by using the well known NetMeeting program, and the experiment was successful. Basically, we were able to do video synchronization and mixing by using commodity equipment, generate 3D-video signal, and send it to a remote PC successfully.

Finally, a faculty member's personal shuttering glasses set is used, to verify that the SYNC DOUBLING mode is a complete hardware solution. This solution does not depend on the display adapter's chipset, and does not need any special drivers. The controller sits between the PCs display adapter and monitor, modifies the signal sent from the display adapter to the monitor, and turns on/off each eye in a cyclic manner. We have gained some experience in 3D video generation/visualization and transfer, and hope that we can complete the remaining experiments in much shorter time once the equipment is received. Please Sections 4.4 and 6.2 for the details.

The approach consists of setting up the 3D visualization system under: (1) a local mode, and (2) networked mode. In the local mode both the scene cameras and the 3D visualization system are attached to one single computer. In the networked mode the scene cameras are attached to a server computer and the 3D visualization system is attached to a client computer, where the client and server computer are interfaced through a LAN. This allows evaluation of the quality of 3D visualization

as well as assessing a number of important features like the rate of refreshment of the frames, effect of parallel and tilted cameras, etc. Using the project provided cameras and a borrowed eye shuttering system we describe the experiments on 3D visualization for both local and networked modes in Section 6.2.

The ordered Digital Cameras (from local market) were delivered in April 2002. This allowed to address the issue of the operability of the stereo system in the local direct connection in which the digital camera system and the visualization display kit are set in the local interconnection, i.e. one single PC without network. Please see Section 6.2 for the details. We studied the performance of stereo views in a confined scene without network. We studied the effect on the quality of stereo views based on: (1) setting of the cameras, (2) values needed for the horizontal shift between left and right images, (3) effect of tilted cameras, (4) effect of the distance from cameras to scene, (5) effect of camera focus on quality of views and value of horizontal shifts, and (6) effect of scene lighting. We also provided general comments on the quality of depth perception. We commented on the extent to which the user can estimate the depth and make correct evaluation of common sense metrics in the scene (estimating the lengths). We also commented on the extent to which the user can bring a device (robot gripper) in the vicinity of another object. Please see Section 6.2 for the details of the client server system which relays stereo-images over the LAN.

We implemented the networked connectivity between the 2-cameras system (server system, server PC) and the visualization station (client PC) that are interconnected by through a 100 Mbps LAN. Please see Report 2 for the details. for this we described (1) the method used to combine the two camera streams at the server node, (2) the transfer method, and (3) the reproduction of the stereo views at the client station. We carried out the testing of the remote stereo reliability, data rate, number of frames/s, and the available resolution at both server and client CRTs. We studied the effect on the quality of networked stereo views of (1) stereo stability of the images (flickering), (2) fine tuning of the system for quality stereo perception, and (3) quality of depth perception with regard to any network or delay aspect. We carried out experiments showing to what extent the user can (1) estimate the depth and make correct evaluation of the metrics, and (2) bring a device (robot gripper) in the vicinity of another object.

5.5.2 Status of V-2 for the second year

The Visual Basic (VB) software implementation of the client-server telerobotic system proved to be ineffective solution for real-time data streaming from the slave robot (server) to a master station (client). Specifically we had to handle the reliability issues of the communication protocol at the application level. In addition we need an end-to-end software solution that allows good control of CPU processing time, creation of effective copying of data from the server devices (like robot, cameras, and force sonsrs), creation of reliable communication sockets, streaming of data transfer in a synchronous way (master-slave control, video, and force), and providing a simple software solution for mirroring of status information at the client site. For the above we decided to use a radically different approach that employs one of the most advanced software solution for effective control multi-streaming of real-time processes in a client-server environment. In the following we present a

summary of the new client-server implementation which is presented in details in Section 6.5.

Microsoft DirectX provides distributed components interface for various graphic related functions. DirectShow is one of these services. DirectShow, further, provides efficient interfaces for capturing and playback of video data. In the scheme we uses a component of DirectShow named SampleGrabber to capture video frames coming through a stream from a stereo camera setup. Media by a PCI card that hosts FireWire input ports for devices using FireWire standard. After that we hook capture filters provided by DirectShow to get hold of the video stream from the cameras. Once we have video stream, the SampleGrabber is attached to capture the video samples from the stream.

The graphical component of the Windows graphical environment is the graphics device interface (GDI). It communicates between the application and the device driver, which performs the hardware-specific functions that generate output. After receiving the video data from windows sockets, we use GDI functions to show the picture on the monitor screen.

We also presented the performance evaluation of the real-time stereo vision system in the subsection entitled "Performance evaluation" of Section 6.5 of client-server multi-streaming of video data, commands, and the standard implementation can achieve a real-time transfer of about 11 pictures per second, where each picture consists of two video images with a total of 5 Mb payload. We re-engineered the multithreaded environment at the server software system so that to create independent threads for video grabbing and streaming which lead to pipelining of the processes of grabbing and streaming. With this new approach we could achieve streaming of about 19 pictures per second which means that the bandwidth of 100 Mbps LAN is fully utilized. We present the statistical analysis and of the communication delays and delay jitter from grabbing of data at a server site to display of the stereo views at a the client site.

Also we started the investigation on the augmented reality as part of the strategy to "Reduce Network Delays in Telerobotics". Please see Section 6.6 for the details of the augmented reality system that allows the operator to carry out task planning by creating a virtual ball (representing the slave arm gripper) and moving it to the vicinity of a target object. This consists of using a simple pointer in the stereo image (client) that is set in a relative position to current display of the robot gripper. The position of the pointer is controlled by the client (for example through a mouse). The setting of the point relative to the gripper needs to take into consideration the geometric relationship, in the slave station, linking the robot gripper frame to its reference camera frame (for both cameras). This enables sending in one packet a trajectory to slave arm to follow up in addition to final setting of the gripper which greatly reduce network delays in telerobotics.

In summary we implemented in the Robotics Lab a real-time stereo image transfer from a slave PC to a client PC by using the digital cameras. The system is reported in Section 6.2 in which we present the implementation for a *Real-time, multi-streaming, for stereo vision system.* We also evaluate the quality of stereo effects with respect to the operations needed in telerobotics. As a strategy to reduce network delays in telerobotics we modified the above stereo vision system to provide augmented reality service at the client side which is presented in Section 6.6.

5.6 Task R-1

Original Task R-1: Literature review of master-slave bilateral systems, control strategies for telerobotics, effect of delays, and methods for improving performance in presence of network and mechanical delays. Ordering of interfacing equipment needed for the slave workstation. The duration of this task is two months.

5.6.1 Status of R-2 for the first year

Task R-1 started on September 18, 2001 and ended on November 18, 2001. The literature surveys on the mapping methods between the master and slave systems, control strategies for telerobotics, effect of delays, and methods for improving performance in presence of network and mechanical delays was completed. These were presented in the literature survey. These approaches have affected the methodology as shown in Task R-2 and R-3.

In the interfacing part of this task we identified the needed equipment as follows: (1) two full option Pentium 4 Personal computers, (2) Fire wire interfacing cards for the interfacing the video equipment, (3) Network cards and other accessories. Since this equipment is available on the local market the ordering and delivery were complete by March 2002. This task completed.

5.7 Task R-2

Original Task R-2: Design and implementation of the master-slave controller and the interfacing of the slave robot to its local computer. The accomplishment of this task can be measured by the following three factors:

1. *The direct connection between master and slave arms is operable.*
2. *The reflected force feedback is properly (quality and quantity) sent from the slave arm to the master arm.*
3. *The direct connection between the master-slave system is properly operating, signals are correctly received by the master and slave, and activation of both master and slave arms.*

The duration of this task is five months.

5.7.1 Status of R-2 for the first year

The team successfully designed and implemented the interface between a PC (server station) and the PUMA 560 slave robot by connecting the PUMA serial link to a PC and developing a driver program written in Visual Basic (VB). To reduce network delays and overhead, the PUMA robot was run under a fine trajectory control so that only coarse master arm positions need to be send while local slave control is continuously active. The interface allows: (1) downloading programs to the PUMA to provide local trajectory control, and (2) uploading state information to control the PUMA from the server station. It was found [70] that using the above approach, the dominant delays in the LAN environment are only due to robot motion. Software

for the Client-Server interface between the robot server and a remote client PC was successfully designed and implemented. The client-server system was also written in VB and augmented with reliability communication functions based on stop-and-wait protocol to monitor the status of its three components: (1) slave PUMA arm, (2) server PC, and (3) client PC.

In the following we describe the work on the Robot Server and the client in order to set up the direct connection between master and slave arms. Please see Report 2 which presents the kinematic model of the PUMA robot arm which is used here in mapping the motion of the master arm to slave arm.

To provide the needed connectivity and control the Server PC carries out the following tasks:

1. The server reads the PUMA joint angles (θ),
2. It computes the direct kinematics model $X = G(\theta)$ that takes as input the joint vector θ and outputs the position and orientation of the robot hand denoted by X,
3. A request from a local or remote client is a request to the slave robot to perform an increment ΔX over its current position X.
4. On reception of increment ΔX from the client the server finds the new joint vector θ_{new} by computing the desired hand position of the PUMA as $X_{new} = X + \Delta X$,
5. The server now evaluates the inverse kinematic model of the slave robot as $\theta_{new} = G^{-1}(X_{new})$ that takes the new hand position vector and evaluates the new joint angles θ_{new},
6. The server sends θ_{new} to the PUMA arm,
7. The PUMA system is programmed to generate a fine-grain trajectory moving the arm from its previous position θ to the new one θ_{new} at a specified speed. Once the motion is completed the PUMA returns a ready signal to the server and the cycle is repeated.

The above mechanism was fully implemented, debugged, and tested. In other words the direct connection of the slave arms is fully operable. Details on the mapping control can be found in Sections 6.1.

To test the direct connection between the client (master arm) and the slave arms we were in need of a master arm. Instead of wasting time and wait for the delivery of a commercial master arm (the haptic device) we decided to experience the following ways:

1. We designed a passive 6 dof simple articulated structure (SAS) having (1) a handle for the operator, and (2) a fixed base. The SAS is equipped with potentiometers which allow its interfacing to the A/D card of a PC to read its joint vector and to compute the operator's hand position and orientation. The SAS was very useful in the process of debugging the direct connection between master and slave arms. The kinematic model of the SAS is presented

in Report 2. which also presents the research on the suited operator interface to provide an effective mapping to slave arm. We proposed a master arm handle that provides a simple mapping of the operator hand frame to control the slave arm.

2. We designed a 3D Vision-based Man-machine interface that consists of two digital cameras which continuously monitor the operator hand holding a frame with colored balls. Please see Section 6.7 for the details of the approach. This allows finding, in real-time, the operator hand frame position and orientation. If one computes the incremental position of the operator hand then he can use it to control the previously defined slave arm. The direct connection between 3D Vision-based Man-machine interface and and slave arms was implemented, debugged, and is currently operable. This approach can be useful to (1) overcome the high cost associated with the use of mechanical master arms, and (2) provide lightweight man-machine interface capable of serving as a 3D pointer including position and orientation.

3. We also designed a 1 dof simple master arm (SMA) with force feedback. The arm consists of a DC motor connected to 1 dof arm through a wire-based transmission system having a gear ratio of 10 with a position potentiometer. The operator can move the arm and a PC can read the position from the potentiometer. The force feedback is implemented as follows. The PC uses a servo function that normally operates as "power steering" system to help the operator moves the arm, however, a received force feedback from the slave arm would be "displayed" to the operator through the servo motor. The result is that the servo motor (1) helps the operator to zero the impedance of the master arm, and (2) provides a way to display reflected force to the operator. We fully implemented the above SMA with its electronics, computer interfacing, and servo motor function, and will shortly use it to study how reflected force feedback can be displayed on the top of the normal "power steering" function.

4. We will also interface the manufactured master arm to the client PC and use it to determine the operator motion as well as to display force feedback. This will start after we complete the design and manufacturing of the master arm.

The connection needed for the reflected force feedback was implemented as a separate network connection that sends packets carrying information about the external force sensed at the wrist of the PUMA arm. This function of sending the reflected force feedback is properly operating between the server PC and the client PC. In summary the direct connection between the server site and client site is properly operating and the signals are correctly received by both sites.

5.7.2 Status of R-2 for the second year

Using the new distributed components and Multithreaded implementation of the client server system we also re-implemented the direct/inverse geometric transforms (kinematic) of PUMA slave arm which is presented in Section 6.1.4.

The motion functions were tested in the local and networked modes. The user interface windows showing the status of the slave robot was mirrored to the client

machine using the advanced .NET software technology which takes care of the communication and refreshing details. Windows 2000 provides powerful tools to control the timing of real-time processes as well as CPU sharing by setting up process priority of corresponding multithreeded processes (See Section 6.4 for more details).

5.8 Task R-3

Original Task R-3: Operability of master-slave system under the direct interconnection. The accomplishment of this task can be measured by the following factors:

1. *The motion of the slave arm is a replica of the motion of the master arm even if the master and slave arms have different architectures,*

2. *The controller must account for the mechanical and network delays by using some predictive scheme.*

3. *The master arm transmits the reflected effort to the master arm handle. An operator holding the master arm handle easily feels the force feedback in the right direction and time.*

The duration of this task is three months.

5.8.1 Status of R-3 for the first year

Task R-2 started as planned. The implementation of the physical interface between the server station and the slave station is described in task M-2. We successfully designed and implemented the LAN network interface between a PC (server station) and the PUMA 560 slave robot by developing: (1) a client program (connection to master arm and to LAN), and (2) a server program (connection master slave arm and to LAN). Both Client and Server programs were first written in Visual Basic (VB) and later in C++ as described in Section 6.4. To reduce network delays and overhead, the PUMA robot was run under a fine trajectory control so that only coarse master arm positions need to be sent while local slave control is continuously active. The interface allows: (1) downloading programs to the PUMA to provide local trajectory control, and (2) uploading state information to control the PUMA from the server station. We found [70] that using the above approach, the dominant delays in the LAN environment are only due to robot motion. Software for the Client-Server interface between the robot server and a remote client PC was successfully designed and implemented. The client-server system was also written in VB and augmented with reliability communication functions based on stop-and-wait protocol to monitor the status of its three components: (1) slave PUMA arm, (2) server PC, and (3) client PC.

Testing of the master slave-system using the 6 dof simple articulated structure SAS showed that the approach of kinematic control of PUMA slave arm provides structure-independent (master arm and slave arm) mapping. The reason is that the server sends the PUMA the new joint vector as $\theta_{new} = G^{-1}(X + \Delta X_{new})$ based on a client request to perform increment ΔX_{new} in cartesian space. This provides full uncoupling between the structure of the master arm and that of the slave arm. For example any mechanism (like the 3D vision system) that is capable of delivering the

increments ΔX_{new} in cartesian space can be used to teleoperate with the slave arm. Details on the mapping control can be found in Sections 6.1. Therefore, the motion of the slave arm is a replica of the motion of the master arm even if the master and slave arms have different architectures.

We are accounting for the mechanical and network delays as follows. We reduce the effects of the mechanical delays by controlling the trajectory of the PUMA in the joint space as opposed to controlling the trajectory in the cartesian space. The old LSI-11 processor is responsible in the PUMA system of generating cartesian space trajectories by computing the Inverse Kinematic Model (IKM) for the slave arm in real-time. This computation is much slower than a control in the joint space which requires no involvement at the IKM level in real-time. In this approach the server sends the PUMA the new joint vector as $\theta_{new} = G^{-1}(X + \Delta X_{new})$, thus the IKM is computed by the fast Server-PC not by the slow PUMA LSI-11 processor. In addition we optimized the data transfer time between the server and the PUMA so that only a few parameters are communicated for each motion. For this the PUMA system is programmed to generate a fine-grain trajectory moving the arm from its previous position θ to the new one θ_{new} at a specified speed. This approach provides the best possible optimization to reduce the mechanical delays occurring during the PUMA motion by using the PUMA system.

We also minimized the network delays by operating the client and server in an asynchronous or independent way. For example, instead of waiting for the arrival of increment ΔX_{new} from the client, the server continuously reads θ and computes $X = G(\theta)$. When the increment ΔX_{new} arrives from the client, the server picks up the most recent value of $X = G(\theta)$ and uses it in computing $\theta_{new} = G^{-1}(X + \Delta X_{new})$. Thus the computation time $X = G(\theta)$ is overlapped with the network communication delays. The same process is done at the client which continuously reads the θ vector of the "master arm", computes $X = G(\theta)$, and evaluates the latest increment ΔX_{new} which is sent to the server if a ready signal is available. The important issue to note here is that all computations are done, in the server or the client, in an asynchronous or independent way to overlap their times with the motion or communication times which result from the motion synchronization of the PUMA with the master arm. In summary we minimize the network delays by carrying out all computations in an asynchronous way in both client and server but we keep motion synchronization by using fast messaging system (busy-wait or ready) between the PUMA motion and the client commands. The motion synchronization system and fast messaging are parts of the *Reliable Client-Server Telerobotic System* that is described in Section 6.4.

For the display of the reflected force feedback we are planning on the following tasks:

1. The 1 dof simple master arm (SMA) with force feedback. The SMA will be used to study how the reflected force feedback can be displayed on the top of the normal "power steering" function. Task R-2 gives the details of the methodology.

2. The interface of the master arm to the client PC will allow us to display the force feedback. Please see the procedures presented for Task 5.10.

5.8.2 Status of R-3 for the second year

The new distributed components and Multithreaded implementation of the client server system is complete (see Section 6.4 for more details). We have also completed the implementing of the direct/inverse geometric transforms (kinematic) of PUMA slave arm (see Section 6.1.4 for more details).

We will implement a new "Master-Slave Workspace Scalability Function" to allow controlled scalability of master motion and consistency when changing the geometric operating mode, i.e. mapping of master to slave. We will also implement simple automated functions at the level of the server. This will provide some supervisory control functions that are useful to support fast slave reaction in the case of exceptions without operator involvement.

However, the completion of this task requires operations with the master arm. This means that we need to complete the manufacturing of the master arm as soon as possible.

5.9 Task R4

This task is on the operability of Master-slave system under the networked interconnection (second year). The accomplishment of this task can be measured by the following factors:

1. *The network connection from master arm to slave arm effectively transfer all needed signals in both directions.*

2. *The networked reflected force feedback is continuously sent from the slave arm to the master arm.*

3. *The networked master-slave connectivity is operable.*

4. *Operability of the master-slave system like in Task R-3.*

The duration of this task is six months.

5.9.1 Status of R-4 for the second year

Using the new distributed components and Multithreaded implementation of the client server system we have also re-implemented the direct/inverse geometric transforms (kinematic) of the PUMA slave arm. These motion functions were tested in the local and networked modes.

Windows 2000 provides powerful tools to control the timing of real-time processes as well as CPU sharing by setting up process priority of corresponding multithreaded processes. Sections 6.4 and 6.5 provides the details. In local mode, we can issue any motion command from a user interface and see its effect on the PUMA. We can also loop repeat an incremental motion function on the joint variables or by using cartesian parameters. We repeated the same tests from the client side. All the test were successful under local or remote modes (network). Furthermore, the test produced correct motion reaction and repeatability testing proved that the new client server system is highly reliable compared to the Windows 98 and VB implementation. The performance evaluation of Section 6.4 describes out setting

for streaming of the reflected force feedback from the slave arm to the master arm and its display on the operator holding the master arm handle.

5.10 Design and manufacturing of the master arm (Extension of 8 months)

The preliminary design of the master arm has been completed during the second year of project. In order to design and manufacture the Master Arm KACST allocated the team an extension period of eight months during which the project team targeted the following New Tasks.

1. New Task NT-1: Manufacturing of threaded transmission wheels for the master arm. Some of these wheels must have a shoulder to allow installing a large gear wheel (LGW). Some wheels must be right-hand and other left-hand. Each wheel is 60 mm D and 6 mm width. Each wheel has 4 threads with minimum clearance between threads. Each LGW must be wide enough to allow at least complete three threads. All effort should be made to remove material from the above wheels to minimize the inertia while maintaining reliable structure and assembly. Task 1 was completed on the First of September, 2003.

   Status of New Task NT-1:
   This task was completed successfully. Please refer to Section 6.8 and specifically sub-section 6.8.3 for the details of this task.

2. New Task NT-2: Manufacturing the assembly which incorporates DOF 3, 4, 5, and 6 as per the preliminary master arm design. The exception are (1) bearing on opposite sides of any wheel should be internally supported at their inner rings so that to avoid friction in the case of axial forces at installation, (2) to provide proper alignment the length of the left and right roller wheels for the master arm must be set to 72 mm instead of 65 mm (left roller) and 70 mm (right roller) in the current master arm design. The setting of the hub gear for the slave arm will be decided later after we determine the right hub gear to be used. All effort should be made to remove material from the above structures so that we obtain the least inertia while maintaining reliable structure and assembly. Task 2 was completed by the First of September, 2003

   Status of New Task NT-2:
   This task was successfully completed. Please refer to Section 6.8 and specifically sub-section 6.8.3 for the details of this task.

3. New Task NT-3: Design and drawing of DOF 1 and 2 including threaded gear wheels, structure to support wiring up to motor, motor housing, and overall housing. We analyzed the details of this task and emphasized the need for a sufficient time period (three months) for this task. For this the task schedule was from the First of September to the end of November.

   Status of New Task NT-3:
   This task was successfully completed. Please refer to Section 6.8 and specifically sub-section 6.8.3 for the details of this task.

4. New Task NT-4: Manufacturing of DOF 1 and 2 including gear wheels transmission, intermediate structure to support wiring up to motor, motor shaft wheels, motor housing, and overall base housing. The main idea is that DOF 1 rotation cylinder is traversed by 5 pairs of wires from DOF 2, 3, 4, 5, and 6. These pairs of wires provide a transmission connection between (1) large threaded gear wheels installed at joint of DOF 2, and (2) fixed DC motors installed on a common housing base. This manufacturing task was performed from the beginning of December until January 15, 2004.

   Status of New Task NT-4:
   This task successfully completed. Please refer to Section 6.8 and specifically sub-section 6.8.3 for the details of this task.

5. New Task NT-5: Assembling of the master arm, installing the motors, installing of the transmission wires, testing of the mechanical and electrical connections, and electronic interfacing to the master work station. The duration of this task is 15 days. This assembly task was from the 15 of January to end of January, 2004.

   Status of New Task NT-5:
   This task successfully completed. Please refer to Section 5.11 for this task. In next Section 5.11 the experiments and evaluation for the previously defined tasks O-1 and O-2 are presented.

5.11 Task on testing and evaluation

The testing and evaluation tasks are as follows:

1. Task O-1: The following set of typical tasks will be used for testing of the system:

   (a) *Mixing of liquids which requires fine trajectory and time control, excellent visual interaction, and multiple views. The liquids are originally available in two small containers. The task is to show: (1) the ability of manipulating small objects (small container), (2) maintaining excellent control of position and orientation, and (2) achieving some tasks like mixing in a reasonable time.*

   The "Mixing of liquids" task

   The setting of the experiment is as follows. First the operator is sitting holding the handle of the manufactured master arm. He has a number of keys on master arm handle to control the system. The most important keys are the shift and scale keys. The master arm is interfaced to a PC which regularly samples the master arm motor angles, compute the change in position and orientation of operator hand, and transmit the change to the slave station (server) through the computer network. At the server station, the server program super-impose the received change in master arm motion to that of the slave arm tool frame and forwards the

data to the PUMA system for execution. The detailed mapping depends on the selected mapping mode and activated slave automation function as described in Section 8. In addition the operator sees the remote scene in stereo 3D using a Head Mounted Display (HMD) that receives about 18 frames per second from the remote scene. The above operator setting is valid for all the experiments described in this section. Please see the attached CD video clip on Mixing of liquids.

This task deals with grasping of a small cap that contains colored water using the slave arm gripper, moving it to the neighborhood of an empty cap, and pouring the water in the later cap. This task was successfully accomplished with short training.

In the following we present the observations on overall telerobotic system which result from repetitive execution of the above task:

i. The operator can focus on the remote task without paying attention to his arm, or the mechanical structure of the master arm. The operator cannot anyway see his arm nor the master arm. The designed master arm is likely to be a transparent tool for telerobotics.

ii. Mapping of motion from the operator hand, who is holding the master arm handle, to the slave arm tool is excellently serving its purpose. As a result the operator feels he is directly acting (moving) on the slave tool or the manipulated object. Since this is a natural capability of human arm the operator needs not to see his arm or the master arm during direct teleoperation but only concentrate his view on the remotely manipulated object or tool.

iii. Grasping the cup requires the operator to move down the slave arm by moving his own hand. He also needs to set the slave arm in a pre-apprehension configuration of slave gripper position and orientation with respect to targeted cup. It is noticed that the *Shift Function* is critical to achieve this task. It is one of the most important Computer Aided Teleoperation functions. Moving the gripper to the vicinity of the object is done by successively moving down the slave gripper and shifting the master arm while adjusting its orientation. The operator shift the master arm to his dexterity area without affecting the current position of slave arm which is freezed during the shift operation. With short training the operator can bring the slave gripper in the vicinity of the cup and set up the final position and orientation of gripper. The gripper orientation is progressively set and the operator advances the gripper towards the cup while continuously centering the jaw. The cup is grasped and the operator moves up the slave gripper to travel to target position. In addition the operator may use the motion scalability function (See section 8 for details) when operating in the vicinity of the object where accurate positioning is required. In this case his motion can be scaled down by up to a factor of 15 both in position and orientation. This provides another critical function which is activated by using a potentiometer, located on the master arm handle, that is controlled by the index of the op-

erator. It is carefully implemented at master arm handle so that it can be easily activated by the operator index without distracting his attention.

iv. Using HMD stereo visualization system, the operator excellently perceives the scene depth information. He can estimate the distance between manipulated objects and perceives the geometric positioning of parts with respect to each other during part mating operations. However to get excellent perception we need high quality focusing for both left and right cameras. First the two cameras must have an accurate horizontal disparity of 6 cm with the least possible vertical shift to eliminate any degradation due to tilted cameras. Second the best stereo effects are obtained for a distance from cameras to scene between 100 mm and 200 mm for Sony Handicam Digital Cameras. Third the zoom levels should be exactly the same for convergence of left and right images. Fourth the effect of scene lightning is important because normal florescent light is found to cause flickering, and hence user uncomfort, proper lightning is very important.

v. Remote camera zooming function and the motion scalability function present excellent operating tool for doing tasks in very small scale like a few mms as well as enabling the generation of fast motion of up to 0.5 mps in larger operating areas. The above function enables scaling up and down the whole eye-hand motion coordination system which is certainly one critical issue in telerobotics.

vi. The pouring operations is accomplished as follows. After grasping the filled cup we need to approach the target cup and set up the pre-pouring position and orientation. This requires some training. For example finding the right position and orientation of the held cup with respect the recipient cup before pouring the water. The mapping function from operator hand to slave tool provides a natural way to the operator for pouring the water by simply twisting the operator wrist. Here also the operator uses the shift function to repeatedly rotate the cup until the water is completely poured. The motion mapping from the operator hand to the manipulated cup is excellent and the operator feels he is directly acting on the manipulated cup. Notice that the operator cannot see the motion of his hand nor the implied motion of the master arm. The operator sees the remote target cup through the HMD (network) and acts by moving his hand. The operator is acting as part of a loop extending over the network. The task performance is excellent in teems of ability to maneuver the manipulated cup and achieve the task objective. The task is accomplished very successfully and its execution can be made faster after a short training period (1 hour) on the system.

(b) *Operating mechanical parts in the following two tasks: (1) assembling and disassembling of simple water pump, and (2) inserting of a peg into a hole with small tolerance (about 100 μm). The above two tasks show the ability to: (1) perform tasks requiring fine force feedback, (2) maintain*

excellent control, (3) use of tools, (4) use of visual information to align axes, and (5) avoid jamming by activation of local compliance loop.

The "Operating Mechanical Pieces" task

During telerobotic tasks involving contact, like insertion of a peg into a hole (1/100 mm tolerance at full insertion) or general assembly operations, leads the slave arm wrist to comply (deflection) in response to the external forces and moments. Since the compliant force sensor is used to interface the slave tool to the robot wrist, a small deflection is consumed by the compliant structure of the force sensor. The wrist force sensor (compliant sensor) provides approximative information on the external forces. The computer that is attached to the sensor reads the external forces as measured at the sensing points and computes the forces and moments that are exerted at the slave arm tool using an analytical model of the sensor. The tool force information is sent to the computer (client) that is connected to the master arm. The client computer displays the force vector on the master arm by outputting the force information as torques to the master arm motors. The motors are linked to operator by using wire-based transmission system. Ideally, the operator feels the generated force as they are detected at the slave arm tool. In general, this powerful concept contributes in effectively extending the human manipulative capability in a remote workstation.

The following are some comments on the quality of force feedback in tasks involving contact with the environment:

i. Although the operator is tightly handling the master arm, the display of force feedback causes a rotation of the motor shaft, at master arm, as a response to the torque applied. This is partially due to the elasticity of the transmission wire in the master arm. The position sensor which is attached to the motor detects the change in the motor shaft angle and the master computer communicates the change as a small operator move. Although the angular reaction is small in magnitude it represents a pre-matured reaction to the displayed force feedback from the master arm before the same force is being sensed by the operator.

ii. In the telerobotic client-server system the reaction to an external force applied on the slave arm tool is distributed to (1) the local slave computer (server), and (2) the remote master arm. The idea is that both local (server) and remote (client) concurrently contribute in moving the slave arm in the direction that reduces the external forces. At the slave server, the detected forces and moments are selectively used to activate zero force regulation by constraining the slave motion over a direction, a plan, or a combination of the above. For example a spring effect can be obtained if the external force is periodically converted into an incremental displacement which is assigned to the slave arm. At the master arm the transmitted forces and moments

are converted into motor torques and displayed on the master arm. Since the local loop is much closer to the slave arm it immediately activates the force regulation mechanism until the external force is removed, i.e. reduced to zero. The operator reaction to displayed forces is slower due to network and protocols which make the remote force control useful for coarse corrections and the local regulation is adequate for fine force control.

iii. A car water-pump is used to carry out the assembling and disassembling operations. The pump consists of three circular parts which are; (1) a plastic cover, (2) a metallic base, and (3) a small DC motor to be assembled in the middle of the above two parts. Initially the metallic base is attached to fixed plate-form. The task is to grasp the motor and assemble it with the fixed metallic base. This requires excellent motion coordination and some force control at the final stage of the assembly. Part mating of the motor and the pump base is successfully achieved through cooperative (local and remote) corrections of axial alignment of motor while exerting adequate vertical force. Note that now the sub-assembly consists of the base and motor. In the assembly station, the axial direction of the above subassembly is not fixed and the operator must bring the plastic cover on motor top. The operation can be successfully accomplished after reasonable (one hour) operator training. The disassembling operation is easier as it involve pulling up the plastic cover and later the motor while maintaining fixed axial orientation with the assembly base. Please see the attached CD video clip on assembly of a water-pump.

iv. For the insertion, the operator tries to land the peg head (held by the slave arm gripper) as close as possible to the hole. The operator makes corrections on the position and orientation of the peg to align the peg axis with that of the hole using both stereo vision (axis alignment) and reflected force feedback (hole edge reaction and jamming forces). The stereo vision is not very helpful in making fine motion or fine axial corrections. On the other hand, contact forces like jamming forces lead the operator to search for a position and orientation of the peg that reduce the above forces while maintaining a light pressure in direction of the hole. This is also done in a cooperative manner by the local force regulation task and the remote operator. Some motion corrections leading to correct axial alignment causes the peg to move down the hole. The above correction strategy is repeated until the peg is fully inserted in the hole. The extraction exposes the operator to pull up the peg while using the cooperative correction strategy for misalignment axial errors. Please see the attached CD video clip on peg-in-hole insertion.

(c) *General operations like opening and closing of drawers, locking and unlocking of locks, operating various types of switches, etc. These require trajectory control, hybrid control (position, velocity, force, and mixed), use of tools, fine force feedback and directional control of force, and ex-*

cellent visual interactions.

The "General operations" task

i. The opening and closing of drawers involves the eye-hand motion coordination with light force feedback. Basically the operator pulls or pushes the drawer while maintaining a sufficient force to cause the motion. The eye-hand motion coordination proved to be excellent to carry out this task. Other tasks like locking and unlocking and operating switches involves simple activities that have been previously addressed. Please see the attached CD video clip on opening and closing of drawers.

ii. We carried out teleoperation of two types of switches: (1) a push-bottom switch for a soldering station and (1) a ON-OFF bistable switch for a power supply. Both operations require careful selection of the switch approach direction to ease accessibility. Another objective is to activate the switch while maintaining good view of the scene to make necessary corrections. Both operations were successful after a few attempts and the operator rapidly improved his performance in operation speed and in adopting the right task direction. It is noticed that the operator rapidly gained confidence in system and could minimize contact forces after a few attempts. The approximate time for activating the switch is about 3s when the operation is repeatedly performed. We also carried out a teleoperation task to unlock a lock by twisting the corresponding key. Please see the attached CD video clip on operating electrical switches.

iii. A task on wire-wrapping operation is added which involves manipulation requiring high level of accuracy in a small space. The task is to insert the head of the wire-wrapping tool into a needle of a specific microprocessor circuit to carry out the wire-wrapping operations. This is a typical operation to test the space scalability feature of the proposed telerobotic system. Here we extensively use (1) the scalability function of motion mapping function, (2) scalability of force feedback, (3) the camera zooming task, and (4) the shift function. The operator moves the wire-wrapping tool while aligning its head with the circuit needle and carries out the insertion. The operator needs to feel the vertical force component to avoid blocking due to a contact between the tool head and the needle because only a small central hole in the head must be inserted in the circuit needle. In this case the operator must (1) carry out corrections for the axis mis-alignment, and (2) try the insertion again. The distance between two circuit needles is about 2 mm. The above task was successful in making three insertions on three successive needles arranged in a line over a microprocessor bread-board including about 100 needles. Please see the attached CD video clip on wire-wrapping operation.

2. Task O-2: *Performance evaluation of overall telerobotics system which in-*

cludes: 1) evaluation of techniques used in the design of various components, 2) evaluation of equipment used in the design, and 3) evaluation of system performance with respect to the above typical tasks. The duration of this task is three months.

(1) EVALUATION OF THE TECHNIQUES USED IN THE DESIGN

(a) The designed master arm is a light serial structure having an excellently distributed 6 DOFs. The operator handle is set at the end of the third link. Thus the operator hand position is controlled by the first three DOFs. There is a deliberate effort to uncouple the operator hand position from hand orientation. This is accomplished by setting the operator hand at end of the first three dofs. While the last 3 dof are designed so that their rotation axes intersect at the operator hand center. The operator may change the orientation of his hand without affecting its position.

(b) The kinesthetic geometric mapping of motion from the operator hand (holding the master arm handle) to the slave arm tool is excellently serving its purpose. Due to mapping of incremental position and orientation from user hand to slave tool the operator mentally think he is in the frame of reference of the slave arm tool. This is done using mathematical mapping functions described in Section 8. As a result the operator feels he is directly acting on the slave tool or the manipulated object.

(c) The computer aided teleoperation functions are extremely useful during teloperation. All the tasks described above are heavily relaying on these functions. The *Shift Function* is critical to achieve this task. It is one of the most important Computer Aided Teleoperation functions (See 8). It has a very practical activation, a simple key in the master arm handle. This way the user is not distracted or disturbed by activating or unactivating the above function. During teleoperations, the operator can repeatedly shift the master arm to his dexterity area without affecting the current position of slave arm which is freezed during the shift operation.

(d) The electro-mechanical structure of the proposed teleoperation system consists of (1) the slave arm, (2) the master arm represented by its motors and its position sensors, (and (3) the human operator. The master arm handle (operator hand) is linked to the master arm motors using a wire-based transmission system. The wires are made of flexible steel. For example the transmission structure linking dof 6 to motor 6 is made of a wire that is 2.5 m long due to the multiple closed loop structure of the transmission system. This approach was adopted to improve the maintainability of the transmission because a local wire fault can be isolated and repaired without overall disassembling of the whole transmission wheels. On the other hand, a tradeoff must be made in setting up two conflicting factors which are: (1) a low friction master arm (operator), and (2) a high-stiffness transmission system (interaction). The need for a

low friction transmission is motivated by the need for a high-quality back-drivability for the master arm. While a high-stiffness transmission system is motivated by the need for a flexible and stiff connectivity between operator hand and master arm motors. The current setting represents a tradeoff between an acceptable level of friction at the level of the master arm handle and a tight transmission system linking the operator hand to various motors that are used to display the reflected force feedback.

During telerobotic tasks involving contact like insertion of a peg into a hole or general assembly operations the contact leads the slave arm wrist to comply (deflection) with the external forces and moments. Since the compliant force sensor is used to interface the slave tool to the robot wrist, the deflection is consumed by the compliant structure of the force sensor. The computer that is

attached to the sensor reads the external forces as measured at the sensing points. The force information is sent to the computer (client) that is connected to the master arm. The client computer displays the force vector on the master arm by outputting the force information as torques to the master arm motors. As described above the motors are linked using wire-based transmission system to the operator handle. Thus the operator feels the generated force as they are detected at the slave arm tool. In general, this powerful concept contributes in effectively extending the human manipulative capability in a remote workstation.

Following are comments on the quality of force feedback in tasks involving contact with the environment:

i. Although the operator is tightly handling the master arm handle the displayed forces lead the motors to rotate by small proportional angles. In order for the operator to feel this displayed force the gain may be increased so that the generated motor toque increases the wire tension. The motor rotation is due to the light elasticity of the transmission wire. The position sensor which is attached to the motor shaft detects the change in the motor shaft angle and the master computer communicates the change as a small operator move. Although the angular reaction is small in magnitude it represents a pre-matured reaction to the displayed force feedback from the master arm before the same force is being sensed by the operator.

ii. To overcome the above problem the tendency of the designer is to increase the wire tension and reduce the magnitude of the reflected force feedback. To minimize pre-matured reaction the designer may: (1) reduce the magnitude of the displayed force up to some level to make sure its magnitude can still be detected by the human arm, (2) reduces the size of the master arm to reduce wire lengths, and (3) use less elastic steel tape as opposed to steel wire at the cost of changing the transmission wheels. One may also adopt a combination of the following practical options: (1) use of less elastic wire in the transmission system to increase transmission stiffness, (2) accept a reduction of master arm work-space and adopt a smaller size master

arm to reduce wire length, (3) minimize overall master arm inertia and friction by adopting a very light structure.

iii. Generally back-drivable robot arms do not need to use explicit force sensor which is often estimated using motor torques and gravity effects. However, robot arms using rigid motion transmission system use wrist force sensors to provide reflected force feedback when they are used as slave arms. The PUMA 560 belongs to the second category and the compliant wrist force sensor is used for the detection of external forces and moments.

The wrist force sensor (compliant sensor) provides approximative information on the applied external forces. Using experimental data we found that the overall force errors is about 10% due to the nonlinear effects caused by the elastic structure, accuracy of used optical sensors (about 5 % errors), and effects of motion dynamics and gripper dynamics on the sensor data. On the other hand the sensor has a simple electronics and computer interfacing using a standard PCI card.

The sensor is regularly sampled by a program (thread) running on the slave arm computer and an external force is represented by three-force and three-moment components. The reaction to an external force applied on the slave arm tool is distributed to (1) the local slave computer (server), and (2) the remote master arm. The idea is that both local (server) and remote (client) concurrently contribute in reducing moving the slave arm in the direction that reduces the external forces. At the slave server, the detected forces and moments can be selectively activated to orchestrate a zero force regulation over a given sub-set of DOFs. For example a spring effect can be obtained in real-time if the external force is periodically converted into an incremental displacement which is assigned to the slave arm. At the master arm the transmitted forces and moments are converted into motor torques and displayed on the master arm. Since the local loop is much closer to the slave arm it immediately activates the force regulation until the external force is removed. The operator reaction to displayed forces is slower due to network and protocols which makes the remote force control useful for coarse corrections.

(e) The processing and network delays were addressed as follows:

i. The team designed and implemented a telerobotic system that consists of master (client) and slave (server) stations which are usually connected by a computer network. A reliable real-time connection between master and slave systems is proposed using *Distributed Components (.NET Remoting)*. This has a number of benefits such as software reusability, ease of extensibility, debugging, and data encapsulation. It is based on most advanced software tools like *.NET Framework* that promise definite advantages over *DCOM (Distributed Component Object Model)* and *RPC (Remote Procedure Call)*, previously used for distributed applications. The components communi-

cate with each other using *.NET Remoting* and *SOAP (Simple Object Access Protocol)* that automatically handle the network resources and data transfer while isolating the components from network protocol issues. This enhances the data security as well as facilitates easy deployment. Implementing telerobotics using the proposed approach gives the advantage of a multi-threaded execution needed to effectively realize multi-streaming of force, command and stereo data over a LAN. To minimize delays we engineering the telerobotic client-server, motion coordination system, and stereo vision server using concurrent programming. The optimized system has refreshing rates of 50 Hz in real-time transfer of commands, 76 Hz for reflected force feedback, and 17 fps for stereo vision over a 100 Mbps LAN.

ii. The team implemented the augmented reality system as part of the strategy to "Reduce Network Delays in Telerobotics". Please see Section 6.6 for the details. The operator carries out task planning by creating a virtual ball (representing the slave arm gripper) and moving it to the vicinity of a target object. This consists of using a simple pointer in the stereo image (client) that is set in a relative position to current display of the robot gripper. The position of the pointer is controlled by the client (for example through a mouse). The setting of the point relative to the gripper needs to take into consideration the geometric relationship, in the slave station, linking the robot gripper frame to its reference camera frame (for both cameras). This enables sending in one packet a trajectory to slave arm to follow up in addition to final setting of the gripper which greatly reduce network delays in telerobotics. Augmented reality contributes in reducing delays by sending only planned tasks that have been refined and validated by the operator which reduces the need for intensive interaction with the remote site.

iii. To reduce processing delays and overhead, the PUMA robot was run under a fine trajectory control so that only coarse master arm positions need to be send while local slave control is continuously active. The interface allows: (1) downloading programs to the PUMA to provide local trajectory control at initialization only, and (2) uploading state information on-line to control the PUMA from the server station. It was found [70] and 6.7 that using the above approach, the dominant delays in the LAN environment are only due to robot motion. This approach provides the best possible optimization to reduce the mechanical delays occurring during the PUMA motion by using the PUMA system.

iv. The team also minimized the network delays by operating the client and server in an asynchronous or independent way. For example, instead of waiting for the arrival of increment ΔX_{new} from the client, the server continuously reads θ and computes $X = G(\theta)$. When the increment ΔX_{new} arrives from the client, the server picks up the most recent value of $X = G(\theta)$ and uses it in computing $\theta_{new} =$

$G^{-1}(X + \Delta X_{new})$. Thus the computation time $X = G(\theta)$ is overlapped with the network communication delays. The same process is done at the client which continuously reads the θ vector of the "master arm", computes $X = G(\theta)$, and evaluates the latest increment ΔX_{new} which is sent to the server if a ready signal is available. The important issue to note here is that all computations are done, in the server or the client, in an asynchronous or independent way to overlap their times with the motion or communication times which result from the motion synchronization of the PUMA with the master arm. In summary we minimize the network delays by carrying out all computations in an asynchronous way in both client and server but we keep motion synchronization by using fast messaging system (busy-wait or ready) between the PUMA motion and the client commands. The motion synchronization system and fast messaging are parts of the *Reliable Client-Server Telerobotic System* that is described in Section 6.4.

(2) EVALUATION OF EQUIPMENT USED

(a) Watching the scene in Stereo 3D using a Head Mounted Display provides critical information about the remote scene that is reliable and accurate. The zooming function allows the operator to focus on a small working area. The operator rapidly develops self-confidence in the eye-hand motion coordination system. The quality of stereo views is excellent and this is more important than the refreshing rate of stero picture which only affects the speed of operations. The HMD resolution is high enough (284x1024 pixels). However to get excellent focusing we must make sure of the following effects. First the two cameras must have an accurate horizontal disparity of 6 cm with the least possible vertical shift to eliminate any degradation due to tilted cameras. Second the best stereo effects are obtained for a distance from cameras to scene between 100 mm and 200 mm for Sony Handicam Digital Cameras. Third the zoom levels should be exactly the same for convergence of left and right images. Fourth the effect of scene lightning is important because normal florescent light is found to cause flickering, and hence user uncomfort, proper lightning is very important. Overall, we are quite satisfied with the performance, quality of 3D views, and depth perception of the local and networked version of the 3D HMD based visualization system. It proved to be the most critical information available to the operator.

Note that camera zooming function and the motion scalability function present excellent operating tools for carrying out tasks in very small scale like a few mms as well as enabling the generation of fast motion of up to 0.5 mps in larger operating areas. The above function enables scaling up and down the whole eye-hand motion coordination system which is certainly one critical issue in telerobotics.

(b) The PUMA 560 is a 6 DOF arm having a rigid transmission system. This

robot arm was used as a slave arm in the proposed telerobotic system. However, VAL II operating system and programming does not allow fine control of the trajectory. In addition the rigid transmission system is not adequate for teleoperation where wire-based slave arm are more adequate for general purpose teleoperation tasks. We recommend to use a wire-based slave arm with fully documented software so that to achieve fine trajectory control by the server component in the telerobotic system. One interesting alternative is to locally design a wire-based, stiff, slave arm based on the structure that is proposed for the master arm in this project.

(c) The last link of slave arm is too long because of its serial structure which contains the last robot link (50 mm), the force sensor (50 mm), and the locally designed gripper (160 mm). As a result the last slave link length including all the above is 260 mm. Thus changing the orientation of the slave arm leads the last link to scan a significant portion of the workspace which may distract the operator attention due to possible collision in addition to the burden of positioning and orienting the large body of last link. We recommend to reduce the length of the last link as much as possible preferably below 150 mm to improve quality of apprehension in telerobotics.

(3) EVALUATION OF SYSTEM PERFORMANCE

(a) We used a 100 Mbps LAN, 2.0 GHZ P-IV client and server computers, 1 GB DRAM, and accelerated graphics card with 256 MB DDR memory. Each force data packet contains 6 double values which equal $6 \times 8 = 48$ bytes. For force and video multistreaming the refreshing rate of the inter-arrival times of force packets is 250 Hz. To provide guaranteed performance to the force packet sampling rate we need to access the worst scenario in which we have intensive video transfer. Using the above data we isolate the instances during which video activities are intensive and study this effect on force packets. The mean value of the inter-arrival times of stereo video frames is 87.57 ms with a 90% confidence interval falling between 72 and 107 ms. For force, command, and video multistreaming when all of the three force, command and video threads are invoked simultaneously, for the force packets we get a mean inter-arrival rate of 1.1 ms while 100% of the population remains under 8 ms. The video transfer rate for an image to move from camera to DRAM is 24 ms using DirectShow. The stereo video client-server transfers two images (stereo frame) of size 288×360 pixels at a rate of 17-18 fps with a delay of around 58 ms only. The results of thread egineering and software optimization is an effective multistreaming running with a sampling rate of 17 Hz for stereo video, 76 Hz for force feedback, and 50 Hz for operator commands over a commodity 100 Mbps LAN. Thanks to multithreading for the graceful degradation of real-time force feedback data in periods of intensive video steaming.

(b) In the designed master arm, the motion uncoupling of translation and orientation proved to be one critical aspect in the design. It optimally maps

to human arm because it allows natural separation of control of available DOFs. As a result the operator feels equal mechanical impedance in all orientation directions. Changing orientation does not expose the operator to the structure effects like the gravity forces or mechanical coupling within the master arm. The master arm structure significantly contributes to ease the telerobotic tasks by letting the operator making abstraction of the master arm structure due to orientation iso-impedance. The operator can focus on the remote task without paying attention to the mechanical structure of the master arm. The designed master arm is likely to be a transparent tool for telerobotics. We complemented the uncoupled master arm with powerful mathematical motion mapping function as described in Section 8. The structure and motion mapping strategy is the result of many years of experience in the design, manufacturing, and testing of passive master arms at the Robotics Lab, Computer Engineering Department, KFUPM. One of these leading structures is presented in Section 6.1.3.

(c) The kinesthetic mapping of motion is excellently serving its purpose (Section 8). Based on the operating tasks the operator feels he is directly acting on the slave tool or the manipulated object. Since this is a natural capability of human arm the operator needs not to see his arm or the master arm during direct teleoperation but only need to concentrate his view on the remotely manipulated object or tool. The implemented functions present the basis for motion mapping in Computer Aided Teleloperation.

(d) The present computer aided teleoperation functions are extremely useful during teloperation. During teleoperations, the operator can repeatedly shift the master arm to his dexterity area without affecting the current position of slave arm which is freezed during the shift operation. With short training period the operator can bring the slave gripper in the vicinity of desired object and set up the final position and orientation of gripper. The gripper orientation is progressively set and the operator advances the gripper towards the target while continuously centering the jaw. The object is grasped and the operator moves up the slave gripper to travel to target position. In addition the operator may use the motion scalability function (Section 8) when operating in the vicinity of the object and when accurate positioning is required. In this case his motion can be linearly scaled down by up to a factor of 15 both in position and orientation. This provides another critical function that consists of a potentiometer which is controlled by the index of the operator located on the master arm handle. It is implemented at master arm handle so that its activation does not cause distraction to the operator. The experience with the designed master arm and activation keys shows that these functions are activated in a smooth fashion requiring insignificant attention and time from the operator.

(e) Elasticity in the wire-transmission causes some problems to the display of reflected force feedback. Although the operator is tightly handling the master arm handle the displayed forces lead the motors to rotate by small

proportional angles. In order for the operator to feel this displayed force the gain may be increased so that the generated motor toque increases the wire tension. The motor rotation is due to the light elasticity of the transmission wire. The position sensor which is attached to the motor shaft detects the change in the motor shaft angle and the master computer communicates the change as a small operator move. Although the angular reaction is small in magnitude it represents a pre-matured reaction to the displayed force feedback from the master arm before the same force is being sensed by the operator. To overcome the above problems we recommend to: (1) use of less elastic wire in the transmission system to increase transmission stiffness, (2) accept a reduction of master arm work-space and adopt a smaller size master arm to reduce wire length, (3) minimize overall master arm inertia and friction by adopting a very light structure.

6 Results and discussion

This section presents the detailed analysis, results, and discussion for each major tasks of the project. We group related issues in one single topic with a number of sub-sections. Each topic is part of the methodology used for addressing a specific project task.

In this section presents the analysis of the following issues:

1. The direct and inverse kinematic models used throughout this project,
2. The implemented functionalities at master and slave arms,
3. The visualization system under local and networked operations,
4. The design of the rigid and compliant force sensors,
5. The difficulties in acquiring the master arm,
6. The real-time Multi-threaded distributed Component system,
7. The stereo vision component,
8. The augmented reality component,
9. The man-machine interface to monitor the operator motion,
10. The description of the manufactured master arm.

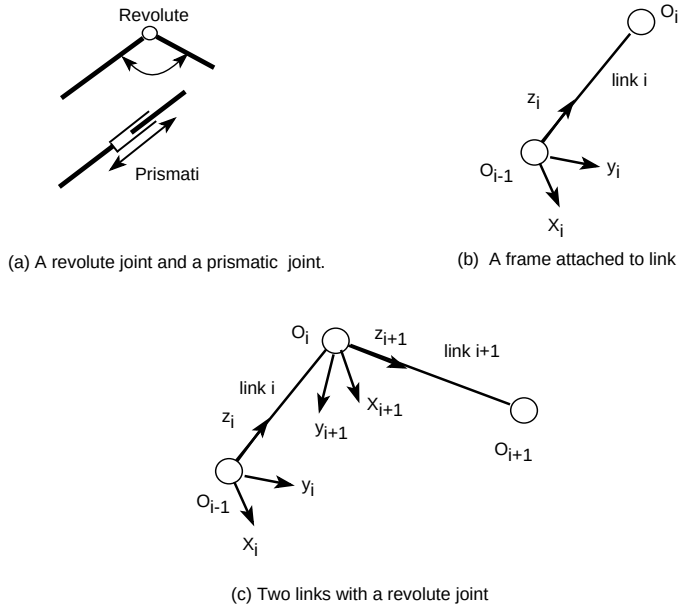


Figure 1: Frame of reference and geometric model of an articulated system

6.1 Modelling of the master and slave arms

An industrial robot is a multi function manipulator that can be modelled as an open chain of rigid bodies, called *links*, connected in series by kinematic joints. The function of the joint is to control the motion between the links. The first link is attached to the supporting base by the first joint, and the last link contains the end effector or other type of manipulator device. Each *joint-link* pair constitutes one degree of freedom. An n degree of freedom manipulator contains n joints, or in more general terms, n link-attached coordinate system. The joints and links are numbered starting from the base. The lowest joint is fixed to the reference coordinate system, while the highest joint is fixed to the local coordinate system of the end effector. Robotic joints can be categorized as either *revolute* or *prismatic* joints as shown in Figure 1-(a). A revolute joint allows link L_{i+1} to rotate with respect to the previous link L_i. A rotation angle θ_{i+1} can be used to define the angular position of L_{i+1} relative to L_i. This is shown on Figure 1-(b) and (c). A prismatic joint allows a link to translate with respect to the previous link. A translation variable θ_{i+1} can also be used to define the linear position of L_{i+1} relative to L_i. Our project is implemented on the PUMA-560 robot arm which will be presented in more detail in the next Section.

6.1.1 The PUMA-560 manipulator arm

The PUMA-560 robot arm has six degrees of freedom and all are rotational joints. The last three have concurrent rotation axes, which simplify their geometric and kinematic models. All the joints are driven and controlled by DC-Servomotors. The servomotors are equipped with electromechanical brakes that can lock the arm in a fixed position. The brakes are released by the controller when the arm power is on. The components of the robot arm are the *Trunk, Shoulder, Upper Arm,*

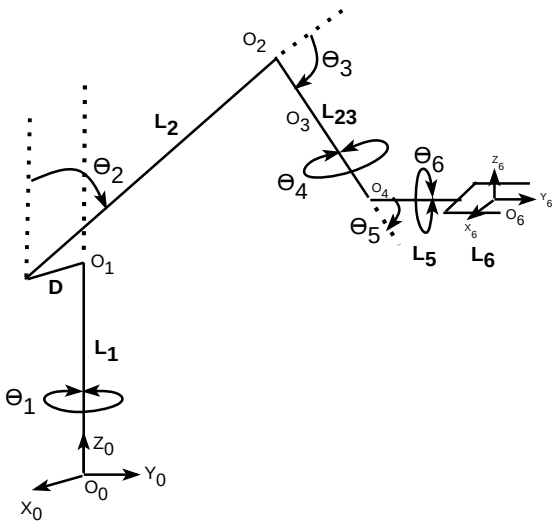


Figure 2: The kinematic model of the PUMA 560 robot arm

Forearm, Wrist and *Geometric Model* as shown in Figure 2. Functionally this arm can be divided into two parts which are the "transporter" and the "effector" parts. The transporter is responsible for transferring and positioning the effector which include the grasping system and the work piece. On the other hand, the effector is responsible for the orientation of the arm. The transporter includes three links that are the shoulder, elbow and forearm, while the end effector part includes the *pitch, yaw* and *roll*.

The motion coordination concept

The problem is to determine the position and orientation of objects in the 3-dimensional space. The objects are the links of the manipulator, and the tools with which it deals. These objects are described by just two attributes: their position and their orientation. In order to describe the position and orientation of an object in space, we will attach a frame of reference to each link. The frame of reference is defined using three orthogonal vectors $\{X, Y, Z\}$. Figure 1-(b) gives an example of frame R_1 translation and rotation relative to frame R_0.

The position of the manipulator is generally described by giving a description of the tool frame, which is attached to the end effector, relative to the base frame which is attached to the fixed base of the manipulator as shown in Figure 2.

The relative position and movement of the individual links with respect to their preceding links provide a description of an entire articulated structure in the operating space and formulate a mathematical model of a kinematic chain of the robotic system. Since robotic manipulation can be achieved only by maneuvering the arm linkages in the task's environmental space, robot kinematics is an important tool in work space design, trajectory planning and motion rate control. Kinematics is concerned with the analytical description of spatial position, orientation, displacement, velocity and acceleration. There are two fundamental problems in studying robotic kinematics: the direct kinematic and the inverse kinematic problem. The direct kinematic problem involves the determination of the position and orientation of the

end effector with respect to the reference coordinate system, given the joint variables of the robot arm. The inverse kinematic problem, on the other hand, involves the determination of the joint controlled variables, given the position and orientation of the end effector.

Selection of cartesian frames

In the 2-dimensional space there are three degrees of freedom (dof): X, Y and θ orientation parameter. In the 3-dimensional space there are six DOF ,three position parameters: X, Y, Z and three angular orientation parameters specifying the orientation of the gripper in the space. We specify both position and orientation using a translation vector (3x1) and a rotation (orientation) matrix (3x3). These notations are opposed to the Denavit-Hartenberg (DH) notations which are used in the majority of cases and represented by a 4 x 4 homogeneous matrix that transforms a vector from one coordinate system to another. A Cartesian coordinate system is defined in the three dimensional space, by introducing three orthogonal vectors X, Y, Z. A frame can be defined by the orthogonal vectors with origin O. We say that the link L_{i+1} revolute with respect to link L_i when frame R_{i+1} rotates relative to either axes X_i, Y_i or Z_i as shown in Figure 1-(c). The end point O_{i+1} of L_{i+1} can be associated a vector O_iO_{i+1} which will be denoted by $O_iO_{i+1,i}$ to indicate that the vector is observed in frame R_i. $M_i^{i+1} = [X_{i+1,i}, Y_{i+1,i}, Z_{i+1,i}]$ is the transfer matrix from frame R_{i+1} to R_i, which represents the rotation between links L_i and L_{i+1}. Therefore, the link vector $O_iO_{i+1,i}$ can be expressed as follows:

$$O_iO_{i+1,i} = M_i^{i+1}.O_iO_{i+1,i+1} \tag{3}$$

where $O_iO_{i+1,i+1}$ denote the vector O_iO_{i+1} observed in frame R_{i+1}. Note that the vector $O_iO_{i+1,i+1}$ has a simple expression because link L_{i+1} is parallel to axis Z_{i+1}. Therefore, the position vector $O_0O_{n,0}$ can be decomposed as the sum of the link vectors:

$$O_0O_{n,0} = \sum_{i=1}^{n} M_0^i.O_{i-1}O_{i,i} \tag{4}$$

vector $O_{i-1}O_{i,i}$ has a simple expression because it is represented with respect to its own frame of reference R_i. Both the vector $O_0O_{i,0}$ and $\sum_{i=1}^{n} M_0^i$ can be expressed in a recursive form, leading to:

$$M_0^i = M_0^{i-1}.M_{i-1}^i \tag{5}$$

$$O_0O_{i,0} = O_0O_{i-1,0} + M_0^i.O_{i-1}O_{i,i} \tag{6}$$

where M_0^{i-1} is the transfer matrix from R_0 to R_{i-1}, and M_{i-1}^i is the transfer matrix from R_i to R_{i-1}. The orientation matrix

$$M_0^n = [X_{n,o}, Y_{n,o}, Z_{n,o}] = \begin{pmatrix} X_x & Y_x & Z_x \\ X_y & Y_y & Z_y \\ X_z & Y_z & Z_z \end{pmatrix} \tag{7}$$

is used to determine the orientation of the frame R_n with respect to the frame R_0, and the position vector $O_0O_{n,0}$ is used to determine the position of the frame R_n with respect to the frame R_0

$$O_0O_{n,0} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{8}$$

The fundamental direct kinematics problem for robotic manipulators is to formulate kinematic equations or what we call geometric model that transform the joint space to Cartesian space. The robot end effector is represented by position vector and orientation matrix in the three dimensional space. Therefore, a point in the joint space can be converted to a position and an orientation in the Cartesian space using the geometric model. Transformation functions relate the position and orientation of the end effector coordinate system to the base coordinate system. These transformations are very important, since a robot is controlled in the joint space, whereas most of the tasks are done in the Cartesian space. The geometric model is denoted by $E = G(\theta)$, where the end effector vector E is a function of the joint variables vector θ. Given the joint variables $\theta_1, \theta_2, ...\theta_n$ we can compute the basic representation of the end effector with respect to a reference Cartesian coordinate R_0 by using the direct geometric model:

$$(\theta_1, \theta_2, ...\theta_n)^t \rightarrow \ \{O_0O_{n,0}(\theta), M_0^n(\theta)\} \tag{9}$$

The basic geometrical representation of the arm is reduced to the following expression:

$$G(\theta) = \{O_0O_{n,0}(\theta), M_0^n(\theta)\} \tag{10}$$

6.1.2 Direct geometric model of PUMA-560

Now we will develop a mathematical model for representing the geometric configuration of the links and joints of the PUMA-560. This manipulator arm has six links and each link can rotate with respect to a reference coordinate system. The geometric model solutions start by assigning a link attached coordinate frame to each link of the manipulator, it then tabulates these link parameters and establishes the transformation matrix M_i^{i-1} for each link. The state of the end effector link attached frame, with respect to the base coordinate frame, can be determined by means of the following information:

- The robot hand orientation matrix $M_0^n = [X_n^0, Y_n^0, Z_n^0]$, determines the orientation of frame R_n with respect to frame R_0.
- The position vector $O_0O_{n,0}$, references the origin of R_n with respect to R_0.

The orientation matrix M_0^i is the products of the rotation matrices:

$$M_0^n = M_0^1.M_1^2.M_2^3...M_{n-1}^n \tag{11}$$

The position vector can be determined as follows:

$$O_0O_{n,0} = O_0O_{n-1,0} + M_0^n.O_{n-1}O_{n,n} \tag{12}$$

The basic concept of the mathematical description of the frames is to provide the translation and rotation characteristics of each link. Evaluation of rotation matrices and position vectors will help later to solve the inverse kinematics problem. The reference arm position required to build the geometric model for the PUMA-560 robot arm is shown in Figure 2.

In order to develop the geometric model for the PUMA-560 robot arm we will establish the following scheme:

- Every joint is attached to a frame of reference.
- Link L_i is between frame R_{i-1} and frame R_i.
- M_i^{i+1} is used to represent the rotation between links L_i and L_{i+1}
- The orientation matrix $M_0^n = [X_n^0, Y_n^0, Zn^0]$ determines the orientation of frame R_n with respect to frame R_0.
- The position vector is given by the following expression:

$$O_0O_{i,0} = O_0O_{i-1,0} + M_0^i O_{i-1}O_{i,i} \tag{13}$$

The frame representation for the robot arm is shown in Figure 1-(b). Functionally our robot arm, having six degrees of freedom and they can be divided into two substructures which are the Transporter and the Effector parts. We have used the following topological form to describe the geometric structure of the arm:
The transporter part is defined by three revolute joints:

$$Linkl(ROTZ(\theta_1), Z(L1))$$

$$Link2(ROTX(\theta_2), Z(L2))$$

$$Link3(ROTX(\theta_3), Z(L3))$$

The effector part is defined by three revolute joints:

$$Link4(ROTZ(\theta_4), Z(L4))$$

$$Link5(ROTX(\theta_5), Z(L5))$$

$$Link6(ROTZ(\theta_6), Z(L6))$$

where $ROTZ(\theta_i)$ is the rotation of link i between frames R_i and R_{i-1} about the Z_{i-1} axis, and $Z(L_i)$ indicates that the link body L_i is along the Z_i vector. Consider L_1 which is revolute and defined as a rotation about the Z_o axis, and the link body L_1 is along vector Z_1 of frame R_1.
We start by computing the transformation matrix between frames R_1 and R_0

$$M_o^1 = ROTZ(\theta_1) \tag{14}$$

$$M_o^1 = \begin{pmatrix} C_1 & -S_1 & 0 \\ S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{15}$$

The solution to the position vector $[O_0O_{1,0}]$ yields:

$$O_0O_{1,0} = M_0^1.O_0O_{1,1} \tag{16}$$

$$O_0O_{1,0} = \begin{pmatrix} C_1 & -S_1 & 0 \\ S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} d \\ 0 \\ \ell_1 \end{pmatrix} \tag{17}$$

$$O_0O_{1,0} = \begin{pmatrix} dC1 \\ dS1 \\ \ell_1 \end{pmatrix} \tag{18}$$

For the second end point O_2 we have:

$$M_0^2 = M_0^1.M_1^2 = ROTZ(\theta_1).ROTX(\theta_2) \tag{19}$$

$$M_0^2 = \begin{pmatrix} C_1 & -S_1 & 0 \\ S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 1 & 0 & 0 \\ 0 & C_2 & -S_2 \\ 0 & S_2 & C_2 \end{pmatrix} \tag{20}$$

$$M_0^2 = \begin{pmatrix} C_1 & -S_1C_2 & S_1S_2 \\ S_1 & C_1C_2 & -C_1S_2 \\ 0 & S_2 & C_2 \end{pmatrix} \tag{21}$$

The position vector $O_2O_{2,0}$ is given by:

$$O_0O_{2,0} = O_0O_{1,0} + M_0^2O_1O_{2,2} \tag{22}$$

The link body L_2 is along vector Z_2 of frame R_2 and is shifted by value d in the X axis direction from the link body L_1.

$$O_0O_{2,0} = O_0O_{1,0} + M_0^2. \begin{pmatrix} 0 \\ 0 \\ \ell_2 \end{pmatrix} \tag{23}$$

$$O_0O_{2,0} = \begin{pmatrix} C_1.d + S_1S_2\ell_2 \\ S_1.d - C_1S_2\ell_2 \\ \ell_1 + C_2\ell_2 \end{pmatrix} \tag{24}$$

The link body L_3 is along vector Z_3 and rotates around X axis.
The orientation matrix M_0^3 is given by:

$$M_0^3 = M_0^2.M_2^3 = M_0^2.ROTX(\theta_3) \tag{25}$$

$$M_0^2 = \begin{pmatrix} C_1 & -S_1C_2 & S_1S_2 \\ S_1 & C_1C_2 & -C_1S_2 \\ 0 & S_2 & C_2 \end{pmatrix} \tag{26}$$

$$M_0^3 = \begin{pmatrix} C_1 & -S_1C_2 & S_1S_2 \\ S_1 & C_1C_2 & -C_1S_2 \\ 0 & S2 & C2 \end{pmatrix} . \begin{pmatrix} 1 & 0 & 0 \\ 0 & C_3 & -S_3 \\ 0 & S_3 & C_3 \end{pmatrix} \tag{27}$$

Given that :
$C_{23} = C_2C_3 - S_2S_3$
$S_{23} = C_2S_3 + C_3S_2$
Then

$$M_0^3 = \begin{pmatrix} C_1 & -S_1C_23 & S_1S_23 \\ S_1 & C_1C_23 & -C_1S_23 \\ 0 & S23 & C23 \end{pmatrix} \tag{28}$$

The position vector $O_0O_{3,0}$ is given by:

$$O_0O_{3,0} = O_0O_{2,0} + M_0^3O_2O_{3,3} \tag{29}$$

Then

$$O_0O_{3,0} = O_0O_{2,0} + M_0^3.\begin{pmatrix} 0 \\ 0 \\ \ell_3 \end{pmatrix} \tag{30}$$

$$O_0O_{3,0} = \begin{pmatrix} C_1.d + S_1S_2\ell_2 \\ S_1.d - C_1S_2\ell_2 \\ \ell_1 + C_2\ell_2 \end{pmatrix} + \begin{pmatrix} C_1 & -S_1C_{23} & S_1S_{23} \\ S_1 & C_1C_{23} & -C_1S_{23} \\ 0 & S_{23} & C_{23} \end{pmatrix}.\begin{pmatrix} 0 \\ 0 \\ \ell_3 \end{pmatrix} \tag{31}$$

$$O_0O_{3,0} = \begin{pmatrix} C_1.d + S_1S_2\ell_2 + S_1S_{23}\ell_3 \\ S_1.d + C_1S_2\ell_2 + C_1S_{23}\ell_3 \\ \ell_1 + C_2\ell_2 + C_{23}\ell_3 \end{pmatrix} \tag{32}$$

The link body L_4 is along vector Z_4 and rotates around Z axis.
For the orientation matrix we have :

$$M_0^4 = M_0^3.M_3^4 = M_0^3.ROTZ(\theta_4) \tag{33}$$

$$M_0^4 = \begin{pmatrix} C_1 & -S_1C_{23} & S_1S_{23} \\ S_1 & C_1C_{23} & -C_1S_{23} \\ 0 & S_{23} & C_{23} \end{pmatrix}.\begin{pmatrix} C_4 & -S_4 & 0 \\ S_4 & C_4 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{34}$$

$$M_0^4 = \begin{pmatrix} C_1C_4 - S_1S_4C_{23} & -C_1S_4 - S_1C_4C_{23} & S_1S_{23} \\ S_1C_4 + C_1C_4C_{23} & -S_1S_4 + C_1C_4C_{23} & -C_1S_{23} \\ S_{23}C_4 & C_4S_{23} & C_{23} \end{pmatrix} \tag{35}$$

For the position vector $O_0O_{4,0}$ we have:

$$O_0O_{4,0} = O_0O_{3,0} + M_0^4O_3O_{4,4} \tag{36}$$

$$O_0O_{4,0} = O_0O_{3,0} + M_0^4.\begin{pmatrix} 0 \\ 0 \\ \ell_4 \end{pmatrix} \tag{37}$$

given that :

$$L_{34} = L_3 + L_4 \tag{38}$$

$$O_0O_{4,0} = \begin{pmatrix} C_1.d + S_1(S_2\ell_2 + S_{23}\ell_{34}) \\ S_1.d - C_1(S_2\ell_2 - S_{23}\ell_{34}) \\ \ell_1 + C_2\ell_2 + C_{23}\ell_{34} \end{pmatrix} \tag{39}$$

To simplify let

$$M_0^4 = \begin{pmatrix} X_{x4} & Y_{x4} & Z_{x4} \\ X_{y4} & Y_{y4} & Z_{y4} \\ X_{z4} & Y_{z4} & Z_{z4} \end{pmatrix} \tag{40}$$

And let

$$O_0O_{4,0} = \begin{pmatrix} X_4 \\ Y_4 \\ Z_4 \end{pmatrix} \tag{41}$$

The link body L_5 is along vector Z_5 and rotates around X axis.
Rotation matrix M_0^5 is given by:

$$M_0^5 = M_0^4.M_4^5 = M_0^4.ROTX(\theta_5) \tag{42}$$

Then

$$M_0^5 = \begin{pmatrix} X_{x4} & C_5(Y_{x4}) + S_5(Z_{x4}) & -S_5(Y_{x4}) + C_5(Z_{x4}) \\ X_{y4} & C_5(Y_{y4}) + S_5(Z_{y4}) & -S_5(Y_{y4}) + C_5(Z_{y4}) \\ X_{z4} & C_5(Y_{z4}) + S_5(Z_{z4}) & -S_5(Y_{z4}) + C_5(Z_{z4}) \end{pmatrix} \tag{43}$$

The position vector $O_0O_{5,0}$ is expressed as in the following:

$$O_0O_{5,0} = O_0O_{4,0} + M_0^5O_4O_{5,5} \tag{44}$$

$$O_0O_{5,0} = O_0O_{4,0} + M_0^5.\begin{pmatrix} 0 \\ 0 \\ \ell_5 \end{pmatrix} \tag{45}$$

$$O_0O_{5,0} = \begin{pmatrix} X_4 - S_5(-C_1S_4 - S_1C_4C_23) + C_5(S_1S_23).\ell_5 \\ Y_4 - S_5(-S_1S_4 + C_1C_4C_23) + C_5(-C_1S_23).\ell_5 \\ Z_4 - S_5(C_4S_23) + C_5(C_23).\ell_5 \end{pmatrix} \tag{46}$$

Finally, the link body L_6 is along vector Z_6 and rotate around Z axis. The basic orientation matrix M_0^6 is computed as follows:

$$M_0^6 = M_0^5.M_5^6 = M_0^5.ROTZ(\theta_6) \tag{47}$$

$$M_0^6 = \begin{pmatrix} X_x & Y_x & Z_x \\ X_y & Y_y & Z_y \\ X_z & Y_z & Z_z \end{pmatrix} \tag{48}$$

The final components of the orientation matrix are:

$$X_x = C_6(C_1C_4 - S_1S_4C_{23}) + S_6(-C_1C_5S_4 - S_1C_4C_5C_{23}) + S_1S_5S_23)$$

$$X_y = C_6(S_1C_4 + C_1C_4C_{23}) + S_6(-S_1S_4C_5 + C_1C_4C_5C_23) - C_1S_5S_23)$$

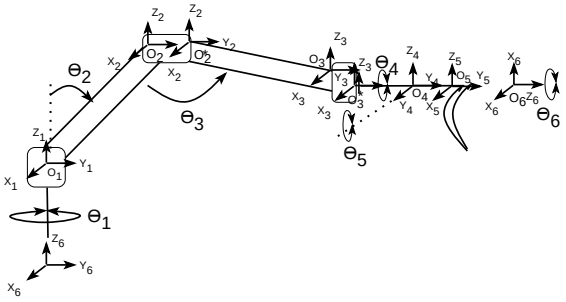
$$X_z = C_6(S_{23}C_4) + S_6(C_4C_5S_{23} + S_5C_{23})$$

$$Y_x = -S_6(C_1C_4 - S_1S_4C_{23}) + C_6(-C_1C_5S_4 - S_1C_4C_5C_{23}) + S_1S_5S_23)$$

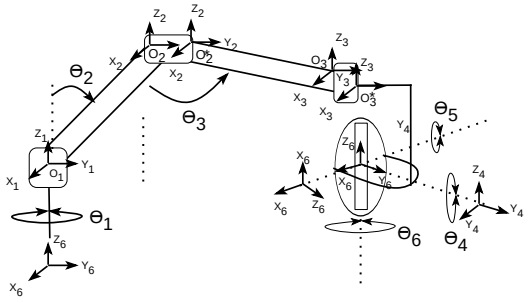
$$Y_y = -S_6(S_1C_4 + C_1C_4C_{23}) + C_6(-S_1S_4C_5 + C_1C_4C_5C_23) - C_1S_5S_23)$$

$$Y_z = -S_6(S_{23}C_4) + C_6(C_4C_5S_{23} + S_5C_{23})$$

$$Z_x = -S_5(-C_1S_4 - S_1C_4C_5C_{23}) + S_1C_5S_23)$$



(a) The kinematic model of the original 6 dof master arm



(b) The kinematic model of the modified 6 dof master arm

Figure 3: The original (a) and modified (b) kinematic models of the master arm

$$Z_y = -S_5(-S_1S_4 + C_1C_4C_5C_23) - C_1C_5S_23)$$
$$Z_z = -S_5(C_4S_{23} + C_5C_{23})$$

The end effector position vector is given by:

$$O_0O_{6,0} = O_0O_{5,0} + M_0^6O_5O_{6,6} \tag{49}$$

$$O_0O_{6,0} = O_0O_{5,0} + M_0^6.\begin{pmatrix} 0 \\ 0 \\ \ell_6 \end{pmatrix} \tag{50}$$

$$O_0O_{6,0} = \begin{pmatrix} X_4 + (l_5 + l_6).Z_x \\ Y_4 + (l_5 + l_6).Z_y \\ Z_4 + (l_5 + l_6).Z_z \end{pmatrix} \tag{51}$$

The final position of the end effector is given by :

$$X = X_4 + (l_5 + l_6).Z_x$$
$$Y = Y_4 + (l_5 + l_6).Z_y$$
$$Z = Z_4 + (l_5 + l_6).Z_z$$

6.1.3 The geometric model for the passive master arm

Our original 6 dof articulated system that can be used as a passive master arm is shown in Figure 3-(a). This system is very useful for testing (logical and mathematical) our telerobotic system before we receive our commercial master arm. It has six

degrees of freedom with all revolute joints. All the joints are equipped with potentiometers (sensors) which are used to measure the angles between joints. In addition to these sensors, the holder at the end effector part of the master arm is attached with a Stop/Start switch and mode selector switches. The stop/start button is used to enable/disable the master arm, and hence can be used by the operator to disable the system temporarily, move the master arm to a comfortable location, and then re-enable the system. The mode selector switch is used to select the mode of operation. The geometric model allows mapping trajectories of the hand effector that are described in the joint space, into the corresponding trajectories in the Cartesian space. The position and the orientation of the effector can be totally determined by means of the following information:

1. The hand center or vector $O_nO_{n,0}$ which references the origin of R_n with respect to R_0:
$$O_nO_{n,0} = \sum_{i=1}^{n} O_{i-1,0}O_{i,0} \tag{52}$$

2. The hand orientation matrix $M_0^n = [X_{n,0}, Y_{n,0}, Z_{n,0}]$ determines the orientation of frame R_n with respect to frame R_0.

The frame representation of the master arm is shown in Figure 1-(b) and (c) and the following topological form is used to describe the geometric structure of the master arm :

$$Link1(ROTZ(\theta_1), Z(L1)); L_1 = 100mm$$

$$Link2(ROTX(\theta_2), Z(L2)); L_2 = 360mm$$
$$Link3(ROTX(\theta_3), Y(L3)); L_3 = 350mm$$
$$Link4(ROTY(\theta_4), Y(L4)); L_4 = 20mm$$
$$Link5(ROTX(\theta_5), Y(L5)); L_5 = 50mm$$
$$Link6(ROTY(\theta_6), Y(L6)); L_6 = 20mm$$

The link body L_1 is along the Z_1 vector and rotate around the Z axis.
For the first end point O_1 in the master arm we have:

$$O_0O_{1,0} = M_0^1.O_0O_{1,1} \tag{53}$$

M_0^1 is the rotation matrix between R_0 and R_1 frames, and is given by:

$$M_0^1 = ROTZ(\theta_1) \tag{54}$$

$$M_o^1 = \begin{pmatrix} C_1 & -S_1 & 0 \\ S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{55}$$

The position vector is given by:

$$O_0O_{1,0} = M_0^1.O_0O_{1,1} \tag{56}$$

where $O_0O_{1,1}$ is simply expressed as:

$$O_0O_{1,1} = \begin{pmatrix} 0 \\ 0 \\ \ell_1 \end{pmatrix} \tag{57}$$

Therefore, position vector $O_0O_{1,0}$ is given by:

$$O_0O_{1,0} = \begin{pmatrix} C_1 & -S_1 & 0 \\ S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 0 \\ 0 \\ \ell_1 \end{pmatrix} \tag{58}$$

$$O_0O_{1,0} = \begin{pmatrix} 0 \\ 0 \\ \ell_1 \end{pmatrix} \tag{59}$$

The link body L_2 is along vector Z_2 of frame R_2 and rotate around X_1

$$M_0^2 = M_0^1.M_1^2 = ROTZ(\theta_1).ROTX(\theta_2) \tag{60}$$

$$M_0^2 = \begin{pmatrix} C_1 & -S_1 & 0 \\ S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 1 & 0 & 0 \\ 0 & C_2 & -S_2 \\ 0 & S_2 & C_2 \end{pmatrix} \tag{61}$$

$$M_0^2 = \begin{pmatrix} C_1 & -S_1C_2 & S_1S_2 \\ S_1 & C_1C_2 & -C_1S_2 \\ 0 & S_2 & C_2 \end{pmatrix} \tag{62}$$

The position vector $O_2O_{2,0}$ is given by:

$$O_0O_{2,0} = O_0O_{1,0} + M_0^2. \begin{pmatrix} 0 \\ 0 \\ \ell_2 \end{pmatrix} = \begin{pmatrix} S_1S_2\ell_2 \\ -C_1S_2\ell_2 \\ \ell_1 + C_2\ell_2 \end{pmatrix} \tag{63}$$

We found that the link L_3 drifts in the Y and Z directions from the origin O_2 as shown in Figure 3-(a). These drifts are measured as:

$$y_a = 45mm$$

$$z_a = 12.5mm$$

The orientation of the mechanical piece holding O_2 and O_2^* is fixed, the modified position vector $O_2O_{2,0}^*$ is given by:

$$O_0O_{2,0}^* = O_0O_{2,0} + M_0^2O_2O_{2,2}^* \tag{64}$$

where

$$O_2O_{2,2}^* = \begin{pmatrix} 0 \\ y_a \\ z_a \end{pmatrix} \tag{65}$$

$$O_0O_{2,0}^* = \begin{pmatrix} X_2^* \\ Y_2^* \\ Z_2^* \end{pmatrix} = \begin{pmatrix} S_1S_2(\ell_2 + z_a) - S_1C_2y_0 \\ -C_1S_2(\ell_2 + z_a) + C_1C_2y_0 \\ \ell_1 + C_2(\ell_2 + z_a) + S_2y_0 \end{pmatrix} \tag{66}$$

The link body L_3 is along vector Y_3 and rotate around X axis.
Frame R_3 depends only on θ_l and θ_3. Therefore, The orientation matrix M_0^3 is given by:

$$M_0^3 = M_0^1.M_2^3 = ROTZ(\theta_1).ROTX(\theta_3) \tag{67}$$

$$M_0^3 = \begin{pmatrix} C_1 & -S_1C_3 & S_1S_3 \\ S_1 & C_1C_3 & -C_1S_3 \\ 0 & S_3 & C_3 \end{pmatrix} \tag{68}$$

The position vector $O_0O_{3,0}$ is given by:

$$O_0O_{3,0} = O_0O_{2,0}^* + M_0^3O_2O_{3,3} \tag{69}$$

$$O_0O_{3,0} = O_0O_{2,0}^* + M_0^3.\begin{pmatrix} 0 \\ \ell_3 \\ 0 \end{pmatrix} \tag{70}$$

The link L_4 drifts in the Y and Z directions from the origin O_3 as shown in Figure 3-(a), which are measured as:

$$y_b = 40mm$$

$$z_b = 22.5mm$$

The modified position vector $O_0O_{3,0}^*$ is expressed as follow:

$$O_0O_{3,0}^* = O_0O_{2,0}^* + M_0^3.\begin{pmatrix} 0 \\ \ell_3 + y_b \\ z_b \end{pmatrix} \tag{71}$$

$$O_0O_{3,0}^* = \begin{pmatrix} X_3^* \\ Y_3^* \\ Z_3^* \end{pmatrix} = \begin{pmatrix} X_2^* - S_1C_3(\ell_3 + y_b) + S_1S_3z_1 \\ Y_2^* + C_1C_3(\ell_3 + y_b) - C_1S_3z_1 \\ Z_2^* + S_3(\ell_3 + y_b) + C_3z_1 \end{pmatrix} \tag{72}$$

The link body L_4 is along vector Y_4 and rotate around Y axis.
The orientation matrix M_0^4 depends only on θ_1 and θ_4:

$$M_0^4 = M_0^1.M_3^4 = ROTZ(\theta_1).R0TY(\theta_4) \tag{73}$$

$$M_0^4 = \begin{pmatrix} C_1C_4 & -S_1 & C_1S_4 \\ S_1C_4 & C_1 & S_1S_4 \\ -S_4 & 0 & C_4 \end{pmatrix} \tag{74}$$

for the position vector $O_0O_{4,0}$ we have:

$$O_0O_{4,0} = O_0O_{3,0}^* + M_0^4O_3O_{4,4}^* \tag{75}$$

$$O_0O_{4,0} = O_0O_{3,0}^* + M_0^4.\begin{pmatrix} 0 \\ \ell_4 \\ 0 \end{pmatrix} \tag{76}$$

$$O_0O_{4,0} = \begin{pmatrix} X_4 \\ Y_4 \\ Z_4 \end{pmatrix} = \begin{pmatrix} X_3^* - S_1\ell_4 \\ Y_3^* + C_1\ell_4 \\ Z_3^* \end{pmatrix} \tag{77}$$

The link body L_5 is along vector Y_5 and rotate around X axis.
The orientation matrix M_0^5 is given by:

$$M_0^5 = M_0^4.M_4^5 = M_0^4.ROTX(\theta_5) \tag{78}$$

$$M_0^5 = \begin{pmatrix} C_1C_4 & -S_1 & C_1S_4 \\ S_1C_4 & C_1 & S_1S_4 \\ -S_4 & 0 & C_4 \end{pmatrix} . \begin{pmatrix} 1 & 0 & 0 \\ 0 & C5 & -S5 \\ 0 & S5 & C5 \end{pmatrix} \tag{79}$$

$$M_0^5 = \begin{pmatrix} C_1C_4 & -S_1C_5 + C_1S_4S_5 & S_1S_5 + C_1S_4C_5 \\ S_1C_4 & C_1C_5 + S_1S_4S_5 & -C_1S_5 + S_1S_4C_5 \\ -S_4 & C_4S_5 & C_4C_5 \end{pmatrix} \tag{80}$$

The position vector $O_0O_{5,0}$ is computed as following:

$$O_0O_{5,0} = O_0O_{4,0} + M_0^5O_4O_{5,5} \tag{81}$$

$$O_0O_{5,0} = O_0O_{4,0} + M_0^5 \begin{pmatrix} 0 \\ \ell_5 \\ 0 \end{pmatrix} \tag{82}$$

$$O_0O_{5,0} = O_0O_{4,0} + M_0^5 \begin{pmatrix} 0 \\ \ell_5 \\ 0 \end{pmatrix} \tag{83}$$

$$O_0O_{5,0} = \begin{pmatrix} X_4 + (-S_1C_5 + C_1S_4S_5).\ell_5 \\ Y_4 + (C_1C_5 + S_1S_4S_5)\ell_5 \\ Z_4 + C_4S_5\ell_5 \end{pmatrix} \tag{84}$$

The link body L_6 is along vector Y_6 and rotate around Y axis.
The final orientation matrix M_0^6 is given by:

$$M_0^6 = M_0^5.M_5^6 = M_0^5.ROTY(\theta_6) \tag{85}$$

$$M_0^6 = M_0^5. \begin{pmatrix} C_6 & 0 & S_6 \\ 0 & 1 & 0 \\ -S_6 & 1 & C_6 \end{pmatrix} \tag{86}$$

The final expression for the orientation matrix is given by:

$$M_0^6 = \begin{pmatrix} X_x & Y_x & Z_x \\ X_y & Y_y & Z_y \\ X_z & Y_z & Z_z \end{pmatrix} \tag{87}$$

The components of the orientation matrix are given below:

$$X_x = C_1C_4C_6 - S_6(S_1S_5 + C_1S_4C_5)$$

$$X_y = S_1C_4C_6 - S_6(-C_1S_5 + C_1S_4C_5)$$

$$X_Z = -S_4C_6 - S_6(C_4C_5)$$

$$Y_x = -S_1C_5 + C_1S_4S_5$$

$$Y_y = C_1C_5 + S_1S_4S_5$$

$$Y_z = C_4 S_5$$
$$Z_x = C_1 C_4 S_6 + C_6(S_1 S_5 + C_1 S_4 C_5)$$
$$Z_y = S_1 C_4 S_6 + C_6(-C_1 S_5 + C_1 S_4 C_5)$$
$$Z_z = -S_4 S_6 + C_6(C_4 C_5)$$

The end effector position vector is given by:

$$O_0 O_{6,0} = O_0 O_{5,0} + M_0^6 O_5 O_{6,6} \tag{88}$$

$$O_0 O_{6,0} = O_0 O_{5,0} + M_0^6 . \begin{pmatrix} 0 \\ \ell_6 \\ 0 \end{pmatrix} \tag{89}$$

$$O_0 O_{6,0} = \begin{pmatrix} X_4 + Y_x . \ell_6 \\ Y_4 + Y_y . \ell_6 \\ Z_4 + Y_z . \ell_6 \end{pmatrix} \tag{90}$$

The $O_0 O_{6,0}$ and M_0^6 components of the position vector of the end effector for the master arm are given below:

$$X = X_4 + Y_x . \ell_6$$
$$Y = Y_4 + Y_y . \ell_6$$
$$Z = Z_4 + Y_z . \ell_6$$

The Geometric model for the new structure

We have developed the structure of the master arm to improve the operator performance and to overcome the problems of the previous structure. The motivation is to provide the operator with a master arm tool that provide an effective way of mapping his own hand frame of reference with the PUMA hand frame.

You may refer to Section 6.1.6 to see more details about this subject. In this Section we will discuss the changes in the geometric model for the new structure of the master arm. The transporter part of the master arm is the same as in Section 6.1.3. The new structure of the master arm is shown in Figure 3-(b).

To obtain the geometric model for the new structure, we have to express vectors $O_0 O_{6,0}$, X_6, Y_6 and Z_6 with respect to the absolute coordinates system R_0. Since the transporter part remains the same, we can use the position vector $O_0 O^*_{3,0}$ and the orientation matrix M_0^3 which have been computed and discussed in this chapter. The orientation matrix M_0^3 is given by:

$$M_0^3 = \begin{pmatrix} C_1 & -S_1 C_3 & S_1 S_3 \\ S_1 & C_1 C_3 & -C_1 S_3 \\ 0 & S_3 & C_3 \end{pmatrix} \tag{91}$$

and the position vector $O_0 O^*_{3,0}$ is given by:

$$O_0 O^*_{3,0} = O_0 O^*_{2,0} + M_0^3 . \begin{pmatrix} 0 \\ \ell_3 + y_b \\ z_b \end{pmatrix} \tag{92}$$

$$O_0O^*_{3,0} = \begin{pmatrix} X^*_3 \\ Y^*_3 \\ Z^*_3 \end{pmatrix} = \begin{pmatrix} X^*_2 - S_1C_3(\ell_3 + y_b) + S_1S_3z_1 \\ Y^*_2 + C_1C_3(\ell_3 + y_b) - C_1S_3z_1 \\ Z^*_2 + S_3(\ell_3 + y_b) + C_3z_1 \end{pmatrix} \tag{93}$$

The operator hand is at the origin of the concurrent rotations and below the frame R^*_3 by small drifts in -Z and y directions as shown in Figure 3-(b), which are measured as $z_c = -150mm$ and $y_c = 60mm$.

So we compute the position vector of the end effector as following:

$$O_0O_{6,0} = O_0O^*_{3,0} + M^1_0O_3O_{6,3} \tag{94}$$

where

$$O_3O_{6,3} = \begin{pmatrix} 0 \\ y_c \\ z_c \end{pmatrix} \tag{95}$$

and

$$O_0O^*_{3,0} = \begin{pmatrix} X^*_3 \\ Y^*_3 \\ Z^*_3 \end{pmatrix} \tag{96}$$

The X, Y, and Z coordinates of the position vector of the end effector are:

$$X = X^*_3 - S_1.y_c$$

$$Y = Y^*_3 + C_1.y_c$$

$$Z = Z^*_3 + z_c$$

Now we will proceed to find the orientation matrix of the system. Because of the new structure of the master arm the frame R^*_4 rotated relative to frame R_3 about X by 180 degrees and about Z by 180 degrees as shown in Figure 3-(b) where $\{X^*_4, Y^*_4, Z^*_4\}$ $= \{X_3, -Y_3, -Z_3\}$. We use the following expression for the $M^{4^*}_0$ orientation matrix:

$$M^{4^*}_0 = M^3_0. \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \tag{97}$$

$$M^{4^*}_0 = \begin{pmatrix} C_1 & S_1 & 0 \\ S_1 & -C_1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \tag{98}$$

The orientation matrix M^4_0 is given by:

$$M^4_0 = M^{4^*}_0.ROTY(\theta_4) \tag{99}$$

$$M^4_0 = \begin{pmatrix} C_1C_4 & S_1 & C_1S_4 \\ S_1C_4 & -C_1 & S_1S_4 \\ S_4 & 0 & -C_4 \end{pmatrix} \tag{100}$$

The orientation matrix M^5_0 is given by:

$$M^5_0 = M^4_0.M^5_4 = M^4_0.ROTX(\theta_5) \tag{101}$$

$$M_0^5 = \begin{pmatrix} C_1C_4 & S_1 & C_1S_4 \\ S_1C_4 & -C_1 & S_1S_4 \\ S_4 & 0 & -C_4 \end{pmatrix} . \begin{pmatrix} 1 & 0 & 0 \\ 0 & C5 & -S5 \\ 0 & S5 & C5 \end{pmatrix} \tag{102}$$

$$M_0^5 = \begin{pmatrix} C_1C_4 & S_1C_5 + C_1S_4S_5 & -S_1S_5 + C_1S_4C_5 \\ S_1C_4 & -C_1C_5 + S_1S_4S_5 & C_1S_5 + S_1S_4C_5 \\ -S_4 & -C_4S_5 & -C_4C_5 \end{pmatrix} \tag{103}$$

The final orientation matrix M_0^6 is given by:

$$M_0^6 = M_0^5.M_5^6 = M_0^5.ROTZ(\theta_6) \tag{104}$$

$$M_0^6 = M_0^5. \begin{pmatrix} C_6 & -S_6 & 0 \\ S_6 & C_6 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{105}$$

The final expression for the orientation matrix is given by:

$$M_0^6 = \begin{pmatrix} X_x & Y_x & Z_x \\ X_y & Y_y & Z_y \\ X_z & Y_z & Z_z \end{pmatrix} \tag{106}$$

the components of this matrix are given below:

$$X_x = C_1C_4C_6 + S_6(S_1C_5 + C_1S_4S_5)$$

$$X_y = S_1C_4C_6 + S_6(-C_1C_5 + S_1S_4S_5)$$

$$X_Z = S_4C_6 - S_6(C_4S_5)$$

$$Y_x = -C_1C_4S_6 + C_6(S_1C_5 + C_1S_4S_5)$$

$$Y_y = -S_1C_4S_6 - C_6(C_1C_5 - C_1S_4S_5)$$

$$Y_z = -S_4S_6 - C_6(C_4S_5)$$

$$Z_x = -S_1S_5 + C_1S_4C_5$$

$$Z_y = C_1C_5 + S_1S_4C_5$$

$$Z_z = -C_4C_5$$

In conclusion we use the position vector $O_0O_{6,0}$ and the orientation matrix M_0^6 to geometrically represent the robot and the master arms in the three dimensional space. The direct geometric model is used to convert a point in the joint space to a position and orientation in the Cartesian space. We have developed a computer program that directly reads the joint angles of both master arm and robot arm and generates the direct geometric solution. The master arm has six potentiometers for joint angles measurements, while the joint angles of the robot arm are determined using incremental optical encoders and potentiometers. The potentiometers that are used in the master arm are normal type available in the market, in which their accuracy is acceptable, but they can be replaced by a better type to improve the accuracy of angle readings. The replacement of the old master arm with the new one does not add any major changes in the geometric model. On the other hand, it improves the operation and performance of the human operator.

6.1.4 The inverse geometric model for the PUMA slave arm

The PUMA slave arm and its frame of references are shown on shown on Figure 2. The geometric system for the transporter part is defined by:

$$(\theta_1, \theta_2, \theta_3) \rightarrow \left\{ O_0O_4(\theta), M_0^4(\theta) \right\}$$

The inverse geometric model of the transporter consists of finding closed form solutions for θ_1, θ_2, and θ_3 as functions of the transporter end point coordinates X, Y, and Z. Evaluation of the geometric model of the transporter allows writing the coordinate of vector $O_0O_{4,0}(\theta)$ as follows:

$$O_0O_{4,0} = \begin{pmatrix} C_1.D + S_1(S_2\ell_2 + S_{23}\ell_{34}) \\ S_1.D - C_1(S_2\ell_2 + S_{23}\ell_{34}) \\ \ell_1 + C_2\ell_2 + C_{23}\ell_{34} \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{107}$$

To solve the system we first consider the expressions of X and Y in order to evaluate $(S2L_2 + S23L_{34})$, we can easily obtain:

$$S2L_2 + S23L_{34} = \pm\sqrt{X^2 + Y^2 - d^2} \tag{108}$$

When the point O_3 is not on the Z_0 axis, i.e., $X^2 + Y^2 - d^2 \neq 0$, then the sine and cosine of θ_1 can be evaluated as follows:

$$S1 = \frac{XD \pm X\sqrt{X^2 + Y^2 - D^2}}{\sqrt{X^2 + Y^2}} \ \ and \ \ C1 = \frac{XD \mp Y\sqrt{X^2 + Y^2 - D^2}}{\sqrt{X^2 + Y^2}} \tag{109}$$

Depending on the sign, we have two solutions for the angle θ_1 . When the sign $(+)$ is selected, we obtain the following solution:

$$S1^+ = \frac{XD + X\sqrt{X^2 + Y^2 - D^2}}{\sqrt{X^2 + Y^2}} \quad and \ \ C1^+ = \frac{XD - Y\sqrt{X^2 + Y^2 - D^2}}{\sqrt{X^2 + Y^2}} \tag{110}$$

Note that the knowledge of both sine and cosine of an angle allows finding a unique solution for that angle. The other solution will be obtained by inverting the sign. Consequently two solutions are expected for the angle θ_1:

$$\begin{aligned} (S1^+, C1^+) &\rightarrow \theta_1^+ \\ (S1^-, C1^-) &\rightarrow \theta_1^- \end{aligned} \tag{111}$$

The solution θ_1^+ can be evaluated as follows:

$$\theta_1^+ = TAN^{-1}(S1^+, C1^+)$$

When the robot is following a trajectory in the cartesian space, the previous solution that corresponds to the previous point of the trajectory can be used in order to select one solution out of two. Therefore, to determine a unique solution θ one may compare θ_1^+ and θ_1^- to the previous value of θ . Clearly, the closest solution to the previous one allows satisfying a continuity criteria on the cartesian trajectory. In addition to finding a solution θ_1, this operation allows finding the sign of $S1$, $C1$, and $S2L_2 + S23L_3$. To determine θ_3, we consider the previous expressions of X, Y,

and Z after the following change $X = X - DC1$, $Y = Y - DS1$, and Z is unchanged. Since θ_1 is known then the new X and Y are also known. We have:

$$\begin{aligned} X^2 + Y^2 &= (S2L_2 + S23L_{34})^2 \\ (Z - L_1)^2 &= (C2L_2 + C23L_{34})^2 \end{aligned} \tag{112}$$

After developing the above relations, we obtain:

$$\begin{aligned} X^2 + Y^2 &= (S2L_2)^2 + (S23L_{34})^2 + 2S2S23L_2L_{34} \\ (Z - L_1)^2 &= (C2L_2)^2 + (C23L_{34})^2 + 2C2C23L_2L_{34} \end{aligned} \tag{113}$$

and adding:

$$X^2 + Y^2 + (Z - L_1)^2 = L_2^2 + L_{34}^2 + 2C3L_2L_{34} \tag{114}$$

We obtain $C(\theta_3)$ and $S(\theta_3)$:

$$\begin{aligned} C3 = (X^2 + Y^2 + (Z - L_1)^2 - L_2^2 - -L_{34}^2)/2L_2L_{34} \\ S3 = \pm\sqrt{1 - C3^2} \end{aligned} \tag{115}$$

Two symmetric solutions for θ_3 are expected. Naturally these solutions correspond to two different configurations but both allows the transporter end point being set at the coordinates specified by X, Y, and Z. Obviously, this arm can reach all the position of its work space by specifying the angle θ_3 in one of the intervals $[0, +\Pi]$ and $[-\Pi, 0]$. Depending on which interval is selected, the sign of $S3$ can then be determined. A criteria on space occupancy of the arm can be used in order to chose one of these intervals. Once the term $S3$ is sign of θ_3 is found, angle θ_3 can then be evaluated as follows:

$$\theta_3 = TAN^{-1}(S3, C3)$$

Finally, to determine angle θ_2, we consider the following equations:

$$\begin{aligned} S1X - C1Y &= S2L_2 + S23L_{34} \\ Z - L_1 &= C2L_2 + C23L_{34} \end{aligned} \tag{116}$$

After developing $S23$ and $C23$, we can write these equations in a matrix form:

$$\begin{bmatrix} S1X - C1Y \\ Z - L_1 \end{bmatrix} = \begin{bmatrix} S3L_{34} & L_2 + C3L_{34} \\ L_2 + C3L_{34} & -S3L_{34} \end{bmatrix} . \begin{bmatrix} C2 \\ S2 \end{bmatrix} \tag{117}$$

The determinant of this matrix is given by:

$$\Delta = (S3L_{34})^2 + (L_2 + C3L_{34})^2 = -(L_2^2 + L_{34}^2 + 2L_2L_{34}C3) \tag{118}$$

In general, Δ is not nil except when $L_2 = L_{34}$ and θ_{34} is equal $\pm\Pi$. This configuration of θ_{34} cannot occur in a mechanical robot arm. The solution $C2$ and $S2$ can always be obtained as follows:

$$\begin{aligned} S2 &= \frac{(XS1 - YC1)(L_2 + C3L_{34}) - (Z - L_1)S3L_{34}}{L_2^2 + L_{34}^2 + 2L_2L_{34}C3} \\ C2 &= \frac{(XS1 - YC1)S3L_{34} + (Z - L_1)(L_2 + C3L_{34})}{L_2^2 + L_{34}^2 + 2L_2L_{34}C3} \end{aligned} \tag{119}$$

The solution for the angle θ_2 can then be obtained as follows:

$$\theta_2 = TAN^{-1}(S2, C2)$$

This solution depends on the selected values of θ_1, θ_3, and their respective signs.

Now we address the problem of having multiple solutions and singularities. Let us consider the transporter defined in Section 2.3. When only considering the links L_2 and L_{34}, two solution are generally expected when the transporter end point is set to any position specified by X, Y, and Z. Mathematically, this is because $S3$ cannot be determined by using the system $O_0O_3, 0(\theta) = (X, Y, Z)^t$. On the other hand, the mechanical structure of this arm indicates clearly that two configurations for (θ_2, θ_3) exist while the end point O_3 is fixed. These configurations are:

$$\{\theta_1, \theta_3^+, \theta_2(\theta_1, \theta_3^+)\} \ \ and \ \ \{\theta_1, \theta_3^+, \theta_2(\theta_1, \theta_3^-)\} \tag{120}$$

To make decision about which solution should be kept, one needs to assign use one of the following methods:

1. Use of a continuity criteria on the cartesian trajectory so that decision will be made by comparing the solutions θ_3^+ and θ_3^- to the previous solution. This method allows maintaining the sign of angle θ_3 fixed during the motion of the arm. To initialize the motion, the starting configuration should implicitly include this information about the selected sign of θ_3.

2. Use of a flag to indicate the current value of the sign of θ_3 . In this case, no comparison will be made but rather the sign of θ_3 will be selected according to the value of the flag which should be appropriately initialized by the system. This solution can be augmented with a function that allows switching the value of the flag and generation of smooth motion between the two configurations.

Regarding the angle θ_1, we always have two solutions θ_1^+ and θ_1^- for each of which there exists two possible configuration for angles θ_2 and θ_3 . Figure 2.5 shows that the arm admits four solutions when solving the system $O_0O_{3,0}(\theta) = (X, Y, Z)^t$:

$$\theta_1^- = \theta_1 - sign(\theta_1^+).\pi \tag{121}$$

Because θ_1 is in $[-\pi, +\pi]$, the solutions are:

$$\left[\theta_1^+, \theta_3^+, \theta_2(\theta_1^+, \theta_3^+)\right]$$

$$\left[\theta_1^+, \theta_3^-, \theta_2(\theta_1^+, \theta_3^-)\right]$$

$$\left[\theta_1^-, \theta_3^+, \theta_2(\theta_1^-, \theta_3^+)\right]$$

$$\left[\theta_1^-, \theta_3^-, \theta_2(\theta_1^-, \theta_3^-)\right]$$

The selection of one solution for angle θ_1 is easier because the values of θ_1^+ and θ_1^- always differ by Π . Therefore, one needs to compare with the previous value of θ_1

in order to select either θ_1^+ or θ_1^- . As angle θ_1 is generally defined in $[-\Pi, +\Pi]$, no pre-selection can be made with respect to the solutions θ_1^+ and θ_1^-. Let us consider the case when ($X = 0$ and $Y = 0$), i.e., the end point 0_3 is on the Z_1 axis as shown in Figure 2.6. The system equation becomes:

$$\begin{aligned} X &= S1(S2L_2 + S23L_{34}) = 0 \\ Y &= -C1(S2L_2 + S23L_{34}) = 0 \\ Z &= L_1 + C2L_2 + C23L_{34} \end{aligned} \tag{122}$$

As $S1$ and $C1$ cannot be equal to zero simultaneously, then the term $S2L_2 + S23L_{34}$ is nil. The system equation becomes:

$$\begin{aligned} S2L_2 + S23L_{34} &= 0 \\ C2L_2 + C23L_{34} &= Z - L_1 \end{aligned} \tag{123}$$

After developing the terms $S23$ and $C23$, the solution for θ_2 can then be obtained as follows:

$$\begin{aligned} C2 &= \frac{(z - L_1)(L_2 + C3L_{34})}{L_2^2 + L_3^2 + 2C3L_2L_3} \\ S2 &= -\frac{S3L_3(Z - L_1)}{L_2^2 + L_3^2 + 2C3L_2L_3} \end{aligned} \tag{124}$$

On the other hand, the solution for θ_3 is obtained by using the precedent equation of $C3$:

$$\begin{aligned} C3 &= \frac{(Z - L_1)^2 - L_2^2 - L_3^2}{2L_2L_3} \\ S3 &= \pm\sqrt{1 - C3^2} \end{aligned} \tag{125}$$

Suppose we determine the solution for θ_2 and θ_3 as previously discussed, the remaining angle θ_1 is undetermined because no information is available about this angle. The system equation $O_0O_{3,0}(\theta) = (X, Y, Z)^t$ does not give any information regarding angle θ_1 when $X = 0$ and $Y = 0$. This situation is called a singularity case for θ_1 because there exists an infinite number of values for θ_1 for which the system's equations (2.38) is satisfied. One may keep angle θ_1 unchanged until the arm moves to a new point in which the condition ($X = 0$ and $Y = 0$) is no more satisfied. Another method consists of using the criteria on trajectory continuity which can be helpful in this case. The application of this criteria consists of extrapolating the trajectory of angle $\theta_1(t)$ by using time polynomial approximation. A discrete time polynomial $\theta_1(nT)$ can be used to approximate the time function $\theta_1(t)$ over a finite number of points $\theta_1(n-1), ..., \theta_1(n-k)$. The values of $\theta_1(n-i)$ represent the ith previous solution of the system equations. Using k previous solutions, the associated polynomial can be used to predict the current value of θ_1:

$$\theta_1(n) = F\left[\theta_1(n-1), ..., \theta_1(n-k)\right]$$

If one assumes constant sampling period of the system, a second order polynomial approximation can then give the following solution:

$$\theta_1(n) = 3\theta(n-1) - 3\theta(n-2) + \theta(n-3) \tag{126}$$

Now we evaluate the inverse geometrical transform for the effector part that consists of finding the joint variables $\theta_4, ..., \theta 6$ given the hand position and orientation matrix:

$$\underbrace{O_0O_{6,0}}_{\text{Hand Center}} \text{ and } \underbrace{M_0^6 = \{X_6, Y_6, Z_6\}}_{\text{Hand Orientation Matrix}} \tag{127}$$

This system equations consists of twelve nonlinear, redundant equations with respect to $\theta_1, ..., \theta 6$. The first three equations concern the three cartesian coordinates of the hand center $O_0O_{6,0}$. These equations may be written such that the unknown terms appear on the right-hand of the equal sign:

$$\begin{aligned} X_6 - L5.Z_{x6} &= X_4 = S1(S2L_2 + S23(L_3 + L4)) \\ Y_6 - L5.Z_{y6} &= Y_4 = -C1(S2L_2 + S23(L_3 + L4)) \\ Z_6 - L5.Z_{z6} &= Z_4 = L_1 + C2L_2 + C23(L_3 + L4) \end{aligned} \tag{128}$$

Now, for the point O_5 we have:

$$O_0O_{5,0} = O_0O_{4,0} + M_0^5.O_4O_{5,5} \tag{129}$$

Link L_5 is a revolute link about X axis. Therefore, M_0^5 is given by:

$$M_0^5 = M_0^4.M_4^5 = M_0^4. \begin{bmatrix} 1 & 0 & 0 \\ 0 & C5 & -S5 \\ 0 & S5 & C5 \end{bmatrix} = \{X_5, Y_5, Z_5\} \tag{130}$$

$$X_5 = \begin{bmatrix} C1C4 - S1C23S4 \\ S1C4 + C1C23S4 \\ S23S4 \end{bmatrix} \tag{131}$$

$$Y_5 = \begin{bmatrix} -(C1S4 + S1C23C4)C5 + S1S23S5 \\ -(S1S4 - C1C23C4)C5 - C1S23S5 \\ -S23C4C5 + C23S5 \end{bmatrix} \tag{132}$$

$$Z_5 = \begin{bmatrix} (C1S4 + S1C23C4)S5 + S1S23C5 \\ -(C1C23C4 - S1S4)S5 - C1S23C5 \\ -S23C4S5 + C23C5 \end{bmatrix} \tag{133}$$

Because Link $L6$ is a revolute link about Z axis we have:

$$Z_{6,0} = Z_5, 0 \tag{134}$$

Since Vector $O_0O_{5,0}$ may be obtained simply as follows:

$$O_0O_{5,0} = O_0O_{4,0} + M_0^5.O_4O_{6,6} \tag{135}$$

Particularly we have:

$$O_0O_{6,0} = O_0O_{4,0} + (L5 + L6).Z_{6,0} \tag{136}$$

The cartesian coordinate of the end Effector is:

$$O_0O_{6,0} = \begin{bmatrix} S1(S2L_2 + S23(L_3 + L_4)) + L5.Z_{x6} \\ -C1(S2L_2 + S23(L_3 + L_4)) + L5.Z_{y6} \\ L_1 + C2L_2 + C23(L_3 + L_4)) + L5.Z_{z6} \end{bmatrix} \tag{137}$$

Finally we compute the basic orientation matrix M_0^6 using M_0^5 and the rotation property of joint L_6, we have:

$$M_0^6 = M_0^4.M_5^6 = \{X_6, Y_6, Z_6\} = \begin{bmatrix} X_x & X_y & X_z \\ X_y & Y_y & Z_y \\ X_z & Y_z & Z_z \end{bmatrix} \tag{138}$$

For each component of this matrix we obtain:

$$\begin{aligned}
X_x &= C1C4C6 - S1C23S4C6 - C1S4C5S6 - S1C23C4C5S6 \\
&\quad +S1S23S5S6 && (139)\\
X_y &= S1C4C6 + C1C23S4C6 - S1S4C5S6 - C1C23C4C5S6 \\
&\quad -C1S23S5S6 && (140)\\
X_z &= S23S4C6 + S23C4C5S6 + C23S5S6 && (141)\\
Y_x &= C1C4S6 + S1C23S4S6 - C1S4C5C6 - S1C23C4C5C6 \\
&\quad +S1S23S5C6 && (142)\\
Y_y &= -S1C4S6 + C1C23S4S6 - S1S4C5C6 + S1C23C4C5C6 \\
&\quad -C1S23S5C6 && (143)\\
Y_z &= S23S4S6 + S23C4C5C6 + C23S5C6 && (144)\\
Z_x &= C1S4S5 + S1C23S4C5 + S1S23C5 && (145)\\
Z_y &= S1S4S5 - C1C23C4C5 - C1S23C5 && (146)\\
Z_z &= -S23C4S5C23C5 && (147)
\end{aligned}$$

Based on the above vector O_0O_5 can be expressed as follows:

$$O_0O_{4,0} = O_0O_{6,0} - L5.Z_{6,0} \tag{148}$$

Equations of X_4, Y_4 and Z_4 are similar to the three revolutes transporter developed in Section 2.4. The solutions for θ_1, θ_2 and θ_3 have the same form as in Section 2.3.

The second step is to determine solutions for $\theta 4, \theta 5$ and $\theta 6$. For that we observe that the rotation axes of links L_4 and L_5 are orthogonal. Since the terms $C_4, 4\ S_4, C_5$ and S_5 should appear in the rotation matrix M_3^5:

$$M_3^5 = M_3^4.M_4^5 = ROTZ(\theta 4).ROTX(\theta 5) \tag{149}$$

$$M_3^5 = \begin{bmatrix} C4 & -S4 & 0 \\ S4 & C4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & C5 & -S5 \\ 0 & S5 & C5 \end{bmatrix} = \begin{bmatrix} C4 & -S4C5 & S4S5 \\ S4 & C4C5 & -C4S5 \\ 0 & S5 & C5 \end{bmatrix} \tag{150}$$

The rotation matrix of the robot hand is known and is given by:

$$M_0^6 = M_0^3.M_3^5.M_5^6 = \{X_6, Y_6, Z_6\} \tag{151}$$

In addition, the last three rotation axes are concurrent, and we have:

$$M_6^5 = \begin{bmatrix} C6 & S6 & 0 \\ -S6 & C6 & 0 \\ 0 & 01 & \end{bmatrix} \tag{152}$$

To identify C_4, S_4, C_5 and S_5 we may use the following equations:

$$M_3^5 = (M_0^3)^{-1}.M_0^6.(M_3^5)^{-1} = M_3^0.M_0^6.M_6^5 \tag{153}$$

The system L_1, L_2, and L_3 has been solved and we suppose a solution $(\theta_1, \theta_2, \theta_3)$ is found. Since the matrix $M_3^0.M_0^6$ is known. As $\theta 6$ is defined as a rotation about $Z5$ axis, then $Z6$ will not be affected by $\theta 6$. For this, we start by expressing the product $M_3^0.M_0^6$. For M_3^0 we have:

$$M_3^0 = \left[M_0^3\right]^t = [ROTZ(\theta_1).ROTX(\theta_2).ROTX(\theta_3)]^t \tag{154}$$

Since the rotation axes of θ_2 and θ_3 are parallel:

$$\begin{aligned} M_0^3 &= ROTZ(\theta_1).ROTX(\theta_2 + \theta_3) \\ M_3^0 &= \begin{bmatrix} C1 & S1 & 0 \\ -S1C23 & C1C23 & S23 \\ S1S23 & -C1S23 & C23 \end{bmatrix} \end{aligned} \tag{155}$$

Any M_0^6 is given by:

$$M_0^6 = \begin{bmatrix} X_x & Y_x & Z_x \\ X_y & Y_y & Z_y \\ X_z & Y_z & Z_z \end{bmatrix} \tag{156}$$

Since we have:

$$M_3^0.M_0^6 = (V1V2V3)$$

$$V1 = \begin{bmatrix} C1X_x + S1X_y \\ -S1C23X_x + C1C23X_y + S23X_z \\ S1S23X_x - C1S23X_y + C23X_z \end{bmatrix}$$

$$V2 = \begin{bmatrix} C1Y_x + S1Y_y \\ -S1C23Y_x + C1C23Y_y + S23Y_z \\ S1S23Y_x - C1S23Y_y + C23Y_z \end{bmatrix}$$

$$V3 = \begin{bmatrix} C1Z_x + S1Z_y \\ -S1C23Z_x + C1C23Z_y + S23Z_Z \\ S1S23Z_X - C1S23Z_Y + C23Z_Z \end{bmatrix}$$

As $\theta 6$ is a rotation about Z axis, then the third column (or vector) of the produce $M_3^0.M_0^6$ will not be affected by $\theta 6$. Recall the relation (3-2) and (3-3) we can express the third column of $M_3^5(3-1)$ **independently from** $\theta 6$:

$$S4S5 = C1Z_x + S1Zy \tag{157}$$

$$C4S5 = C23(S1Z_x - C1Z_y) - S23Z_z \tag{158}$$

$$C5 = S23(S1Z_x - C1Z_y) + C23Z_z \tag{159}$$

We note that it is out of interest to express $S5$, $C4$, and $S4$ from the same Equations 2.116 because of their dependency of the unknown angle $\theta 6$. Certainly a similar equations relating $\theta 6$ to $\theta 5$ can be obtained while $\theta 4$ is unknown. As this is completely equivalent to the system 2.116, then it is of no interest. If we examine the mechanical system, it will clearly indicate the impossibility to uniquely determine

solution for $\theta 4$ and $\theta 6$ independently from $\theta 5$. In particular, when $\theta 5 = 0$ the hand orientation will only depend on $\theta 4 + \theta 6$. Since $\theta 4$ and $\theta 6$ cannot be determined independently from $\theta 5$. In fact, the mathematical system 2.106 does not give more information than the system 2.116. Using the system equations 2.116 we have:

$$\begin{aligned} C5 &= C23(S1Z_x - C1Z_y) - C23Z_z \\ S5 &= \pm\sqrt{1 - C5^2} \end{aligned} \tag{160}$$

Two symmetrical solutions are then possible for angle $\theta 5$ within $[-\pi, +\pi]$: Angle $\theta 4$ can be determined from the system (3-4) only when $S \neq 0$ (see next pages), we have:

$$\begin{aligned} S4 &= \frac{Z_x C1 + Z_y S1}{S5} \\ C4 &= \frac{C23(S1Z_x - C1Z_y) - S23Z_z}{S5} \end{aligned} \tag{161}$$

As $C4$ and $S4$ depend on the sign of $S5$, then two solutions are expected within $[-\pi, +\pi]$:

$$\{C4^+, S4^+\} \text{ and } \{C4^- = -C4^+ \ \ and \ \ S4^- = -S4^+\} \tag{162}$$

The solutions $\theta 4^+$ and $\theta 4^-$ differ by π . In the domain $[-\pi, +\pi]$ we have: $\theta 4^+(C4^+, S4^+)$

$$\theta 4^- = \theta 4^+ - sign(\theta 4^+).\pi \tag{163}$$

To determine $\theta 6$ when $\theta 5 \neq 0$ we assume the value of $C4, S4, C5$ and $S5$ have been computed according to equations 2.117 and 2.118. Let us express the rotation matrix M_5^6, we have:

$$M_5^6 = M_5^3.M_3^0.M_0^6 \tag{164}$$

Matrices $M_5^3(\theta 4, \theta 5)$, $M_3^0(\theta_1, \theta_2, \theta_3)$, and M_0^6 are given. The matrix M_5^6 is a $ROTZ(\theta 6)$:

$$M_5^6 = \begin{bmatrix} C6 & -S6 & 0 \\ S6 & C6 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{165}$$

Recall the matrices $M_3^0.M_0^6(3-3)$ and $M_3^5(3-1)$ that have been previously evaluated, we have:

$$M_5^3 = \begin{bmatrix} C4 & S4 & 0 \\ -S4C5 & C4C5 & S5 \\ S4S5 & -C4S5 & C5 \end{bmatrix} \tag{166}$$

Expression of $C6$ and $S6$ are then obtained from the product $M_5^3.(M_3^0.M_0^6)$ as follows:

$$\begin{aligned} C6 &= C4(C1X_x + S1X_y) + S4(-S1C23X_x + C1C23X_y + S23X_z) \\ S6 &= C4(C1Y_x + S1Y_y) + S4(-S1C23Y_x + C1C23Y_y + S23Y_z) \end{aligned} \tag{167}$$

Depending on the sign of $S5$, i.e. $C4$ and $S4$, we determine two solution for $\theta 6$:

$$\begin{aligned} S5^+ &= +\sqrt{1 - C5^2} \rightarrow (C4^+, S4^+) \rightarrow (C6^+, S6^+) \\ \text{And } S5^- &= -S5^+ \rightarrow (C4^-, S4^-) \rightarrow (C6^-, S6^-) \end{aligned} \tag{168}$$

Since, two solutions are also expected for $\theta 6$:

$$\theta 6^{+}(C6^{+}, S6^{+})$$

$$\theta 6^{-} = \theta 6^{+} - Sign(\theta 6^{+}).\pi \tag{169}$$

Figure 2.13 shows how these solution can be obtained: consider one initial solution (Figure A) and consider the operations $\theta_5 \leftarrow -\theta_5$ (Figure B), $\theta_4 \leftarrow -\theta_4 + \Pi$ (Figure C), and $\theta_6 \leftarrow -\theta_6 + \Pi$ (Figure D). Obviously, this lead to obtain another possible solution for the position and the orientation of the arm.

Conclusion on the case $\theta 5 \neq 0$

Two set of solutions are expected:

$$(\theta 5^{+}, \theta 4^{+}, \theta 6^{+}) \text{ and } (\theta 5^{+}, \theta 4^{+}, \theta 6^{+})$$

where

$$\begin{array}{lll} \theta 5^{+} & \text{and} & \theta 5^{-} \text{are symmetrical} \\ \theta 4^{+} & \text{and} & \theta 4^{-} \text{Differ by} \pi \\ \theta 6^{+} & \text{and} & \theta 6^{-} \text{Differ by} \pi \end{array} \tag{170}$$

By trajectory continuity, we may identify a solution. For example, by comparing $(\theta 4^{+}, \theta 4^{-})$ to the previous value of $\theta 4$ (Initial):
Example:

$$\begin{aligned} \text{If } |\theta 4_I - \theta 4^{+}| &> |\theta 4_I - \theta 4^{-}| \ THEN & \\ \theta 4 &= \theta 4^{+}; \theta 5 = \theta^{+}; \theta 6 = \theta 6^{+} & (171) \\ ELSE & & (172) \\ \theta 4 &= \theta 4^{-}; \theta 5^{+}; \theta 6 = -\theta 6^{+} & (173) \\ END & & (174) \end{aligned}$$

Given the hand center $O_0O_{6,0}$ and the orientation matrix M_0^6, it is possible to find at least two configurations for the effector part $(\theta 4, \theta 5, \theta 6)$:

$$(\theta 5, \theta 4^{+}, \theta 6^{+}) \text{ and } (-\theta 5, \theta 4^{-}, \theta 6^{-}) \tag{175}$$

Case where $\theta 5 = 0$
This case corresponds to a singular configuration as shown on Figure 2.14.In this situation, the rotation axes of $\theta 4$ and $\theta 6$ are co-linear and concurrent, we have:

$$M_3^6 = M_3^4.M_4^5.M_5^6 = M_3^4.M_5^6 = ROTZ(\theta 4^{\theta} 6) \tag{176}$$

And

$$M_3^6 = \begin{bmatrix} C46 & -S46 & 0 \\ C46 & C46 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{177}$$

In order to express C46 and S46, we may use the following equation:

$$M_3^6 = M_3^0.M_0^6 \tag{178}$$

The product $M_3^0.M_0^6$ has been previously expressed, we have:

$$\begin{aligned} C46 &= C1X_x + S1X_y \\ S46 &= -(C1Y_X + S1Y_Y) \end{aligned} \tag{179}$$

Since $\theta 4$ and $\theta 6$ cannot be expressed independently from each other. In fact the mechanical analysis confirms this mathematical result, and then heuristics can be applied to obtain a possible alternative. For example we may keep unchanged one angle, i.e. $\theta 4$ or $\theta 6$, and determine the other angle from their sum $(\theta 4 + \theta 6)$ as it is identified using $C46$ and $S46$. Another alternative consists of estimating one angle, i.e. $\theta 4$ or $\theta 6$, by extrapolating its time function according to the previous values. Assume the previous values of $\theta 4$ are:

$$\theta 4(t-k), ..., \theta 4(t-1), \theta 4(t) \tag{180}$$

And

$$\theta 6(t-k), ..., \theta 6(t-1), \theta 6(t) \tag{181}$$

A polynomial approximation with degree K is given by:

$$\theta(t^1) = \sum_{i=0}^{K} a_i \theta(t-i) \tag{182}$$

Higher accuracy is obtained for lower degree polynomial. Since we may select to extrapolate either $\theta 4$ or $\theta 6$, according to the lowest polynomial degree. Assume $\theta 4$ is obtained by extrapolation, then $\theta 6$ could now be simply obtained as $\theta 6 = \theta 46 - \theta 4$.

6.1.5 Kinematics of the designed and manufactured master arm

The designed and manufactured master arm has six revolute links. The operator holds the end effector at the center of effector frame of reference that can be determined by means of the following information:

- The operator hand orientation matrix $M_0^6 = [X_6^0, Y_6^0, Z_6^0]$, determines the orientation of frame R_6 with respect to frame R_0.
- The position vector $O_0O_{6,0}$, references the origin of R_6 with respect to R_0.

The orientation matrix M_0^i is the products of the rotation matrices:

$$M_0^6 = M_0^1.M_1^2.M_2^3...M_{5-1}^6 \tag{183}$$

The operator hand position vector can be determined as follows:

$$O_0O_{6,0} = \sum_{i=1}^{6} M_0^i.O_{i-1}O_{i,i} \tag{184}$$

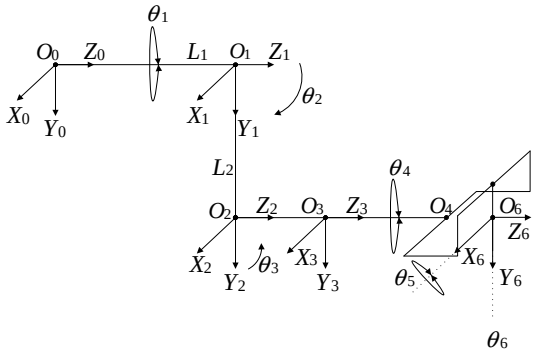


Figure 4: Geometric model of the manufactured Master Arm

Direct kinematics of the manufactured master arm

The reference arm position required to build the geometric model for the master arm is shown in Figure 4. We have used the following topological form to describe the geometric structure of the arm:
The operator hand position is defined by the first three revolute joints:

$$Linkl(ROTZ(\theta_1), Z(L1))$$

$$Link2(ROTX(\theta_2), Z(L2))$$

$$Link3(ROTX(\theta_3), Z(L3))$$

The operator hand orientation is function of first 3 dof and last three concurrent revolute joints:

$$Link4(ROTZ(\theta_4), Z(L4))$$

$$Link5(ROTX(\theta_5), Z(L5))$$

$$Link6(ROTZ(\theta_6), Z(L6))$$

where $ROTZ(\theta_i)$ is the rotation of link i between frames R_i and R_{i-1} about the Z_{i-1} axis, and $Z(L_i)$ indicates that the link body L_i is along the Z_i vector. Consider L_1 which is revolute and defined as a rotation about the Z_o axis, and the link body L_1 is along vector Z_1 of frame R_1.
We start by computing the transformation matrix between frames R_1 and R_0

$$M_o^1 = ROTZ(\theta_1) \tag{185}$$

$$M_o^1 = \begin{pmatrix} C_1 & -S_1 & 0 \\ S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{186}$$

The solution to the position vector $[O_0O_{1,0}]$ yields:

$$O_0O_{1,0} = M_0^1.O_0O_{1,1} \tag{187}$$

$$O_0O_{1,0} = \begin{pmatrix} C_1 & -S_1 & 0 \\ S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 0 \\ \ell_1 \end{pmatrix} \tag{188}$$

$$O_0O_{1,0} = \begin{pmatrix} 0 \\ 0 \\ \ell_1 \end{pmatrix} \tag{189}$$

For the second end point O_2 we have:

$$M_0^2 = M_0^1.M_1^2 = ROTZ(\theta_1).ROTX(\theta_2) \tag{190}$$

$$M_0^2 = \begin{pmatrix} C_1 & -S_1 & 0 \\ S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 1 & 0 & 0 \\ 0 & C_2 & -S_2 \\ 0 & S_2 & C_2 \end{pmatrix} \tag{191}$$

$$M_0^2 = \begin{pmatrix} C_1 & -S_1C_2 & S_1S_2 \\ S_1 & C_1C_2 & -C_1S_2 \\ 0 & S_2 & C_2 \end{pmatrix} \tag{192}$$

The position vector $O_2O_{2,0}$ is given by:

$$O_0O_{2,0} = O_0O_{1,0} + M_0^2O_1O_{2,2} \tag{193}$$

The link body L_2 is along vector Z_2 of frame R_2.

$$O_0O_{2,0} = O_0O_{1,0} + M_0^2. \begin{pmatrix} 0 \\ 0 \\ \ell_2 \end{pmatrix} \tag{194}$$

$$O_0O_{2,0} = \begin{pmatrix} S_1S_2\ell_2 \\ -C_1S_2\ell_2 \\ \ell_1 + C_2\ell_2 \end{pmatrix} \tag{195}$$

The link body L_3 is along vector Z_3 and rotates around X axis.
The orientation matrix M_0^3 is given by:

$$M_0^3 = M_0^2.M_2^3 = M_0^2.ROTX(\theta_3) \tag{196}$$

$$M_0^2 = \begin{pmatrix} C_1 & -S_1C_2 & S_1S_2 \\ S_1 & C_1C_2 & -C_1S_2 \\ 0 & S_2 & C_2 \end{pmatrix} \tag{197}$$

$$M_0^3 = \begin{pmatrix} C_1 & -S_1C_2 & S_1S_2 \\ S_1 & C_1C_2 & -C_1S_2 \\ 0 & S2 & C2 \end{pmatrix} . \begin{pmatrix} 1 & 0 & 0 \\ 0 & C_3 & -S_3 \\ 0 & S_3 & C_3 \end{pmatrix} \tag{198}$$

Given that :
$C_{23} = C_2C_3 - S_2S_3$
$S_{23} = C_2S_3 + C_3S_2$
Then

$$M_0^3 = \begin{pmatrix} C_1 & -S_1C_23 & S_1S_23 \\ S_1 & C_1C_23 & -C_1S_23 \\ 0 & S23 & C23 \end{pmatrix} \tag{199}$$

The position vector $O_0O_{3,0}$ is given by:

$$O_0O_{3,0} = O_0O_{2,0} + M_0^3 O_2O_{3,3} \tag{200}$$

Then

$$O_0O_{3,0} = O_0O_{2,0} + M_0^3 . \begin{pmatrix} 0 \\ 0 \\ \ell_3 \end{pmatrix} \tag{201}$$

$$O_0O_{3,0} = \begin{pmatrix} S_1S_2\ell_2 \\ -C_1S_2\ell_2 \\ \ell_1 + C_2\ell_2 \end{pmatrix} + \begin{pmatrix} C_1 & -S_1C_{23} & S_1S_{23} \\ S_1 & C_1C_{23} & -C_1S_{23} \\ 0 & S_{23} & C_{23} \end{pmatrix} . \begin{pmatrix} 0 \\ 0 \\ \ell_3 \end{pmatrix} \tag{202}$$

$$O_0O_{3,0} = \begin{pmatrix} S_1S_2\ell_2 + S_1S_{23}\ell_3 \\ C_1S_2\ell_2 + C_1S_{23}\ell_3 \\ \ell_1 + C_2\ell_2 + C_{23}\ell_3 \end{pmatrix} \tag{203}$$

The link body L_4, L_5, and L_6 are along vector Z_4, Z_5, and Z_6, respectively. For the orientation matrix we have :

$$M_0^4 = M_0^3 . M_3^4 = M_0^3 . ROTZ(\theta_4) \tag{204}$$

$$M_0^4 = \begin{pmatrix} C_1 & -S_1C_{23} & S_1S_{23} \\ S_1 & C_1C_{23} & -C_1S_{23} \\ 0 & S_{23} & C_{23} \end{pmatrix} . \begin{pmatrix} C_4 & -S_4 & 0 \\ S_4 & C_4 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{205}$$

$$M_0^4 = \begin{pmatrix} C_1C_4 - S_1S_4C_{23} & -C_1S_4 - S_1C_4C_{23} & S_1S_{23} \\ S_1C_4 + C_1C_4C_{23} & -S_1S_4 + C_1C_4C_{23} & -C_1S_{23} \\ S_{23}C_4 & C_4S_{23} & C_{23} \end{pmatrix} \tag{206}$$

For the position vector $O_0O_{4,0}$ we have:

$$O_0O_{4,0} = O_0O_{3,0} + M_0^4 O_3O_{4,4} \tag{207}$$

$$O_0O_{4,0} = O_0O_{3,0} + M_0^4 . \begin{pmatrix} 0 \\ 0 \\ \ell_4 \end{pmatrix} \tag{208}$$

The third link of master arm has the sum of links from link 3, 4, 5, and 6. We denote this by ℓ_{3456}. Therefore we have:

$$O_0O_{4,0} = O_0O_{5,0} = O_0O_{6,0} = \begin{pmatrix} S_1(S_2\ell_2 + S_{23}\ell_{3456}) \\ C_1(S_2\ell_2 - S_{23}\ell_{3456}) \\ \ell_1 + C_2\ell_2 + C_{23}\ell_{3456} \end{pmatrix} \tag{209}$$

Vector $O_0O_{6,0}$ defined above gives the position vector of the operator hand.

Now we compute the operator hand orientation matrix which is M_0^6 because the operator holds the arm at $O_0O_{6,0}$. Since $M_0^6 = M_0^1 M_1^2 M_2^3 M_3^4 M_4^5 M_5^6$ are defined by:

$$M_0^6 = \begin{pmatrix} X_x & Y_x & Z_x \\ X_y & Y_y & Z_y \\ X_z & Y_z & Z_z \end{pmatrix}$$

The final components of the orientation matrix M_0^6 are the orthogonal vectors which can be easily obtained:

$$X_x = C_6(C_1C_4 - S_1S_4C_{23}) + S_6(-C_1C_5S_4 - S_1C_4C_5C_{23}) + S_1S_5S_23)$$

$$X_y = C_6(S_1C_4 + C_1C_4C_{23}) + S_6(-S_1S_4C_5 + C_1C_4C_5C_23) - C_1S_5S_23)$$

$$X_z = C_6(S_{23}C_4) + S_6(C_4C_5S_{23} + S_5C_{23})$$

$$Y_x = -S_6(C_1C_4 - S_1S_4C_{23}) + C_6(-C_1C_5S_4 - S_1C_4C_5C_{23}) + S_1S_5S_23)$$

$$Y_y = -S_6(S_1C_4 + C_1C_4C_{23}) + C_6(-S_1S_4C_5 + C_1C_4C_5C_23) - C_1S_5S_23)$$

$$Y_z = -S_6(S_{23}C_4) + C_6(C_4C_5S_{23} + S_5C_{23})$$

$$Z_x = -S_5(-C_1S_4 - S_1C_4C_5C_{23}) + S_1C_5S_23)$$

$$Z_y = -S_5(-S_1S_4 + C_1C_4C_5C_23) - C_1C_5S_23)$$

$$Z_z = -S_5(C_4S_{23} + C_5C_{23})$$

The inverse kinematics of the manufactured master arm

The master arm and its frame of references are shown on Figure 2. The geometric model is defined by:

$$(\theta_1, \theta_2, ...\theta_6) \rightarrow \left\{O_0O_6(\theta), M_0^6(\theta)\right\}$$

The inverse geometric model of the master arm consists of finding closed form solutions for θ_1, θ_2, ..., θ_6 as functions of the operator hand position $O_0O_{6,0}$ and orientation matrix M_0^6.

The operator hand position vector $O_0O_{6,0}(\theta)$ is: follows:

$$O_0O_{4,0} = \begin{pmatrix} S_1(S_2\ell_2 + S_{23}\ell_{34}) \\ C_1(S_2\ell_2 + S_{23}\ell_{34}) \\ \ell_1 + C_2\ell_2 + C_{23}\ell_{34} \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{211}$$

To solve the system we first consider the expressions of X and Y in order to evaluate $(S2L_2 + S23L_{34})$, we can easily obtain:

$$S2L_2 + S23L_{34} = \pm\sqrt{X^2 + Y^2} \tag{212}$$

When the point O_3 is not on the Z_0 axis, i.e., $X^2 + Y^2 - d^2 \neq 0$, then the sine and cosine of θ_1 can be evaluated as follows:

$$S1 = \frac{\pm X}{\sqrt{X^2 + Y^2}} \ \ and \ \ C1 = \frac{\mp Y}{\sqrt{X^2 + Y^2}} \tag{213}$$

Depending on the sign, we have two solutions for the angle θ_1 . When the sign $(+)$ is selected, we obtain the following solution:

$$S1^+ = \frac{X}{\sqrt{X^2 + Y^2}} \quad and \ \ C1^+ = \frac{-Y}{\sqrt{X^2 + Y^2}} \tag{214}$$

Note that the knowledge of both sine and cosine of an angle allows finding a unique solution for that angle. The other solution will be obtained by inverting the sign. Consequently two solutions are expected for the angle θ_1:

$$\begin{aligned}(S1^+, C1^+) &\rightarrow \theta_1^+ \\ (S1^-, C1^-) &\rightarrow \theta_1^-\end{aligned} \tag{215}$$

The solution θ_1^+ can be evaluated as follows:

$$\theta_1^+ = TAN^{-1}(S1^+, C1^+)$$

When the arm is following a trajectory in the cartesian space, the previous solution that corresponds to the previous point of the trajectory can be used in order to select one solution out of two. Therefore, to determine a unique solution θ one may compare θ_1^+ and θ_1^- to the previous value of θ . Clearly, the closest solution to the previous one allows satisfying a continuity criteria on the cartesian trajectory. In addition to finding a solution θ_1, this operation allows finding the sign of $S1$, $C1$, and $S2L_2 + S23L_3$. Since θ_1 is known then the new X and Y are also known. We have:

$$\begin{aligned}X^2 + Y^2 &= (S2L_2 + S23L_{34})^2 \\ (Z - L_1)^2 &= (C2L_2 + C23L_{34})^2\end{aligned} \tag{216}$$

After developing the above relations, we obtain:

$$\begin{aligned}X^2 + Y^2 &= (S2L_2)^2 + (S23L_{34})^2 + 2S2S23L_2L_{34} \\ (Z - L_1)^2 &= (C2L_2)^2 + (C23L_{34})^2 + 2C2C23L_2L_{34}\end{aligned} \tag{217}$$

and adding:

$$X^2 + Y^2 + (Z - L_1)^2 = L_2^2 + L_{34}^2 + 2C3L_2L_{34} \tag{218}$$

We obtain $C(\theta_3)$ and $S(\theta_3)$:

$$\begin{aligned}C3 = (X^2 + Y^2 + (Z - L_1)^2 - L_2^2 - -L_{34}^2)/2L_2L_{34} \\ S3 = \pm\sqrt{1 - C3^2}\end{aligned} \tag{219}$$

Two symmetric solutions for θ_3 are expected. Naturally these solutions correspond to two different configurations but both allows the transporter end point being set at the coordinates specified by X, Y, and Z. Obviously, this arm can reach all the position of its work space by specifying the angle θ_3 in one of the intervals $[0, +\Pi]$ and $[-\Pi, 0]$. Depending on which interval is selected, the sign of $S3$ can then be determined. A criteria on space occupancy of the arm can be used in order to chose one of these intervals. Once the term $S3$ is sign of θ_3 is found, angle θ_3 can then be evaluated as follows:

$$\theta_3 = TAN^{-1}(S3, C3)$$

Finally, to determine angle θ_2, we consider the following equations:

$$\begin{aligned}S1X - C1Y &= S2L_2 + S23L_{34} \\ Z - L_1 &= C2L_2 + C23L_{34}\end{aligned} \tag{220}$$

After developing $S23$ and $C23$, we can write these equations in a matrix form:

$$\begin{bmatrix} S1X - C1Y \\ Z - L_1 \end{bmatrix} = \begin{bmatrix} S3L_{34} & L_2 + C3L_{34} \\ L_2 + C3L_{34} & -S3L_{34} \end{bmatrix} . \begin{bmatrix} C2 \\ S2 \end{bmatrix} \tag{221}$$

The determinant of this matrix is given by:

$$\Delta = (S3L_{34})^2 + (L_2 + C3L_{34})^2 = -(L_2^2 + L_{34}^2 + 2L_2L_{34}C3) \tag{222}$$

In general, Δ is not nil except when $L_2 = L_{34}$ and θ_{34} is equal $\pm\Pi$. This configuration of θ_{34} cannot occur in a mechanical robot arm. The solution $C2$ and $S2$ can always be obtained as follows:

$$\begin{aligned} S2 &= \frac{(XS1 - YC1)(L_2 + C3L_{34}) - (Z - L_1)S3L_{34}}{L_2^2 + L_{34}^2 + 2L_2L_{34}C3} \\ C2 &= \frac{(XS1 - YC1)S3L_{34} + (Z - L_1)(L_2 + C3L_{34})}{L_2^2 + L_{34}^2 + 2L_2L_{34}C3} \end{aligned} \tag{223}$$

The solution for the angle θ_2 can then be obtained as follows:

$$\theta_2 = TAN^{-1}(S2, C2)$$

This solution depends on the selected values of θ_1, θ_3, and their respective signs.

Now we address the problem of having multiple solutions and singularities. Let us consider the transporter defined in Section 2.3. When only considering the links L_2 and L_{34}, two solution are generally expected when the transporter end point is set to any position specified by X, Y, and Z. Mathematically, this is because $S3$ cannot be determined by using the system $O_0O_3, 0(\theta) = (X, Y, Z)^t$. On the other hand, the mechanical structure of this arm indicates clearly that two configurations for (θ_2, θ_3) exist while the end point O_3 is fixed. These configurations are:

$$\{\theta_1, \theta_3^+, \theta_2(\theta_1, \theta_3^+)\} \ \textit{and} \ \{\theta_1, \theta_3^+, \theta_2(\theta_1, \theta_3^-)\} \tag{224}$$

To make decision about which solution should be kept, one needs to assign use one of the following methods:

1. Use of a continuity criteria on the cartesian trajectory so that decision will be made by comparing the solutions θ_3^+ and θ_3^- to the previous solution. This method allows maintaining the sign of angle θ_3 fixed during the motion of the arm. To initialize the motion, the starting configuration should implicitly include this information about the selected sign of θ_3.

2. Use of a flag to indicate the current value of the sign of θ_3 . In this case, no comparison will be made but rather the sign of θ_3 will be selected according to the value of the flag which should be appropriately initialized by the system. This solution can be augmented with a function that allows switching the value of the flag and generation of smooth motion between the two configurations.

Regarding the angle θ_1, we always have two solutions θ_1^+ and θ_1^- for each of which there exists two possible configuration for angles θ_2 and θ_3 . Figure 2.5 shows that the arm admits four solutions when solving the system $O_0O_{3,0}(\theta) = (X, Y, Z)^t$:

$$\theta_1^- = \theta_1 - sign(\theta_1^+).\pi \tag{225}$$

Because θ_1 is in $[-\pi, +\pi]$, the solutions are:

$$\left[\theta_1^+, \theta_3^+, \theta_2(\theta_1^+, \theta_3^+)\right]$$

$$\left[\theta_1^+, \theta_3^-, \theta_2(\theta_1^+, \theta_3^-)\right]$$

$$\left[\theta_1^-, \theta_3^+, \theta_2(\theta_1^-, \theta_3^+)\right]$$

$$\left[\theta_1^-, \theta_3^-, \theta_2(\theta_1^-, \theta_3^-)\right]$$

The selection of one solution for angle θ_1 is easier because the values of θ_1^+ and θ_1^- always differ by Π . Therefore, one needs to compare with the previous value of θ_1 in order to select either θ_1^+ or θ_1^- . As angle θ_1 is generally defined in $[-\Pi, +\Pi]$, no pre-selection can be made with respect to the solutions θ_1^+ and θ_1^-. Let us consider the case when ($X = 0$ and $Y = 0$), i.e., the end point 0_3 is on the Z_1 axis as shown in Figure 2.6. The system equation becomes:

$$\begin{aligned} X &= S1(S2L_2 + S23L_{34}) = 0 \\ Y &= -C1(S2L_2 + S23L_{34}) = 0 \\ Z &= L_1 + C2L_2 + C23L_{34} \end{aligned} \tag{226}$$

As $S1$ and $C1$ cannot be equal to zero simultaneously, then the term $S2L_2 + S23L_{34}$ is nil. The system equation becomes:

$$\begin{aligned} S2L_2 + S23L_{34} &= 0 \\ C2L_2 + C23L_{34} &= Z - L_1 \end{aligned} \tag{227}$$

After developing the terms $S23$ and $C23$, the solution for θ_2 can then be obtained as follows:

$$\begin{aligned} C2 &= \frac{(z - L_1)(L_2 + C3L_{34})}{L_2^2 + L_3^2 + 2C3L_2L_3} \\ S2 &= -\frac{S3L_3(Z - L_1)}{L_2^2 + L_3^2 + 2C3L_2L_3} \end{aligned} \tag{228}$$

On the other hand, the solution for θ_3 is obtained by using the precedent equation of $C3$:

$$\begin{aligned} C3 &= \frac{(Z - L_1)^2 - L_2^2 - L_3^2}{2L_2L_3} \\ S3 &= \pm\sqrt{1 - C3^2} \end{aligned} \tag{229}$$

Suppose we determine the solution for θ_2 and θ_3 as previously discussed, the remaining angle θ_1 is undetermined because no information is available about this angle. The system equation $O_0O_{3,0}(\theta) = (X, Y, Z)^t$ does not give any information

regarding angle θ_1 when $X = 0$ and $Y = 0$. This situation is called a singularity case for θ_1 because there exists an infinite number of values for θ_1 for which the system's equations (2.38) is satisfied. One may keep angle θ_1 unchanged until the arm moves to a new point in which the condition ($X = 0$ and $Y = 0$) is no more satisfied. Another method consists of using the criteria on trajectory continuity which can be helpful in this case. The application of this criteria consists of extrapolating the trajectory of angle $\theta_1(t)$ by using time polynomial approximation. A discrete time polynomial $\theta_1(nT)$ can be used to approximate the time function $\theta_1(t)$ over a finite number of points $\theta_1(n-1), ..., \theta_1(n-k)$. The values of $\theta_1(n-i)$ represent the ith previous solution of the system equations. Using k previous solutions, the associated polynomial can be used to predict the current value of θ_1:

$$\theta_1(n) = F\left[\theta_1(n-1), ..., \theta_1(n-k)\right]$$

If one assumes constant sampling period of the system, a second order polynomial approximation can then give the following solution:

$$\theta_1(n) = 3\theta(n-1) - 3\theta(n-2) + \theta(n-3) \tag{230}$$

Now we evaluate the inverse geometrical transform for the effector part that consists of finding the joint variables $\theta_4, ..., \theta 6$ given the hand position and orientation matrix:

$$\frac{O_0O_{6,0}}{\text{Hand Center}} \text{ and } \frac{M_0^6 = \{X_6, Y_6, Z_6\}}{\text{Hand Orientation Matrix}} \tag{231}$$

This system equations consists of twelve nonlinear, redundant equations with respect to $\theta_1, ..., \theta 6$. The first three equations concern the three cartesian coordinates of the hand center $O_0O_{6,0}$. These equations may be written such that the unknown terms appear on the right-hand of the equal sign:

$$\begin{aligned} X_6 &= X_4 = S1(S2L_2 + S23(L_3 + L4)) \\ Y_6 &= Y_4 = -C1(S2L_2 + S23(L_3 + L4)) \\ Z_6 &= Z_4 = L_1 + C2L_2 + C23(L_3 + L4) \end{aligned} \tag{232}$$

Finally we compute the basic orientation matrix M_0^6 using M_0^5 and the rotation property of joint L_6, we have:

$$M_0^6 = M_0^4.M_5^6 = \{X_6, Y_6, Z_6\} = \begin{bmatrix} X_x & X_y & X_z \\ X_y & Y_y & Z_y \\ X_z & Y_z & Z_z \end{bmatrix} \tag{233}$$

For each component of this matrix we obtain:

$$\begin{aligned} X_x &= C1C4C6 - S1C23S4C6 - C1S4C5S6 - S1C23C4C5S6 \\ &\quad +S1S23S5S6 \qquad (234) \\ X_y &= S1C4C6 + C1C23S4C6 - S1S4C5S6 - C1C23C4C5S6 \\ &\quad -C1S23S5S6 \qquad (235) \\ X_z &= S23S4C6 + S23C4C5S6 + C23S5S6 \qquad (236) \\ Y_x &= C1C4S6 + S1C23S4S6 - C1S4C5C6 - S1C23C4C5C6 \end{aligned}$$

$$+S1S23S5C6 \tag{237}$$

$$\begin{aligned} Y_y &= -S1C4S6 + C1C23S4S6 - S1S4C5C6 + S1C23C4C5C6 \\ &\quad -C1S23S5C6 \end{aligned} \tag{238}$$

$$Y_z = S23S4S6 + S23C4C5C6 + C23S5C6 \tag{239}$$

$$Z_x = C1S4S5 + S1C23S4C5 + S1S23C5 \tag{240}$$

$$Z_y = S1S4S5 - C1C23C4C5 - C1S23C5 \tag{241}$$

$$Z_z = -S23C4S5C23C5 \tag{242}$$

Based on the above vector O_0O_5 can be expressed as follows:

$$O_0O_{4,0} = O_0O_{6,0} - L5.Z_{6,0} \tag{243}$$

We need to determine solutions for $\theta 4, \theta 5$ and $\theta 6$. For that we observe that the rotation axes of links L_4 and L_5 are orthogonal. Since the terms $C_4, 4\ S_4, C_5$ and S_5 should appear in the rotation matrix M_3^5:

$$M_3^5 = M_3^4.M_4^5 = ROTZ(\theta 4).ROTX(\theta 5) \tag{244}$$

$$M_3^5 = \begin{bmatrix} C4 & -S4 & 0 \\ S4 & C4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & C5 & -S5 \\ 0 & S5 & C5 \end{bmatrix} = \begin{bmatrix} C4 & -S4C5 & S4S5 \\ S4 & C4C5 & -C4S5 \\ 0 & S5 & C5 \end{bmatrix} \tag{245}$$

The rotation matrix of the robot hand is known and is given by:

$$M_0^6 = M_0^3.M_3^5.M_5^6 = \{X_6, Y_6, Z_6\} \tag{246}$$

In addition, the last three rotation axes are concurrent, and we have:

$$M_6^5 = \begin{bmatrix} C6 & S6 & 0 \\ -S6 & C6 & 0 \\ 0 & 01 & \end{bmatrix} \tag{247}$$

To identify C_4, S_4, C_5 and S_5 we may use the following equations:

$$M_3^5 = (M_0^3)^{-1}.M_0^6.(M_3^5)^{-1} = M_3^0.M_0^6.M_6^5 \tag{248}$$

The system L_1, L_2, and L_3 has been solved and we suppose a solution $(\theta_1, \theta_2, \theta_3)$ is found. Since the matrix $M_3^0.M_0^6$ is known. As $\theta 6$ is defined as a rotation about $Z5$ axis, then $Z6$ will not be affected by $\theta 6$. For this, we start by expressing the product $M_3^0.M_0^6$. For M_3^0 we have:

$$M_3^0 = \left[M_0^3\right]^t = [ROTZ(\theta_1).ROTX(\theta_2).ROTX(\theta_3)]^t \tag{249}$$

Since the rotation axes of θ_2 and θ_3 are parallel:

$$\begin{aligned} M_0^3 &= ROTZ(\theta_1).ROTX(\theta_2 + \theta_3) \\ M_3^0 &= \begin{bmatrix} C1 & S1 & 0 \\ -S1C23 & C1C23 & S23 \\ S1S23 & -C1S23 & C23 \end{bmatrix} \end{aligned} \tag{250}$$

Any M_0^6 is given by:

$$M_0^6 = \begin{bmatrix} X_x & Y_x & Z_x \\ X_y & Y_y & Z_y \\ X_z & Y_z & Z_z \end{bmatrix} \tag{251}$$

Since we have:

$$M_3^0.M_0^6 = (V1V2V3)$$

$$V1 = \begin{bmatrix} C1X_x + S1X_y \\ -S1C23X_x + C1C23X_y + S23X_z \\ S1S23X_x - C1S23X_y + C23X_z \end{bmatrix}$$

$$V2 = \begin{bmatrix} C1Y_x + S1Y_y \\ -S1C23Y_x + C1C23Y_y + S23Y_z \\ S1S23Y_x - C1S23Y_y + C23Y_z \end{bmatrix}$$

$$V3 = \begin{bmatrix} C1Z_x + S1Z_y \\ -S1C23Z_x + C1C23Z_y + S23Z_Z \\ S1S23Z_X - C1S23Z_Y + C23Z_Z \end{bmatrix}$$

As $\theta 6$ is a rotation about Z axis, then the third column (or vector) of the produce $M_3^0.M_0^6$ will not be affected by $\theta 6$. Recall the relation (3-2) and (3-3) we can express the third column of $M_3^5(3-1)$ **independently from** $\theta 6$:

$$S4S5 = C1Z_x + S1Zy \tag{252}$$

$$C4S5 = C23(S1Z_x - C1Z_y) - S23Z_z \tag{253}$$

$$C5 = S23(S1Z_x - C1Z_y) + C23Z_z \tag{254}$$

We note that it is out of interest to express $S5$, $C4$, and $S4$ from the same Equations 2.116 because of their dependency of the unknown angle $\theta 6$. Certainly a similar equations relating $\theta 6$ to $\theta 5$ can be obtained while $\theta 4$ is unknown. As this is completely equivalent to the system 2.116, then it is of no interest. If we examine the mechanical system, it will clearly indicate the impossibility to uniquely determine solution for $\theta 4$ and $\theta 6$ independently from $\theta 5$. In particular, when $\theta 5 = 0$ the hand orientation will only depend on $\theta 4 + \theta 6$. Since $\theta 4$ and $\theta 6$ cannot be determined independently from $\theta 5$. Using the system equations 2.116 we have:

$$\begin{aligned} C5 &= C23(S1Z_x - C1Z_y) - C23Z_z \\ S5 &= \pm\sqrt{1 - C5^2} \end{aligned} \tag{255}$$

Two symmetrical solutions are then possible for angle $\theta 5$ within $[-\pi, +\pi]$: Angle $\theta 4$ can be determined from the system (3-4) only when $S \neq 0$ (see next pages), we have:

$$\begin{aligned} S4 &= \frac{Z_x C1 + Z_y S1}{S5} \\ C4 &= \frac{C23(S1Z_x - C1Z_y) - S23Z_z}{S5} \end{aligned} \tag{256}$$

As $C4$ and $S4$ depend on the sign of $S5$, then two solutions are expected within $[-\pi, +\pi]$:

$$\{C4^+, S4^+\} \text{ and } \{C4^- = -C4^+ \ and \ S4^- = -S4^+\} \tag{257}$$

The solutions $\theta 4^+$ and $\theta 4^-$ differ by π . In the domain $[-\pi, +\pi]$ we have: $\theta 4^+(C4^+, S4^+)$

$$\theta 4^- = \theta 4^+ - sign(\theta 4^+).\pi \tag{258}$$

To determine $\theta 6$ when $\theta 5 \neq 0$ we assume the value of $C4, S4, C5$ and $S5$ have been computed according to equations 2.117 and 2.118. Let us express the rotation matrix M_5^6, we have:

$$M_5^6 = M_5^3.M_3^0.M_0^6 \tag{259}$$

Matrices $M_5^3(\theta 4, \theta 5)$, $M_3^0(\theta_1, \theta_2, \theta_3)$, and M_0^6 are given. The matrix M_5^6 is a $ROTZ(\theta 6)$:

$$M_5^6 = \begin{bmatrix} C6 & -S6 & 0 \\ S6 & C6 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{260}$$

Recall the matrices $M_3^0.M_0^6(3-3)$ and $M_3^5(3-1)$ that have been previously evaluated, we have:

$$M_5^3 = \begin{bmatrix} C4 & S4 & 0 \\ -S4C5 & C4C5 & S5 \\ S4S5 & -C4S5 & C5 \end{bmatrix} \tag{261}$$

Expression of $C6$ and $S6$ are then obtained from the product $M_5^3.(M_3^0.M_0^6)$ as follows:

$$\begin{aligned} C6 &= C4(C1X_x + S1X_y) + S4(-S1C23X_x + C1C23X_y + S23X_z) \\ S6 &= C4(C1Y_x + S1Y_y) + S4(-S1C23Y_x + C1C23Y_y + S23Y_z) \end{aligned} \tag{262}$$

Depending on the sign of $S5$, i.e. $C4$ and $S4$, we determine two solution for $\theta 6$:

$$\begin{aligned} S5^+ &= +\sqrt{1 - C5^2} \rightarrow (C4^+, S4^+) \rightarrow (C6^+, S6^+) \\ \text{And } S5^- &= -S5^+ \rightarrow (C4^-, S4^-) \rightarrow (C6^-, S6^-) \end{aligned} \tag{263}$$

Since, two solutions are also expected for $\theta 6$:

$$\theta 6^+(C6^+, S6^+)$$

$$\theta 6^- = \theta 6^+ - Sign(\theta 6^+).\pi \tag{264}$$

Figure 2.13 shows how these solution can be obtained: consider one initial solution (Figure A) and consider the operations $\theta_5 \leftarrow -\theta_5$ (Figure B), $\theta_4 \leftarrow -\theta_4 + \Pi$, and $\theta_6 \leftarrow -\theta_6 + \Pi$. Obviously, this lead to obtain another possible solution for the position and the orientation of the arm.

Conclusion on the case $\theta 5 \neq 0$

Two set of solutions are expected:

$$(\theta 5^+, \theta 4^+, \theta 6^+) \text{ and } (\theta 5^+, \theta 4^+, \theta 6^+)$$

where

$$\begin{aligned} &\theta 5^+ \quad \text{and} \quad \theta 5^- \text{are symmetrical} \\ &\theta 4^+ \quad \text{and} \quad \theta 4^- \text{Differ by} \pi \\ &\theta 6^+ \quad \text{and} \quad \theta 6^- \text{Differ by} \pi \end{aligned} \tag{265}$$

By trajectory continuity, we may identify a solution. For example, by comparing $(\theta 4^+, \theta 4^-)$ to the previous value of $\theta 4$ (Initial):
Example:

$$\begin{aligned} \text{If } |\theta 4_I - \theta 4^+| &> |\theta 4_I - \theta 4^-| \ THEN \\ \theta 4 &= \theta 4^+; \theta 5 = \theta^+; \theta 6 = \theta 6^+ \quad (266) \\ ELSE & \quad (267) \\ \theta 4 &= \theta 4^-; \theta 5^+; \theta 6 = -\theta 6^+ \quad (268) \\ END & \quad (269) \end{aligned}$$

Given the hand center $O_0O_{6,0}$ and the orientation matrix M_0^6, it is possible to find at least two configurations for the effector part $(\theta 4, \theta 5, \theta 6)$:

$$(\theta 5, \theta 4^+, \theta 6^+) \text{ and } (-\theta 5, \theta 4^-, \theta 6^-) \quad (270)$$

Case where $\theta 5 = 0$
This case corresponds to a singular configuration as shown on Figure 2.14.In this situation, the rotation axes of $\theta 4$ and $\theta 6$ are co-linear and concurrent, we have:

$$M_3^6 = M_3^4.M_4^5.M_5^6 = M_3^4.M_5^6 = ROTZ(\theta 4^\theta 6) \quad (271)$$

And

$$M_3^6 = \begin{bmatrix} C46 & -S46 & 0 \\ C46 & C46 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (272)$$

In order to express C46 and S46, we may use the following equation:

$$M_3^6 = M_3^0.M_0^6 \quad (273)$$

The product $M_3^0.M_0^6$ has been previously expressed, we have:

$$\begin{aligned} C46 &= C1X_x + S1X_y \\ S46 &= -(C1Y_X + S1Y_Y) \quad (274) \end{aligned}$$

Since $\theta 4$ and $\theta 6$ cannot be expressed independently from each other. In fact the mechanical analysis confirms this mathematical result, and then heuristics can be applied to obtain a possible alternative. For example we may keep unchanged one angle, i.e. $\theta 4$ or $\theta 6$, and determine the other angle from their sum $(\theta 4 + \theta 6)$ as it is identified using $C46$ and $S46$. Another alternative consists of estimating one angle, i.e. $\theta 4$ or $\theta 6$, by extrapolating its time function according to the previous values. Assume the previous values of $\theta 4$ are:

$$\theta 4(t-k), ..., \theta 4(t-1), \theta 4(t) \quad (275)$$

And

$$\theta 6(t-k), ..., \theta 6(t-1), \theta 6(t) \quad (276)$$

A polynomial approximation with degree K is given by:

$$\theta(t^1) = \sum_{i=0}^{K} a_i \theta(t-i) \quad (277)$$

Higher accuracy is obtained for lower degree polynomial. Since we may select to extrapolate either $\theta 4$ or $\theta 6$, according to the lowest polynomial degree. Assume $\theta 4$ is obtained by extrapolation, then $\theta 6$ could now be simply obtained as $\theta 6 = \theta 46 - \theta 4$. The above direct and inverse transformations were implemented and tested on the client station.

6.1.6 The man-machine interface

In telerobotics the objective of design of a man-machine interface is to provide the operator with an effective way to dictate his motion to the slave arm by only acting on a master arm. A good geometric interface must allow the user to set up the position and orientation of the slave arm gripper in a minimal number of trials, i.e. decoupling most of the geometric parameters from the operator hand to the slave hand frams. One of the man-machine interface problems is the problem of finding a good geometric mapping between the master arm (also operator hand) motion and that of the slave arm hand. In this section we examine a number of potential geometric mapping from the master arm to slave arm. We discuss each of these solutions based on its implementation and its experimentation and provide justification of selected mapping.

Teleoperation with master arm

Teleoperation and telerobotics are technologies that enable remote operations. Many teleoperation and telerobotic systems use dedicated communication links between human operator (master) and the remote manipulator (slave). Recently the LANs and Internet have supplied the communication link for many systems. In some cases, teleoperation systems include force feedback. Teleoperation technology supports a form of control in which the human directly guides and causes each increment of motion to be applied to the slave. Typically the slave robot follows the human motion exactly although in more advanced, computer controlled, systems there may be coordinate transformations imposed between the master and the slave. Many systems that are remotely controlled allow the operation to one of two predefined states, either on or off. Thus, we say that the remote control system is not a telerobot if it only permits the on/off selection of state. In many systems, master and slave arms are geometrically identical within a scale constant. In this case it is sufficient to transmit the angles of the individual joints in the master arm. Because of the geometric similarity, the motion of the slave end effector will exactly follow that of the master arm. Both master and slave side motions are therefore interconnected in joint coordinates. The original teleoperators developed for the nuclear operations used joint coordinates interconnection. For many applications, it becomes desirable for the master arm and slave sides to be geometrically different. In this case, the joint coordinate of the master arm do not specify the desired joint motion of the slave. Coordinate transformations based on the kinematic equations of the two device are required to resolve these different languages. This coordinate transformation became feasible in real time, because of the rapid advance in computer technology. For example, our master arm is designed independent of the slave arm, by completely different people/companies. Such systems where master and slave

have different geometric designs are called generalized teleoperation systems. Our passive articulated system was used to remotely control the PUMA-560 robot arm. The communication link between the operator and the robot is the LAN of 100Mbps bandwidth. The above master arm has six serial degrees of freedom with all revolute joints. Each joint is equipped with a potentiometer (sensor) which are used to measure the angles between joints. In addition to these six sensors, the holder at the end effector part of the master arm is attached with a Stop/Start switch and mode selector switch. The stop/start button is used to enable/disable the master arm, and hence can be used by the operator to disable the system temporarily, move the master arm to a comfortable location, and then enable the system. The mode selector switch is used to select the mode of operation. A data acquisition card is used for interfacing the master arm with the personal computer. A visual basic environment is developed to allow the communication between master arm and slave arm. The slave arm(PUMA-560) is connected to the server environment, while the master arm is connected to the client environment.

The effector part of the master arm

The effector part of the master arm is the orienting machine which is defined by the three revolute joints. θ_4, θ_5 and θ_6 with three concurrent rotation axes, i.e. the last three dof of the arm. The design of the effector part was changed several times for improvement and development purposes, while the transporter part remains the same.

We found later that the construction of this part is not suitable for the remote control operation, for the following reasons:

- A singularity problem arises when the joint 5 becomes along Y_4 axis. In that case the joint 5 cannot rotate about Z_5 axis and as a result the operator cannot rotate his hand about Z_H axis.
- The mechanical impedance is not identical in all directions which means that the operator cannot make abstraction of the structure due to lack of symmetric impedance.
- When the operator rotates his hand about Z_H, Y_H, or X_H, the transporter part will be translated for some distance which illustrates the mechanical coupling problem between the effector part and the transporter part.

To avoid the problem of singularity we decided to reduce the range of joint 5 to avoid a problem of singularity, but the operator hand get away from the origin of the concurrent rotations.

To keep the operator hand at the origin of the concurrent rotations, we modified the effector part as indicated in Section 6.1 which partially solved the problems of singularity and the origin of rotation. The modified structure makes the last three dof as concurrent and places the hand of the operator at the center of the rotation axes of the last three dof. Although this effector part gave better results and allow better operation than the previous one, it could not overcome the mechanical coupling and the non-symmetric impedance problems.

We decided to build another effector part as indicated in Section 6.1 which which overcomes all the problems of the previous structures. This effector part enables the human operator to make abstraction of the master arm and concentrate on the operation and the control of the robot arm.

Design and analysis of mapping schemes

To solve the problem of the difference in the geometric designs between the master arm and the slave arm (PUMA-560), we developed two mapping schemes. In which the operator at remote side can control the robot arm at the other end of the LAN by using the master arm. Initially we have developed the first mapping scheme. This scheme allows us for the first time to remotely control the robot arm. However, there are some drawbacks which complicate the control of the robot arm. Consequently, we developed the second mapping scheme. In the second mapping scheme we got excellent results. In the following, we will explain both mappings schemes.

The first mapping scheme

The first scheme was developed based on the idea that the operator will virtually operate the end point of the slave arm, while he is holding the end point of the master arm. In this scheme the position and orientation of master arm is assigned to the slave arm. The operator does not know the actual position of the slave arm, instead the master arm vector E_m is used to find the corresponding slave angles θ_P by using the inverse geometric model of the robot arm. All the computations of the control system are implemented in the client side. First of all, the angles of the master arm are gathered via the data acquisition card and then transformed to the corresponding position X_m and orientation matrix Φ_m, by using the direct geometric model of the master arm, $E_m = G_m(\theta_m)$ where $E_m = (X_m, \Phi_m)$. The vector E_m is used to find the corresponding slave angles by using the inverse geometric model of the slave arm as following:

$$\theta_p = G_p^{-1}(E_m) \tag{278}$$

The computed angles are incrementally assigned to the slave arm so that the end points and orientations of the slave and the master arm are equal. This scheme requires the configuration of both arms to be initialized to the same position. This is based on one single configuration mapping, but any shift will result in loss of correspondence. And as a result, the quality of operation will degrade significantly. This also means that there is accumulation of errors. The dexterous operation is not possible in this scheme, because shifting the master arm will lead to losing the control of the robot arm. The only feedback signal the operator receives from the robot side is the image data. This data is not enough to control the robot arm in an efficient way. Therefore, with this control scheme there is no way to correct the drift between the master arm and the slave arm.

Evaluation of the first mapping scheme

As we mentioned above, once the mapping is lost, the system get out of control and can not be returned to the correct state. There are few reasons for the loss of mapping:

1. Loss of data: This may happen at any time during the operation where the operator moves the master arm to some position but the data that is sent is lost. The data may get lost or deformed at client side, network or at server module.

2. Different solution: There is a chance that the robot arm receives a solution that is not acceptable, because of the different configuration of the master arm and the robot arm. In this case the mapping will be lost once it receives a wrong solution.

We experimented with this mapping scheme and we found that it needs very large number of trials every time we need to set up the robot arm at the same position and orientation. We found that this mapping scheme is poor and is not practical.

The second mapping scheme

In the second mapping scheme, the changes in the operator effector ΔE_m is assigned to the current effector of the slave arm, regardless of the current position of the operator hand. The computation of the master arm positions will be implemented in the client side. The increments in the motion of the master arm $\Delta E_n^m = (\Delta X_n, \Phi_n)$ Where:

$$\Delta E_n^m = G_m(\theta_n^m) - G_m(\theta_{n-1}^m) \tag{279}$$

$$\Delta X_n^m = X_n^m - X_{n-1}^m \tag{280}$$

$$\Phi_n^m = M_{n-1}^{T^m}.M_n^m \tag{281}$$

where n represents the current value, (n-1) represents the previous value and m is the symbol for the master arm.
The motion increments will be sent to the server module, where they will be added to the actual position parameters of the slave arm. The result of this operation is the new position parameters:

$$X_{n+1} = X_n^p + \Delta X_n^m \tag{282}$$

$$M_{n+1} = M_n^P.\Phi_n^m \tag{283}$$

where (n+1) represents the new or the expected value, and p is the symbol for the PUMA-560 robot arm.
The new position parameters , X_{n+1} and M_{n+1} will be transformed to the corresponding slave angles by using the inverse geometric model of the slave arm(PUMA-560 Arm):
$\theta_{n+1} = G_p^{-1}(E_{n+1})$ where $E_{n+1} = \{X_{n+1}, M_{n+1}\}$.

The new slave arm angles will be subtracted from the current angles of the robot arm as following:

$$\Delta\theta = \theta_{n+1} - \theta_n^P \tag{284}$$

$\Delta\theta$ will be sent to the robot controller and there they will be added to the current robot angles and will cause the robot motion. The actual position data of PUMA-560 are obtained from incremental encoders and potentiometers in the robot arm. This feedback signal from the robot arm provides a closed loop control of arm motion. The advantage of this control system is that the operator forgets about the master arm and only thinks that he is mapped to the gripper frame, of the robot arm without any consideration to the master arm position or configuration. This mapping scheme is dexterous, because the master arm can be shifted without affecting the mapping. This type of mapping between the master arm and the slave arm is dynamic and ensures there is no accumulation of errors, because the corrections are always made to the current robot effector.

Evaluation of the second mapping scheme

We have developed a mathematical interface at both client and server side. The interface is used to test the second mapping scheme before applying it to the master arm. We found this mapping scheme to be robust and universal. In the following we will give some explanation for the advantages of this mapping:

- Universal input: Once the incremental position and orientation is supplied to this mapping scheme, it will do the motion precisely regardless of the data source.
- Master-Independent: This mapping will accept the incremental motion from any master arm regardless of its configuration.
- Language-Independent.
- Scalable : The incremental position can be scaled up or down to fit the user requirements.

Anyhow, controlling the robot arm to perform some task is not an easy job. It requires some practice, specially if you are using the base frame as your reference. We will explain the different frames that are attached to the robot arm in the next section.

Mapping control

We have classified the modes to match the frames attached to the robot arm. The three frames are used as a reference to the modes that are used. All robot motions will be described in terms of these frames. The operator selects between the different modes, to perform his task easily. This facility makes it easier for the operator to control the robot arm in a more intelligent way. For example, he may find the operation with the base frame mode easier if he likes to perform a straight motion. If the operator likes the tool to move forward and backward without any change in

the orientation, he may need to use the tool frame mode...etc.

Brief explanation of the modes and their corresponding frames are listed below :

1. The base frame mode:
   The base frame {B} is located at the base of the robot or at some distance from the base. The base frame is also known as R_0. When the operator selects this mode, the motion of the robot will be described in terms of the world frame. In this mode the position and orientation of the tool frame which is attached to the end-effector is described or calculated relative to the base frame {B}.

2. The wrist frame mode:
   The wrist frame {N} is the frame attached to the last link of the transporter part of the robot arm (link 3). The origin of this frame is fixed at the wrist of the arm, and moves with the link 3. In this mode the position and orientation of the tool frame is described or calculated relative to the wrist frame {N}

3. The tool frame mode:
   The tool frame {T} is fixed to the end of any tool the robot is holding. In this mode the position and orientation of the tool frame is described or calculated relative to the frame attached to itself.

With the proper mode selection and by using good mapping scheme, the control of the robot arm becomes easier and better. with excellent systems the operator concentration is directed to perform his task as he is using his hand without any regard to the master arm configuration. The human operator will be asked to enter inputs in the form of keystrokes and mouse clicks to remotely operate and control the motion of the robot arm. How natural his interaction with the robot arm? How easy is it to drive the robot arm to a specific location?
The answer will clarify that the remote operation by using the mouse or the keyboard will not enable the operator to interact naturally and he can not forget that he is using these input devices. The human operator will be confused by the complex controlling mechanism and thereby the he will constantly think of what inputs should be entered rather than directly manipulating the robot arm, which reflects the difficulty of using such devices.
Our target was to design a telerobotic system in such a way that the human operator can remotely control the motion of the remote arm in natural and easy way and as simple as possible. This target is implemented by using good design of master arm, good mapping schemes, and by building good client-server system. The design of the master arm helps the human operator to forget about the complex controlling mechanism and to perform any complex motion with it. The development of the mapping schemes allows the human operator to lose the awareness that he is manipulating the master arm and see himself rather manipulating the robot arm directly. The operator can interact naturally and effortlessly with the robot arm, even with no skills. Our client-server system designed to keep the system reliable and to provides good environment for data transfer. The issues of network reliability, speed of data transfer, amount of data transferred, and informative and simple user interface were all taken into account in the design of our client-server system, which improve the teleoperation system performance.

Figure 5: The Eye3D box and LCD glasses.

6.2 The 3D visualization system

This section describes the setting of 3D visualization system under: (1) a local mode, and (2) networked mode. In the local mode both the scene cameras and the 3D visulalization system are attached to one single computer. In the networked mode the scene cameras are attached to a server computer and the 3D visulalization system is attached to a client computer, where the client and server computer are interfaced through a LAN. Using the project provided cameras and a borrowed eye shuttering system we describe our experiments on 3D visualization for both local and networked modes.

6.2.1 Local version of the 3D visualization experiment

In the local version of the 3D visualization experiment, we have a single host PC which has an IEEE-1394 interface card, and two digital video cameras that are connected to the card via Firewire cables. Between the host PC's display card and the monitor, we have the Eye3D hardware box (See Figure 5), which is capable of doing sync doubling. Namely, it can double the sync frequency of the signal output from the display adapter, and send it to the monitor. Finally, we have an liquid crystal display (LCD) glasses (See Figure 5) which opens/closes right/left eyes according to the infrared (IR) signal sent from the Eye3D hardware.

Figure 6: Two digital cameras mounted on an aluminum platform. A single tripod is holding the platform,

The experiment is performed as follows. First we turn on the two cameras, and start the AMCAP.EXE program twice to display the live video captured by these two cameras. Then we resize each AMCAP window so that they have equal size, and one is in the upper half of the screen and the other in the lower half. By using the Eye3D hardware, we start sync doubling. This will stretch the upper and lower halves of the screen to full screen size, and these stretched halves will be seen as overlapped by naked eye. At this point, we can make some fine tuning to make sure that both AMCAP windows are perfectly overlapped. In reality, because of the Eye3D hardware, the monitor will be scanning first one camera image, and then the other camera image, and continue like this in a periodic fashion. If one wears the LCD glasses, then one eye will be forced to see one camera image, and the other eye will be forced to see the other camera image, and this will give a 3D feeling to the user.

During the local version of the experiment, we made the following observations:

(1) Setting of the cameras: We tried first two seperate tripods, and observed that in order to get convergence of the left and right images viewed trhough LCD glasses: (i) One has to have the same tripod height and distance to scene (ii) The rotation angles of the last tripod elements that the cameras are attached, has to be the same (iii) Finally, one has to have proper orientation of the two video cameras, namely rotation along the z-axis. It is really difficult to adjust all these tripod/camera parameters. Motivated by this difficulty, an aluminum platform is designed, and the two cameras are mounted a single tripod as shown on Figure 6. This eliminated adjustment problems for (i) and (ii). Finally, once an adjustment is made for (iii), the cameras can be fixed to the aluminum platform. This greatly simplified the camera/tripod adjustment problems.

(2) Horizontal shift between two images: If the human eye is modelled as a pin-hole camera, then the object that we are looking at will not have any horizontal shift in right and left eye images. However, other objects will have

different horizontal shifts depending on their location. We have also verified this through experiments. We moved an object in the scene, and when its distance was close to the object that two cameras are focusing at, we observed no horizontal shift. However, as we move it backward or forward, we observed that the horizontal shift is also changing. Furthermore, we have done a very primitive augmented reality type experiment. We placed button figures on both camera images, and by playing with the horizontal location of the button figures on both camera images, we were able to add an artificial object at a given depth. Indeed, when there is no horizontal shift in button figures, the button is viewed at the same depth as the object that the two cameras are focused. Adding some horizontal shift in positive or negative directions moved the button figure backward or forward. This also confirmed one more time that, depth of the object is closely related to its horizontal shift in right and left images.

(3) Effect of tilted cameras: When the two cameras are parallel, in order to get convergence of the right and left images viewed through LCD glasses, we had to point the cameras to a far away object, the stereo effect was observed but it was not strong. When we use tilted cameras on the other hand, we had to point the cameras to a nearer object for convergence, and the stereo effect was stronger.

(4) Distance from cameras to scene: First, we have done a test where the two cameras are focused to a far away object. We have adjusted the camera orientation vectors so that the far away object has no horizontal displacement in right and left images. The result was successful, and we were able to see 3D. However the effect was not as strong as the case where the two cameras are focused to a nearby object. Indeed, when we focused the two digital video cameras to a nearby object, and then adjusted the camera orientation vectors so that the nearby object has no horizontal displacement in right and left images, the 3D effect was much stronger.

(5) Effect of camera focus and zoom: A normal human eye will get tired easily, if forced to look at out of focus images for a long period of time. Therefore, both cameras has be in focus. Furthermore, zoom levels should be exactly the same for convergence of left and right images viewed through LCD glasses. To guarantee this, we used a single remote controller for video cameras.

(6) Effect of scene lightning: Unless one is using very high end production quality professional video cameras, there will be noticable image noise (snowing effect) under low lightning. We also observed the same. Furthermore, florescent light is found to cause flickering, and hence user uncomfort. Therefore, proper lightning is very important.

Finally, we have done experiments where users look at the screen through LCD glasses, and just based on this visual feedback they try to place objects on top of each other by using their own hands. These experiments were successful, and users were able to stack objects on top of each other. Overall, we are quite satisfied with the performance of the local version of the 3D visualization experiments.

6.2.2 Networked version of the 3D visualization experiment

The networked version of the experiment exhibits much more complexity than the local version. There are two (or three) PCs in this version of the experiment. The server PC has the IEEE-1394 card, and two digital video cameras are connected to this card via Firewire cables. The server is supposed to send somehow right and left camera images to the client PC, which is also called as the visualization station. The visualization PC should draw these received right and left images on the screen in above-below format. Namely, one camera image should occupy the whole upper half of the screen, and the other one should occupy the whole lower half. When sync doubling is applied through the Eye3D hardware box, a naked eye will see the two images overlapped, but when viewed through LCD glasses, each eye will be forced to see a different image, and 3D visualization will be achieved. Therefore, the principle of 3D visualization is the same, i.e. sync doubling with above-below format, and observations made for the camera orientation, distance to scene, etc. all apply to this case as well. The main difference is in how to generate these right and left images on the screen in above-below format. In the local version we were able to generate two camera images in above-below format by using two copies of the AMCAP program. In the networked version, this needs more work. We tried the following three different techniques and observed the following results:

(1) Video Mixer PC: In this approach, on server side we have two PCs instead of one. The first server PC, which also acts as the video mixer PC, has the IEEE-1394 card and two cameras connected to it. Furthermore, its display adapter has video out. By using two copies of the AMCAP program, one can generate two camera images on the screen in above-below format. The video out signal output from the card is a composite video signal, and has two camera images in above-below format. On the server side, the second PC has a video capture card which has Video For Windows (VFW) style drivers, and takes as input the composite video signal output from the first PC's display adapter. The second PC also runs the popular NetMeeting software, and uses the video capture board as the source device. On the client side, again the NetMeeting software is running which is capable of displaying the received video signal on the screen. Note that, the received video is in above-below format, and can be used for 3D visualization by sync doubling through the Eye3D hardware box. Overall, we are not satisfied with this approach, because (i) It uses two PCs on the server side and makes the server complicated and fragile (ii) The available video resolutions provided by the NetMeeting software are quite low as it is optimized for higher frame rates per second while sacrificing from image quality. However, for 3D visualization one needs first high quality images, and for teleoperation high frame rates per second. Therefore, we ruled out this approach as the images drawn on the client PC were of poor quality although the frame rate was a couple of frames per second on a 100Mbps LAN.

(2) Using a SoftMixer: In this approach, on server side we have only one PC which does both the video mixing and transmission tasks. Namely, it has the IEEE-1394 card and two cameras connected to it, and also transmits video to the client PC via NetMeeting. A SoftMixer is indeed a virtual screen camera

utility. By using this software, the system "sees" a selected region on the screen as a video source. Therefore, we open two video display windows for the two cameras, stack them on top of each other, and then select only the area covered by these two windows. When NetMeeting is started, it will recognize this virtual screen camera as a video source. When a connection is established with the client PC, whatever is drawn on the screen inside this selected area is transmitted to the client station. Again the same sync doubling technique is used together with above-below format images for 3D visualization. Overall, we are not satisfied with this approach because (i) It uses a single PC for video mixing and transmission, and hence puts too much load on the server side which caused unstability (ii) The available video resolutions provided by the NetMeeting software are quite low as it is optimized for higher frame rates per second while sacrificing from image quality. Therefore, we ruled out this approach as the images drawn on the client PC were of poor quality, and because of high server load the achieved frame rate was less than 1 frames/sec.

(3) Custom Video Conferencing Methods: For 3D visualization one needs first high quality images, and for teleoperation high frame rates per second. This looks feasible only by custom video conferencing methods. We have designed and successfully implemented one such solution. First of all, on the server side, again we open two windows to display right and left camera images on the screen. We have designed a small server program which captures the contents of these two windows and writes them a to RamDisk. This RamDisk is also shared over the 100Mbps network. On the client side, again we have a small client program which reads images from the RamDisk and displays them in above-below format on the client station's screen. Furthermore, a simple sockets based synchronization technique is used for the client to wait for the server to generate images and save them to RamDisk, and server to wait for the client for clearance to delete these image files from RamDisk. Overall, the result was much better than the previous two NetMeeting based methods. First of all, the image quality was much better, which is essential for 3D visualization. The frame rate was 1-2 frames per second, and has room for further improvements. After the success of this method, we analyzed the reasons which reduce the frame rate, and come up with the following conclusions: (i) Use DirectX based Visual C++ programs to save images to RamDisk directly, rather than first copying them to video memory (i.e. displaying on the screen), and then capturing from video memory and saving them to RamDisk. (ii) To avoid server and client serialization, design a server which continously captures images from the two cameras and writes them to the RamDisk, and deletes files as it receives clearance from the client. Then client continously reads ready images from RamDisk, and sends OK messages to the server for clearance to delete. These two recommendations are expected to be implemented as soon as possible to further increase the achieved frame rate. Finally, augmented reality based techniques are also considered as a possible method for increasing the frame rate.

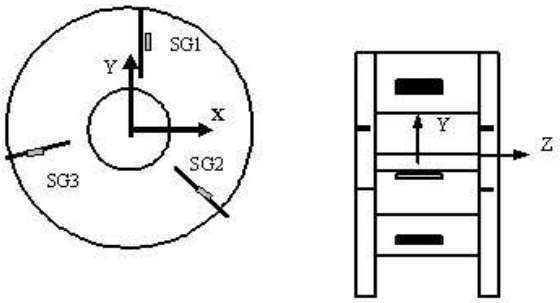


Figure 7: Basic Layout of Force / Torque Sensor

6.3 The design of a Force Sensor

In this task it is proposed to design and construct a 6 dof force/torque (F/T) sensor that will be mounted at the wrist of the Puma 560 robot arm. The ending link of the robot has a disk of 50 mm in diameter and has three threaded holes for mounting of wrist force/torque sensor or gripper. This section describes the design, construction and testing of a prototype 6 dof force/torque sensor. Finite element analysis of the prototype is also performed and results are presented and compared to the experimental data. Both results seem to be in good agreement. An attempt at estimating the sensitivity of the sensor to direct forces, bending moments and pure torques is also included. The design description is given below.

6.3.1 Design of a rigid force sensing structure

The basic structure of preliminary design of the sensor is composed of two solid aluminum disks 60 mm in diameter and 3 mm in thickness. Aluminum is the material of choice in the commercially available 6-dof force/torque sensors (JR3 and ATI). The material is usually selected for applications such as robotics where high strength to weight ratio is important. The solid disks are linked by three flexible elements (wings) cut from C- channel aluminum profiles. The channel height is 25 mm, the width is 15 mm and the thickness is 1.5 mm. The wings are bolted to the disks by M3 screws and are installed at 120^o in the radial direction, as shown in Figure 7.

To measure the deformations due to forces and moments acting on the wrist of the Puma 560 robot, three 2x4 mm foil-type precision strain gages are bonded on the faces of channels as shown in the Figure 7. The strain gages were bonded in the vertical direction in the middle of the wing. The radial position of the gages is also shown in the above figure. The selected strain gages allow a maximum deformation of $\pm 3\%$. The gage resistance is 120 $\pm 0.3\%$ ohms with a gage factor (GF) of 2 and a transverse-sensitivity factor K_t of 0.4 %.

Strain gages are commonly used for F/T sensing in applications where remote manipulation is practiced. Examples of these uses include the NASA Space Shuttle Remote Manipulation Sensor [71], the grasping force sensor [46], torque sensors [72],

and most of commercially available 6 dof sensors such as the Multi-Axis Load Cells manufactured by Multi-Axis Load Cell Technologies JR3 [73] and ATI-Industrial Automation [74]. In all these applications, the strain gage is used to measure the deflection of the structure due to forces and/or moments and their combinations. The measurement of the strain allows the determination of the applied force or moment. For simple cases, the relationships between these quantities can be summarized in the following.

- The axial load F, is obtained from the pure axial strain ϵ:

$$F = \epsilon AE \tag{285}$$

  Where A is the section area of the elastic element under tension or compression and E is the Young's Modulus of elasticity.

  For the foil-type gages the strain value should be corrected for transverse effects. The correction for transverse strain effects as suggested by Dally and Riley is:

$$\epsilon_{xx} = \frac{1 - \mu K_t}{1 - K_t^2}(\epsilon'_{xx} - K_t \epsilon'_{yy}) \tag{286}$$

  Where ϵ_{xx} is the true strain in the axial direction and ϵ'_{xx} and ϵ'_{yy} are the apparent strains measured in the axial and transverse directions respectively.

- The bending moment M is calculated from the axial strain ϵ as follows:

$$M = \epsilon ZE \tag{287}$$

$$Z = I/c \tag{288}$$

  Where Z is the sectional modulus which is equal to the second area moment of the section, I, divided by the distance from the neutral axis to the surface on which the strain is measured, c. In this case the strain gage is placed in the axial direction preferably one gage on each side as recommended by the manual by Omega [75],

- The torsional strain is better measured by placing strain gages at an angle of 45^o with the torsional axis. The torque T can thus be determined from the strain at 45^o (ϵ^o_{45}) :

$$T = \epsilon^o_{45} GJ \tag{289}$$

  Where J is the polar second area moment and G is the shear modulus of elasticity calculated as : $G = E/2(1 + \mu)$, where μ is Poisson's ratio that has a value of 0.33 for aluminum.

The output of the strain gage may be taken as an indication of the impressed force. The main problem with the use of strain gages for force measurement applications is that a moment may be impressed on the element under test because of the eccentric loading [76], This would result in an alteration of the basic strain distributions by the strain gages. There are means for compensating for these effects through installation of multiple gages properly interconnected to cancel out the deformation resulting from the impressed moment. Other methods for eliminating the shear strain which may be impressed upon the bending moment strain are described in the book by Dally and W. F. Riley [76].

6.3.2 Calibration (Testing of the prototype)

It is known that the position and the orientation of the strain gages has a significant effect on the strain measured and its sensitivity to applied loads. Before designing the appropriate electronics for the sensor it was decided to test the force/torque sensor using an existing multiple channel strain indicator. The gage wirings are connected to a portable data logger, which allows direct measurement of micro-strains. After setting the logger to the desired specifications testing started by checking the symmetry of the device.

Axial load

The axial load is applied at the center of the upper sensor disk. This is achieved by placing a spherical ball in the central hole of the upper disk on top of which a force is applied. The material testing machine is used to apply the load and give its intensity through the force cell. After initialization, forces varying from 10 to 80 N are used in all the testing cases to be presented here.

The readings from the strain gages (SG1, SG2 and SG3) in micro-strains ($\mu\epsilon$) are represented as a function of the axial load F(N) in Figure 8. It should be noted that the compressive strains are read as positive values in the strain indicator.

As expected, the first observation from these results is that of the linear variation of the strain (μ/m) with the applied load, F. The second is that all three strain gages yield similar readings, with a small increase in the difference between the strain indications at higher loads. These results indicate that the three strain gages are mounted symmetrically and that the combined data can be approximately described by the solid straight line obeying the following equation:

$$\epsilon = 0.615F + 3.84 \tag{290}$$

Where ϵ is the measured strain in (μ/m) and F is the applied load in N.

The sensitivity of the gage to the axial compressive load can be expressed as the slope of the line, ie 0.615 $\mu\epsilon$ / N. It can be seen from these results that the three lines intersect the Y-axis at approximately 4 μ m/m. Theoretically, all the lines should pass by the origin. The discrepancy can be explained by the fact that strain gage initialization was not done at zero load. This is mainly attributed to the low sensitivity of the testing machine load cell.

The values of the strain calculated from the above equation are apparent strains, created by a combination of axial compressive load and bending moment. As ex-

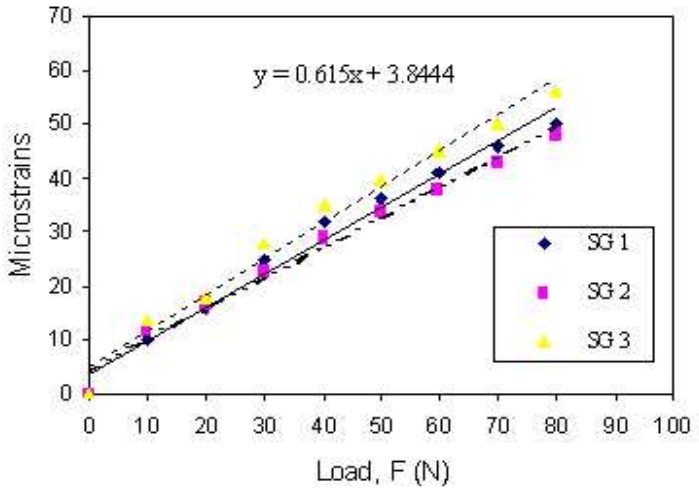


Figure 8: Variation of Micro-strain with Axially Applied Central Load

plained earlier, the load eccentricity with respect to gage axes impresses a bending strain on the axial one. This combination of loadings will be shown in more detail in finite element analysis. This problem if it deems important will be addressed in the improved version of the sensor.

Bending around X-axis (Eccentric Load)

In this test, the load F is applied in the center of the region between strain gages 2 and 3 (SG 2 and SG 3). Thus, the upper disk will be put under bending around the X-Axis (see Figure 7). The strain gages SG 2 and SG 3 will be under compression while SG 1 will be under tension. It should be mentioned here that the location of the center of the region required a lot of trial and error. Furthermore, the end of the rod applying the load is rectangular resulting more in a distributed load than in a point load. The variation of the strain ϵ (μm/m) with the applied load F(N) is illustrated in Figure 9.

Strain gages SG2 and SG3 show similar behavior under the eccentric load. The measured compressive strains are almost identical, proving that the load is applied equidistantly from these gages. Again, larger differences are obtained at higher loads. The combined results can be represented by either of the equations displayed in the graph. The reading from gage 1 (SG1) show that the corresponding wing is under tension. Its deformation is approximately half of the wings 2 and 3. The load F is later applied in the center of the other two regions (SG1-SG2) and (SG1-SG3) resulting in similar micro-strain load general behavior. Under compression loading, the sensitivity of the sensor varied from 0.58 to 1.02 $\mu\epsilon$ / N with an average of 0.8 $\mu\epsilon$ / N. Under tension (SG1), the slope of the curves varied from -0.29 to -0.37 $\mu\epsilon$ / N. This difference is mainly related to the sensor geometry and to the method by which the wings are bolted to the disks. The difficulty in applying the load at the

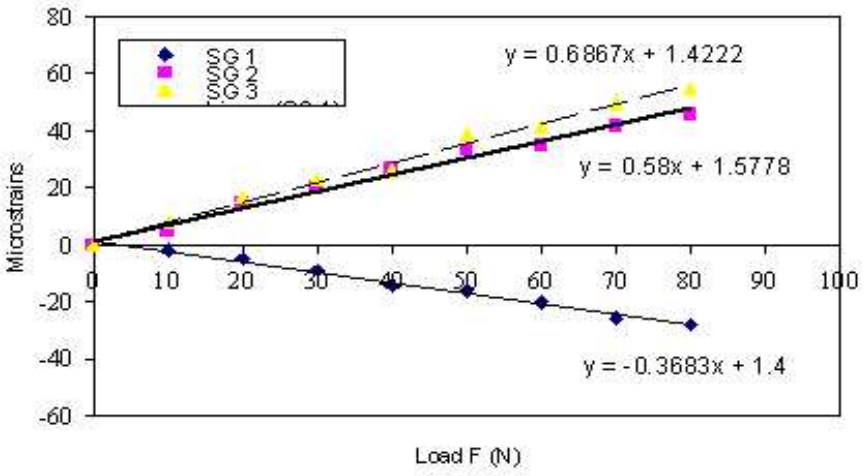


Figure 9: Variation of Micro-strain with Eccentrically Applied Load

exact center of the regions is another source of error. Again due to initialization problems, most of the curves intersected the y-axis at a value of approximately 1.5 $\mu\epsilon$. However, these results can be used to have a feeling on the position of the force on the sensor. A more involved analysis will allow the determination of the force magnitude. This will require determining of the stiffness matrix of the sensor.

Torsion Test

To test the device under pure torsion, a set-up was designed. The lower disk is rigidly fixed while two equal loads are applied to the upper one. This is achieved by hanging equal weights at the end of the ropes which are connected to the upper disk. The design is done such that it will result in forces tangent to the disk; creating a pure torque. To achieve this, two adjustable pulleys are employed. The friction in the pulleys is minimized by using ball bearings. The hanging weights were varied from 5 to 40 N resulting in torques varying from 0.3 to 2.40 N.m. Strain gages SG2 and SG3 showed similar behavior where the measured strain increased with the torque while SG1 was responding in a completely different manner. Troubleshooting and more torsion testing are underway to verify these results.

These preliminary experimental results will be analyzed to improve the design of the sensor. The following finite element analysis will also be used to help optimize the design of the sensor.

6.3.3 Finite element analysis

Given the complexity of the geometry, the experimental results cannot be validated by hand calculation. The finite element method (FEM) is thus used to give an

Load F (N)	Torque (N.m)	SG1((m/m)	SG2 ((m/m)	SG3((m/m)
5	0.3	19	-15	-16
10	0.6	32	-31	-36
15	0.9	39	-48	-59
20	1.2	42	-66	-85
25	1.5	34	-85	-109
30	1.8	20	-104	-138
40	2.4	10	-125	-170

Table 1: Strain Measurements During Torsion Test

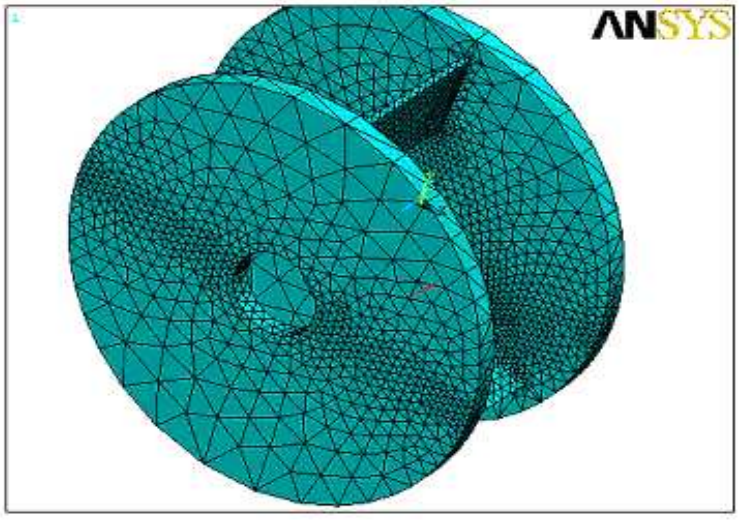


Figure 10: FE Model of the Sensor

indication of the strains under the different types of loadings described above [77]. The results will be used to optimize the design and to select the location and proper size of the strain gages in the final sensor design.

The Model

The finite element computer program, ANSYS [78] is utilized to model and analyze the structure of the sensor. To simplify the analysis, the wings and the disks are considered as one body. The whole structure is modelled using tetrahedral solid elements as shown in Figure 10. Finer meshing is applied in the areas where stress and strain concentrations are expected.

The disk at the base is constrained while the upper disk is subjected to the different loads to simulate the experimental set up.

Axial Loading Results

To simulate the first experimental test an axial load of 100 N is applied to a spherical ball modelled in the central hole of the upper disk. The model is as shown in

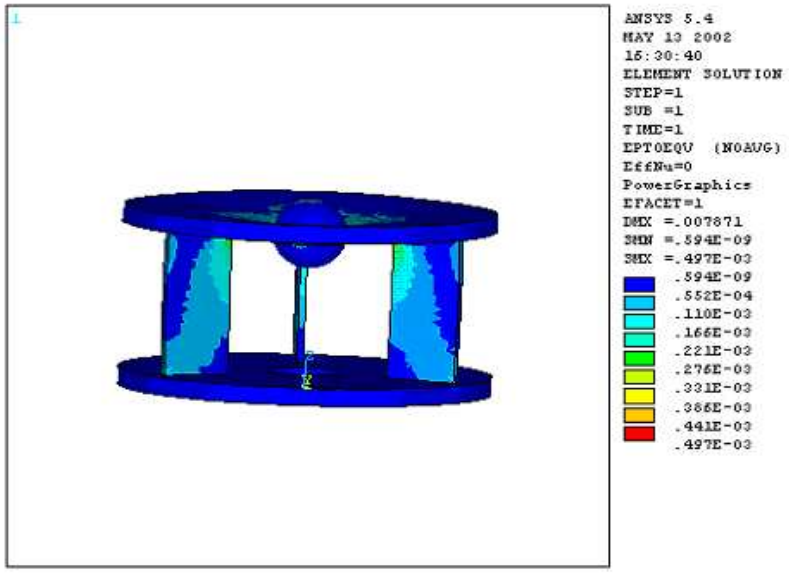


Figure 11: Finite Element Model of Sensor for Central Axial Loading

figure 11.

Figure 12 illustrates the deformed structure of the sensor under a load of 100 N applied at the center of the upper disk. The strain distribution is depicted by the iso-strains with different colors. It can be seen that there is an inward deformation of the upper disk with strain concentration at the inner end of the wings. The approximate position of the strain gage is in the mid-length of the wing in the center. From the calculated strains, shown on the right hand of the drawing, the values of the strains in this area are contained between 55 and 110 μm/m. The value of the measured strain for 100 N can be estimated from equation 5 as 65μm/m, which shows a good agreement between the calculated and the measured data. The way the structure has deformed gives a strong indication of the presence of bending under this type of load. The strain measured by the gages are thus the apparent axial strains which include bending strains.

Eccentric Load Results

In this part of the analysis, a concentrated load of 100 N is applied in the region between SG2 and SG3. The deformation of the upper disk as well as the strain distribution in the wings where SG2 and SG3 are bonded. Again the rectangle indicates the approximate position of SG3. The average value of the strain in this area varies between 63 and 189 μm/m. The value of the experimentally measured average strain (Figure 9) is within this range. The difference between the measured strain values and the computed ones is partially due to the fact that in the experimental test a distributed load was applied while in the FEM the load was modelled as a concentrated one. The idealization of the structure in the finite element model should be a factor contributing to the observed difference.

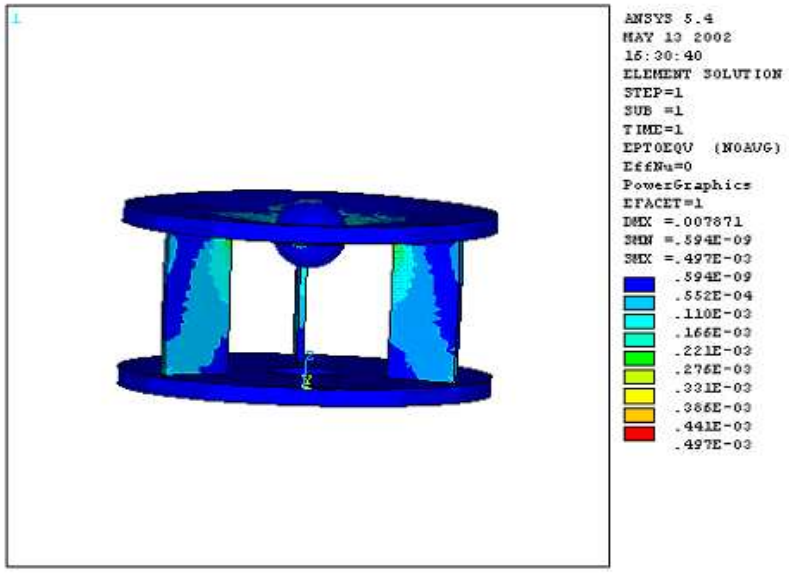


Figure 12: Strain Distribution in the Sensor Structure under Axial load of 100 N

Torsion Results

A torque of 4.8 N.m is applied to the upper disk. Figure 13 shows the resulting distorted structure along with the strain distribution in the wings. Neglecting the excess deformation, it can be seen that the strain distribution on the appearing wings is similar proving the axis symmetric behavior of the structure under torsion. The maximum tensile strain ranging from 0.035 to 0.045 μm/m occurs on the face of the right wing at the maximum radius in the mid-height. Similar behavior is observed on the back of the left wing where the strains are negative. The bending of the wing that produces this type of strain distribution is depicted in the front wing.

It should be mentioned that under this value of the applied torque large deformations of the wings occurred leading to yielding of the material. Future analysis will take into account the limits of the material and that of the gage (3%).

The above preliminary experimental and numerical results seem to indicate that the structure may not be adequate for sensing torques around the Z-axis. Further improvements and analysis are underway to arrive at final design.

6.3.4 The design and testing of a Compliant force sensor

A compliant force sensor is useful to provide both mechanical and electrical compliance at the tip of the slave robot arm. The electrical compliance or active compliance can be programmed to control the behavior of the slave arm in the presence of external forces. Its reaction depends on the activation of its control program. The presence of some mechanical compliance at the tip of the slave arm increases system reliability in simple and complex assembly operations involving contact forces with unknown positioning. These conditions are present in most telerobotic operations involving object manipulation. In the following we present the design of compliant

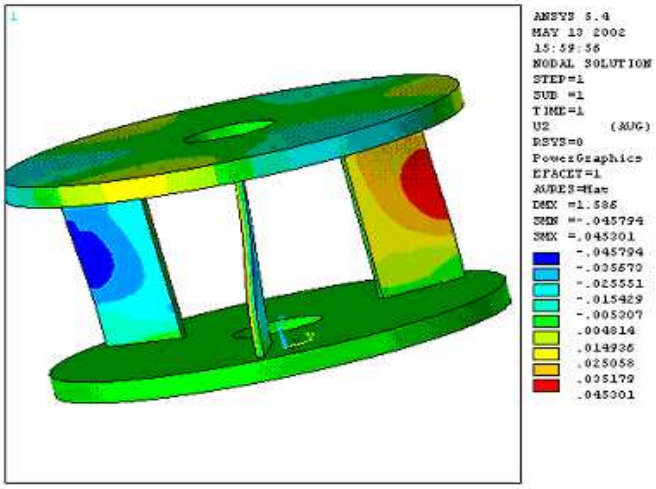


Figure 13: Distortion of Sensor Structure under Torsion

force sensor for teleoperations.

The compliant force sensor (CFS) is a wrist device that consists two solid aluminum disks each is 60 mm in diameter and 3 mm in thickness as shown in Figure 14-(a). The two disks are interconnected by means of three cubic rubber blocks which are distributed along the three directions (120 degrees) of the disk. The three rubber blocks are attached by constraining their base motion at each side of each disk. As a result an external force or moment applied on the top disk causes a small deformation in the position and the orientation of the top disk compared to that of the bottom disk. The magnitude of the deformation is function of the rubber shape and elasticity. Ideally an exitation applied in different directions produces a deformation having the same magnitude in all directions.

Since the external forces and moments produce deformations on the top side of each rubber block compared to its bottom it is normal to place the measurement points (sensing points) close to the rubber top as shown in Figure 14-(b). Since we have three rubber blocks we need six sensing points. For simplicity of the associated electronics a photo-electric sensor is used at each sensing point. The sensor consists of a light emitting diode (LED) that generates a circular beam (3 mm diameter) of red light in front of a photo-transistor (PT). A wing is used to mask the light flowing from the LED to the PT. Depending on the amount of masked light the PT generates a voltage that is proportional to the intensity of the received light, i.e. unmasked light. The wing is attached to the rubber top which is also attached to the top disk. Both LED and PT are attached to the bottom disk. Thus the motion of the top disk can be measured by the electrical signal generated by each PT.

The associated electronics is very simple because the internal impedance of each PT is indirectly modulated by the amount of received light. Thus each PT is set in series with a resistance and a constant voltage of four volts is applied on one side

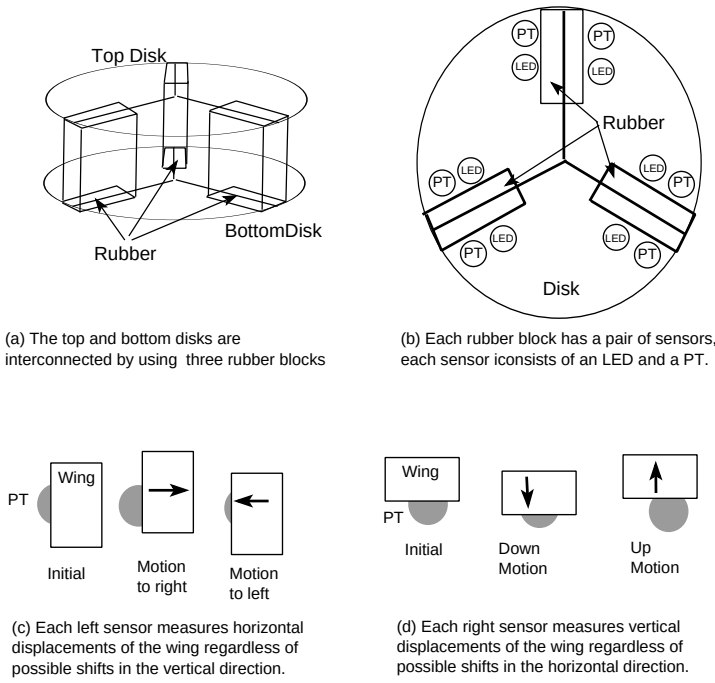


Figure 14: Some details of the compliant force sensor

while the other side is grounded. Variation in the PT impedance produces variation in the voltage at the middle point between the PT and the resistance. Therefore the output of the sensor is the voltage at the middle point. This produces a range of output voltage is about three volts without the need of instrumentation amplifier.

Each rubber block has a left and a right sensors. The left sensor is to detect horizontal displacement and the right sensor is to detect the vertical displacement of the top rubber that is attached to the top disk. Our design allows to uncouple horizontal displacement from vertical displacement at each sensor. A sensor that is instrumented to detect horizontal motion uses a wing that is relatively large compare to that of the PT in the vertical direction. In this way a translation of the wing in the vertical direction does not produce a change in the horizontal direction of the wing and consequently the light received by the PT is the same regardless of the vertical motion of the wing. This is shown on Figure 14-(c). This mechanism allows uncoupling the vertical displacement from a measured horizontal displacement. Similar effect is also implemented for uncoupling the horizontal displacement from a measured vertical displacement which is shown on Figure 14-(d).

The compliant force sensor was tested as a wrist sensor on the PUMA 560 robot arm by implementing its own simple electronics and interfacing the sensor to a PC card providing the needed A/D conversion and data acquisition functions. The testing experiments are based on selective application of force and torque and measurements of the corresponding output signal from the compliant force sensor. Specifically, the experiments consist of (1) applying specific external forces and moments, (2) reading by the PC of the sensor signals, and (3) plotting the six sensor responses. We studied each setting of individual force and torque with both positive and negative magnitude. The plots are shown on Figures 15 and 16. Analysis of the

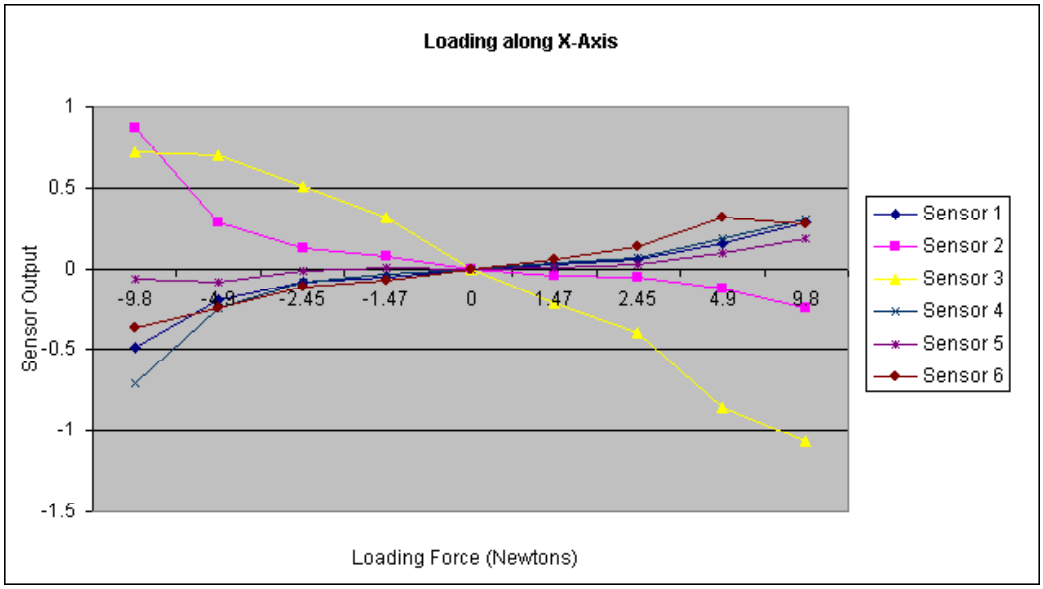


Figure 15: Optical sensor outputs versus loading force F_x

output signal proved to be consistent with the applied force or moment. The lack of anti-symmetry in the signals is attributed to the fact that the gripper is attached to one disk while the two disks are linked to each other by using rubber traversed by a screw shaft. The rubber must tight enough to hold the gripper. This reduces the sensor ability to detect axial forces due to the above axial attachment. The problem of having different force sensitivity was solved by finding software constant, in a heuristic way, that provides some accpetable linearity in the information.

Our conclusion from the testing are the following:

1. The attachment of the rubber blocks to the disks is not reliable and we need a simple mechanical attachment. For this we designed attachement bases to constrain the rubber blocks on each disk.

2. The use of rubber to provide some compliance is subject to the traditional hysteresis problem. As a result the sensor signal do not return to their previous values when the external force is removed. To reduce this effect we used very elastic rubber and directed the software to measure signal variations as opposed to absolute signal values. We also adopted a model of the sensor that estimates the external forces exerted on the manipulated tool.

3. Due to the use of optical devices for the detection of small motion the setting of the zero reference for all the sensors is difficult. However, measuring and processing only signal variations aleviates this problem.

4. There is a clear linear variation of sensor signal with the applied load. For master-slave teleoperation, the linearity of the sensor was found to be acceptable after applying the above processing techniques.

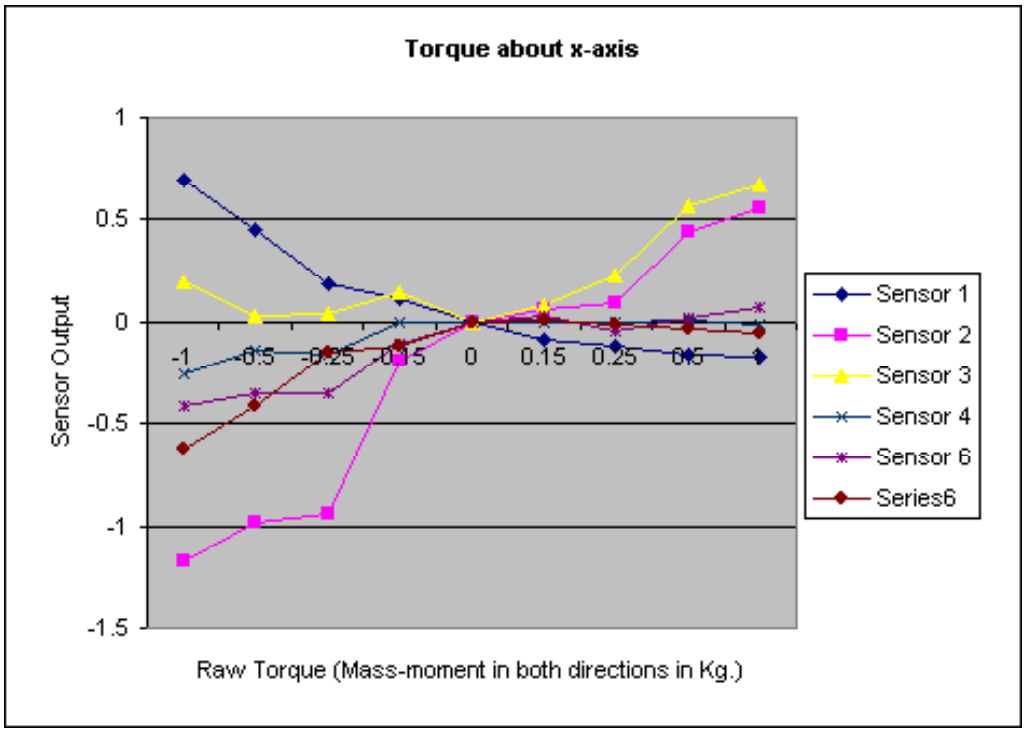


Figure 16: Optical sensor outputs versus loading moment C_x

6.4 A multi-threaded distributed telerobotic framework

In this section we describe the development of proposed *Multi-Threaded Distributed Framework* (MTDF) which is based on server and client side telerobotic components and their interactions with each other in a distributed application. Client and server support to stereo vision for telerobotics including stereo image acquisition, streaming, and display are presented in section 6.2. In this section we only present the software architecture of proposed MTDF. Appendix A presents the client-server software components with their mathematical functionalities. The mathematical functionality of each component will help the reader understand the service provided at each component activation in proposed object-oriented, distributed components client-server software system.

The MTDF is based on is based on the implementation of client and server components that are reliably connected by stereo, force, and commands data streams through a LAN network. The multi-threaded aspects stem from the simultaneous client and server threads. Server threads simultaneously carry out grabbing of stereo video data, reading force sensors, sending control signals to the robot, reading the feedback from the robot servo controller, and sending and receiving all of the above information over a LAN to one or more clients. The client threads, simultaneously, take care of (1) grabbing of position data, performing impedance control, and displaying of force feedback on master arm, (3) displaying stereo images on client 3D visualization system, and (2) transmitting to server differential master arm position through the LAN. This approach allows the logic of the system to be distributed in different software and hardware components.

6.4.1 Server side components

The server components are: (1) PUMA Component, (2) Force Sensor Component, and (3) Decision Server Component. These components implement interfaces that are used in remoting the objects. The details of the functionalities of these components and their interfaces will be discussed in the following sub-sections.

PUMA component

PUMA component acts as a software proxy of the robot for which commands are issued to the component as they are issued to the robot. Whenever robot changes its states, the component updates itself automatically to reflet these changes. A block diagram of the PUMA component is shown in Figure 17.

Some important *public methods* exposed by PUMA component include *ConnectRobot* that connects server to slave robot, *InitializeRobot* sends a program to robot that repeatedly moves the robot in an incremental fashion which reduces the communication data payload. Two overloaded methods allow us to incrementally move robot using either joint (angular position) or cartesian space parameters.

The PUMA component reads current robot joint $\theta_P(t)$ as a 6×1 vector. A command for an incremental joint motion $\Delta\theta$ is sent directly to the robot. A command for an incremental cartesian motion is specified in hand frame translation vector ΔX (3×1) and orientation matrix ΔM (3×3). PUMA computes the new robot hand position $X_{new} = G(\theta_P) + \Delta X$ and orientation matrix $M_{new} = M_P.\Delta M$, where

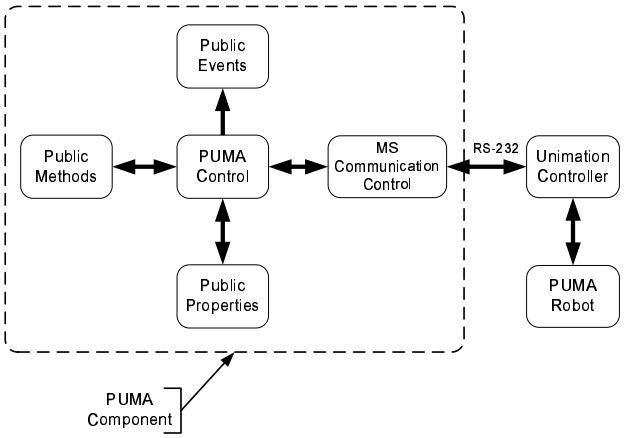


Figure 17: PUMA component and its interface to robot arm

$G(\theta)$ is the direct kinematic model of slave arm and M_P is the current robot hand orientation matrix. PUMA computes the inverse kinematic for X_{new} and M_{new} and finds the corresponding joint vector $\Delta\theta$ which is sent to robot.

The PUMA component accepts a user defined tool frame of reference as (1) robot base frame (world), (2) robot wrist frame, or (3) robot tool frame. Robot statuses are (1) connection to robot is not detected or robot not initialized, (2) Robot is connected but not initialized, (3) initialization is pending, (4) robot is ready to receive a motion parameter, (5) robot is moving, etc. The PUMA component also makes available some public properties to set or retrieve the attributes of the robot in realtime. The major properties are: (1) reading robot Angles, (2) computing the position vector and orientation Matrix of robot hand frame, (3) setting up specification of robot tool frame, (4) setting up the communication between server and robot, etc. The events invoked by PUMA component include: (1) data received from PUMA, (2) some error occurred with PUMA, (3) robot moved to a new location, and (4) PUMA status changed.

Force sensor component

The force sensing component (FSC) reads the robot wrist force sensors and creates a stream of reflected force feedback directed to the master station. The block diagram of FSC is shown in Fig.18. FSC is implemented as a separate thread, the priority of which can be adjusted during runtime to allow for the management of CPU usage.

A new instance of FSC creates a new thread with a default *normal* priority and waits until the sensing is triggered. After the reading has started, it continues sensing the force information at a pre-specified, alterable, default frequency. It invokes an event to inform the parent application of the availability of a new force packet. The parent application can respond to this event using some event handler at the higher level of application hierarchy. The event directly transfer the force information bypassing the global memory. Similarly the component also provides *StopReading()* function to abort the force sensing thread anytime we want. This will free the CPU of the load of the force thread and all events coming from the Force Component will stop. After the force thread is stopped, the force sensing can

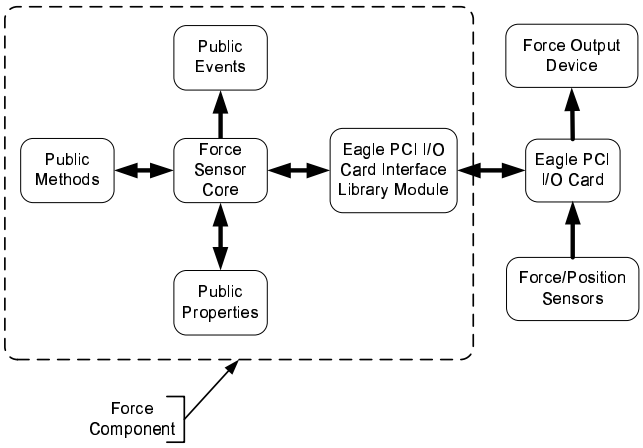


Figure 18: Force sensor component and its interface to sensor

be triggered again using *StartReading()*.

There are three public properties exposed by FSC. The *SensorThreadPriority* is used to set the thread priority that is one out of five OS(operating system) provided levels. However, the *Highest* priority does not guarantee non-preemptive thread behavior because of the operating system constraints. The *TimerValue* is used to set a time interval between two successive readings. This is useful to set up the frequency of the force sampling using this property. The *ThresholdValue* is helpful in situations when we need the force event to be invoked only when there is noticeable change in at least one of the force sensor values. We can set the *ThresholdValue* property in accordance with the minimum change that we want to observe.

Decision server component

DecisionServer is a component that provides an autonomous local loop on the server to support supervisory telerobotic control. This is needed to avoid teleoperator using stop-and-wait strategy in the presence of significant network delays. This is a higher abstraction layer which is used as an agent to implement local robot impedance control and workspace scalability functions. This layer can also accommodate the repeatability of a set of movement commands. For this we need an interface that allows a remote client to control a distributed telerobotic system.

The presence of this layer allows high-level control of the robot based on force sensor data. Another advantage of this layer is the implementation of different modifiers to the commands coming from the client, for example workspace scalability function can be implemented on this level. If a learning mode is implemented in the future, DecisionServer can serve as an agent that will record the trajectories and will reproduce them by implementing an impedance control using PUMA and Force Sensor components. A block diagram explaining the role of DecisionServer in the hierarchy of the system on server side is shown in Figure 19.

The server side logic is implemented in four layers. The last layer down the hierarchy is the physical layer consisting of robot and force sensors. On the highest level of the hierarchy is the human operator that might interact with the system using a UI(user interface). A possible autonomous local loop on the server side can

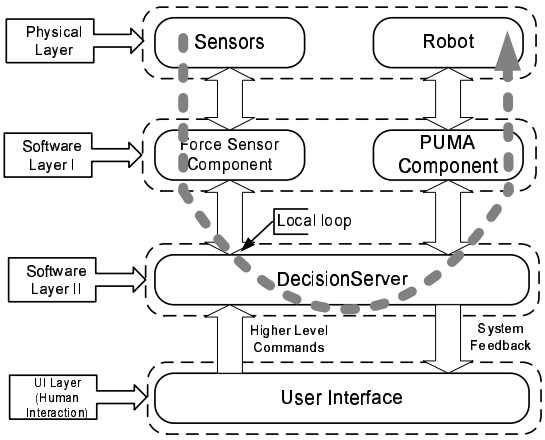


Figure 19: Component fierarchy on the server side

be constructed in the lower three layers. This can help automate the execution of simpler tasks in the presence of large time delays.

Server side interfaces and .NET remoting

An interface is a set carrying definitions of public methods and properties. It servers as a contract for any component that implements this interface. In other words, any component that inherits or implements the definitions contained in an interface, must provide the implementation of all the methods or properties enumerated in the interface. This scheme is needed in .NET based distributed applications because any client that accesses or executes the methods of a component on the server needs an access to the server assembly or component. By giving a reference to an interface that the server component implements, we can hide the actual component or assembly from the client. This provides security from potential unsafe clients as well as gives the developers freedom to the easily amend the logic of the server methods while the interface remains unchanged for all the clients because an interface is only a definition, the implementation being inside the component.

In order to attain references to both the PUMA and Force Sensor components, two interfaces are defined: *IProxyRobot* and *IForceSensor*. These interfaces carry the definitions of public methods, properties and events of PUMA and Force Sensor components as explained in sections 8.1.1 and 6.4.1. Further we define another interface *IDecisionServer* which inherits both the *IProxyRobot* and *IForceSensor* interfaces. Using this approach we are able to define a unified set of public members (methods, properties and events) that are required to be implemented in the form of DecisionServer component on the server side.

Once *IDecisionServer* is fully implemented, .NET Remoting can be used to publish an instance of DecisionServer component on the LAN. This instance is identified to potential clients by a unique object identifier issued by .NET Remoting. Any client can get a reference to this instance through an *IDecisionServer* interface. .NET Remoting enables accessing objects using SOAP(Simple Object Access Protocol). This scheme isolates the network protocol issues from the software development of a distributed application. Any object/component that might be located

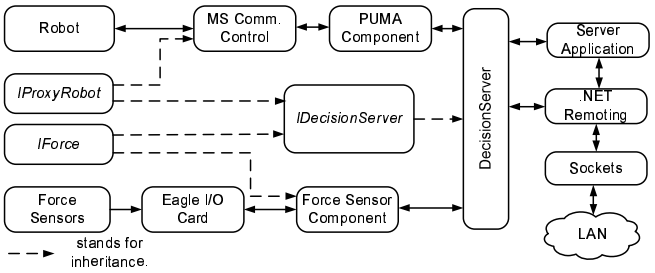


Figure 20: Integrated scheme - server side

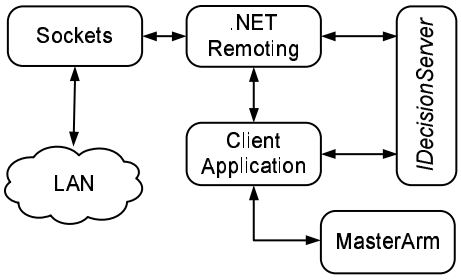


Figure 21: Integrated scheme - client side

on the other end of the world can be referenced using this distributed scheme as if it was available on the same machine.

6.4.2 Client side components

The client contains the *IDecisionServer* interface to reference the server side component through .NET Remoting. In addition to *IDecisionServer*, there are instances of .NET Remoting and client GUI(Graphic User Interface).

Decision server interface

The integrated scheme incorporating all the components on client and server side is shown in Figures 20 and 21. .NET Remoting is responsible for making socket calls to the client and we may choose either network protocol for these requests. The client side, as shown in Fig. 21, is fairly simple and contains all the familiar components which have been explained previously.

An important feature that we need on client side is to receive the events fired by the DecisionServer instance on server side. In order to use an event handler for any event invoked by DecisionServer, we must provide the client assembly to the DecisionServer. This violates object oriented design philosophy and introduces potential security threats. To overcome this issue, we have used *shim* classes as intermediatory agents to forward DecisionServer events over to the client or *IDecisionServer* interface. Shim classes are thin assemblies visible to both the server and the client. DecisionServer invokes the event which is received by an event handler hooked by shim classes. This event handler then calls the event handler of the client (*IDecisionServer*). By following this approach we hide the server and client assemblies

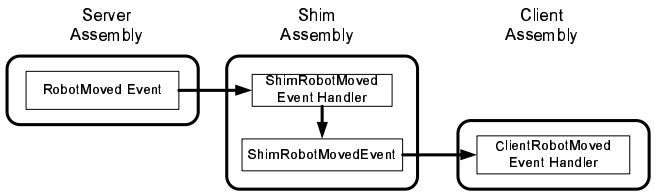


Figure 22: Forwarding events from server to client using shim classes

from each other. A diagram showing the events being forwarded with the help of shim classes is shown in Fig. 22.

Care must be taken while receiving events from the server and writing event handlers for them because these are synchronous events which means that the thread that is invoking the event on the server side will be blocked until all the event handlers for this event are executed. So manipulating different threads in a multi-threaded application, especially the GUI thread during the invocation of the events may cause deadlocks in the distributed client-server environment.

MasterArm Component

The *MasterArm* component implements the functionality required to carry out real-time rendering of the operator motion as well as to display haptic feedback (force) on operator hand. The *MasterArm* component, after initialization, has active instances of two force components for reading and writing in different threads.

It also implements a generic impedance control to minimize arm inertia that aims at reducing the mechanical impedance of master arm manipulated by the operator. The local force feedback uses a second order model for minimizing the mechanical impedance of the master arm. In order to estimate the force feedback, the component maintains a record of all the force data read for a certain number of samples(history) along with the record of the system time. Then it evaluates the velocity and acceleration of the master arm at each sampling instant. This information is used to calculate the force proportional to what the operator is applying which is then fed back to the master arm.

In the following we shortly describe the generic impedance control implemented as part of the *MasterArm* component. Using a 6 dof master arm, the *MasterArm* component has control of a 6×1 motor torque vector τ which in turn controls the dynamics [11] of the master arm articulated system:

$$\tau = D(q)\ddot{q} + C(q, \dot{q}) + G(q) \tag{291}$$

where q is master arm joint angular vector, $\dot{q}$ velocity vector, $\ddot{q}$ acceleration vector, $D(q)$ is the inertia matrix, $C(q, \dot{q})$ is the coriolis and centrifugal coefficients, and $G(q)$ is the gravity vector. The operator motion is characterized by vectors q, $\dot{q}$, and $\ddot{q}$. This allows computing terms $C(q, \dot{q})$ and $G(q)$ based on the geometry and dynamics of the master arm. The inertia matrix $D(q)$ is nearly constant for a light master arm operating in a restricted work volume. The *MasterArm* component computes the motor torque as follows:

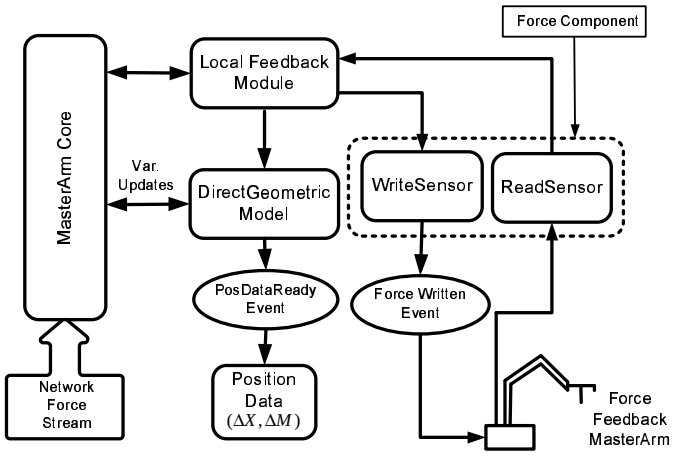


Figure 23: MasterArm component

$$\tau = \alpha\ddot{q} + \beta\dot{q} + C(q,\dot{q}) + G(q) + \tau_{ff} \tag{292}$$

where term $\alpha\ddot{q} + \beta\dot{q}$ is generated based on the operator motion, terms $C(q,\dot{q})$ and $G(q)$ are used to compensate for dynamic effects and gravity, and τ_{ff} is the reflected force feedback torque. The overall dynamic motion equation becomes:

$$\tau_{ff} = (D(q) - \alpha)\ddot{q} + \beta\dot{q} \tag{293}$$

where term $D(q) - \alpha$ represents the reduced master arm inertia (impedance) and $\beta\dot{q}$ is a motion damping factor. The motivation for injecting term $\alpha\ddot{q} + \beta\dot{q}$ in the master arm torque is to reduce overall mechanical impedance felt by the operator. The values of the parameters α and β can be found experimentally so that the operator feels that the master arm is really helping him in moving it. A functional block diagram of the *MasterArm* component is given in figure 23.

There are two major inputs to the *MasterArm* component, 1) angular position being read from master arm and 2) the network stream of force data coming from the remote side. *MasterArm* uses the *ReadForce* module of *Force* component to read the position data from master arm joints.

Here we describe the real-time monitoring and forwarding of operator hand position and orientation. Using the joint vector q computing the kinematic model $G_M(q)$ of the master arm allows computing the operator hand position vector X_k and orientation matrix M_k at iteration k. The component also computes the incremental hand vector $\Delta X = X_k - X_{k-1}$ and incremental orientation matrix $\Delta M_k = (M_{k-1})^t M_k$ because $Mk = M_{k-1}\Delta M_k$. The *MasterArm* sends the computed data ΔX_k and ΔM_k to the slave arm as an incremental motion command for the current slave tool position and orientation. *MasterArm* also injects the force feedback term in the output of the master arm torque vector which is sufficient to "display" to operator the received force feedback.

MasterArm is a multi-threaded component that can read and write data simultaneously as well as process lengthy operation in worker threads. The *MasterArm* component also invokes events when 1) a fresh copy of position data (incremental

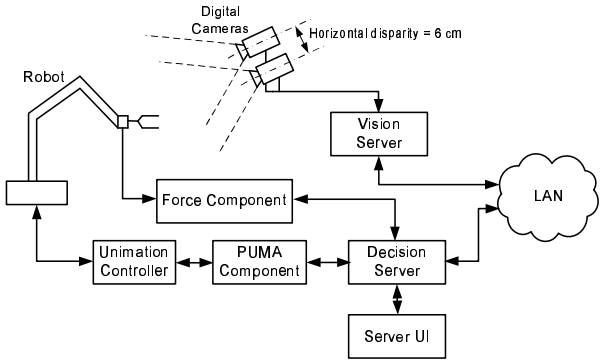


Figure 24: Server side of the distributed framework

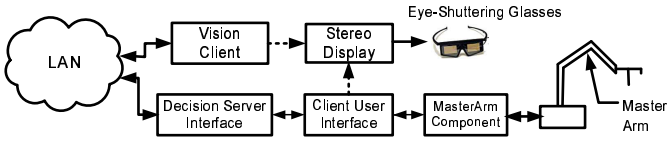


Figure 25: Client side of the distributed framework

cartesian position data) is available from ReadForce and 2) when some force data is written to the master arm.

Some of the public methods revealed by *MasterArm* are used to: (1) start and stop reading the master arm position (Inherited from *Force* component), and (2) write the given force data to the master arm in a separate thread. Now we describe some of the the public properties. One property computes the change in position vector and orientation matrix after the position data ready event is fired, A boolean property is used to find/set whether the master arm is engaged or not. If this property is false, the direct geometric model will not be evaluated to save thread time. A get/set boolean property is used to indicate whether to provide force feedback to the master arm or not. This feedback is the force stream coming from remote side. Other properties are also used to compute the local impedance control function for the master arm.

The integrated scheme incorporating all the components on client and server side is shown in Figures 20 and 21.

6.4.3 The complete system

This multi-threaded distributed telerobotic system is based on the implementation of all previously described client and server components. A complete view of server and client sides of the proposed multi-threaded distributed telerobotic system is shown in Figures 24 and 25.

Two video cameras generate stereo pictures which are sent to the client using the vision server. The user may issue commands to the DecisionServer which in turn makes use of PUMA and Force Sensor components to carry out these commands. Both the stereo video data and the distributed component calls share the same LAN,

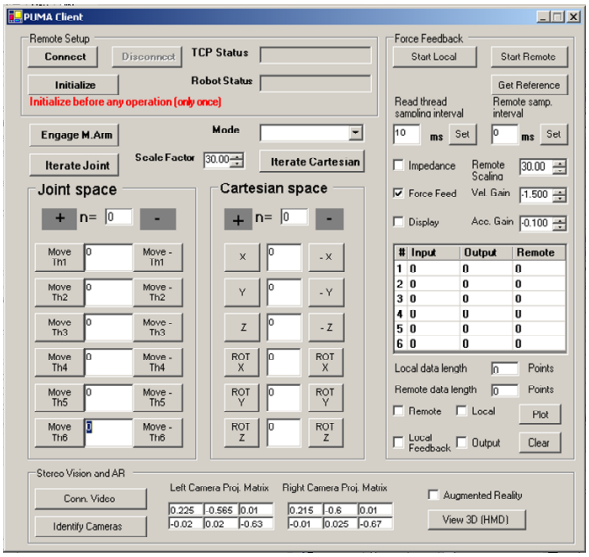


Figure 26: Client Side Graphic User Interface

however they open different ports for the data transfer.

The client side uses the GUI as well as master arm to issue commands to the slave arm on remote side. The vision client receives the synchronized stereo data from the LAN through windows sockets and provides a stereo display of the remote scene to the viewer with the help of eye-shuttering glasses. A view of the client GUI is shown in Figure 26.

6.4.4 Performance evaluation

Evaluation is carried out at the following levels: (1) force feedback multistreaming delays, (2) stereo vision multistreaming delays and thread engineering, (3) evaluation of force feedback, and (4) comparision to others.

Multistreaming effects on force feedback

Performance evaluation experiments under different conditions were carried out on the distributed framework described in section 6.4.3. The bandwidth of the LAN is 100 Mbps and both the client and server PCs are 2.0 GHZ P-IV machines with 1 GB DRAM. Each force data packet contains 6 double values or 48 bytes. The experiments are explained in the following sections.

Force streaming only

In this setup, only force information is transferred from the server to client. There is no video transfer, neither any command signal is present during the experiment. A distribution of inter-arrival times of force packets is shown in Figure 27. This data

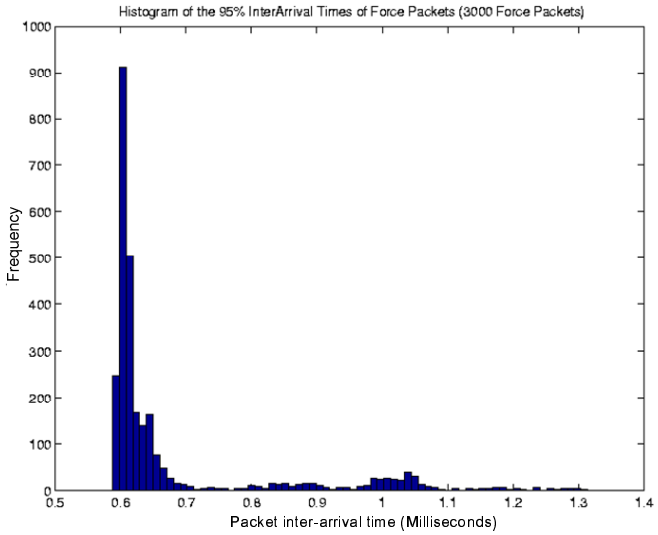


Figure 27: Distribution of force packet inter-arrival times

fits to an Inverse Gaussian distribution with a mean value of 0.679 ms and 90% of the data lying between 0.59 to 0.92 ms, i.e. a sampling rate of no less than 1 KHz.

Force and video multistreaming

In this case force thread alongside video thread is running on the server. A distribution of the the inter-arrival times of force packets in the presence of video transfer is shown in Figure 28. This is an Inverse Gaussian distribution with a mean value of 1.08 ms and 90% of the data lying between 0.5 and 3.9 ms. Clearly the presence of the video has pushed the mean value from 0.68 to 1.08 ms. As a result the sampling rate drops to about 250 Hz.

A magnified plot of the inter-arrival times of force packets in presence of video thread is shown in Figure 29. The pulse below the actual plot shows the interval during which the transfer of a stereo video frame was in progress. On the x-axis is the force packet number while on y-axis we have milliseconds.

To provide guaranteed performance to the force packet sampling rate we need to access the worst scenario in which we have intensive video transfer. Using the above data we isolate the instances during which video activities are intensive and study this effect on force packets. The distribution of force packet inter-arrival times of only those packets that were received during the transfer of a stereo video frame is shown in Figure 30. The data best fits to a Logistic distribution with a mean value of 5.41 ms and 90% confidence interval lying between 0.5 and 13.0 ms, i.e. a sampling rate of about 76 Hz. Clearly there is a large difference between the inter-arrival times of force packets without video, which is 0.679 ms and the case of intensive video activity during the transfer of a stereo video frame where it has a mean inter-arrival time of 5.41 ms.

The mean value of the inter-arrival times of stereo video frames is 87.57 ms with a 90% confidence interval falling between 72 and 107 ms. A distribution of the data for the video is shown in Figure 32.

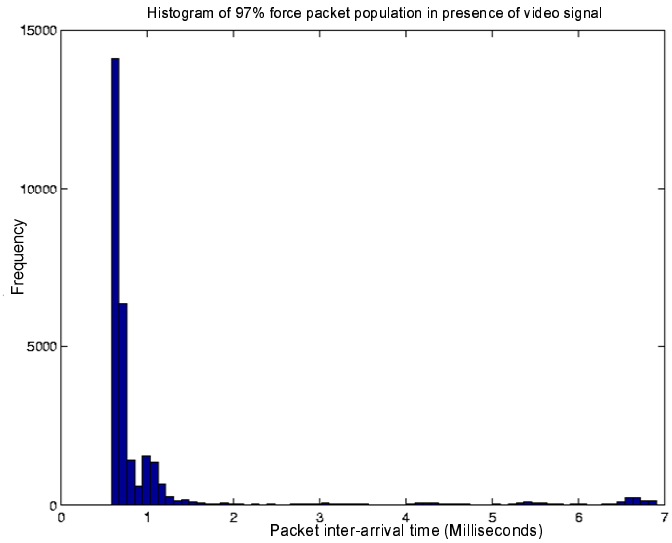


Figure 28: Distribution of force packet inter-arrival times with active video

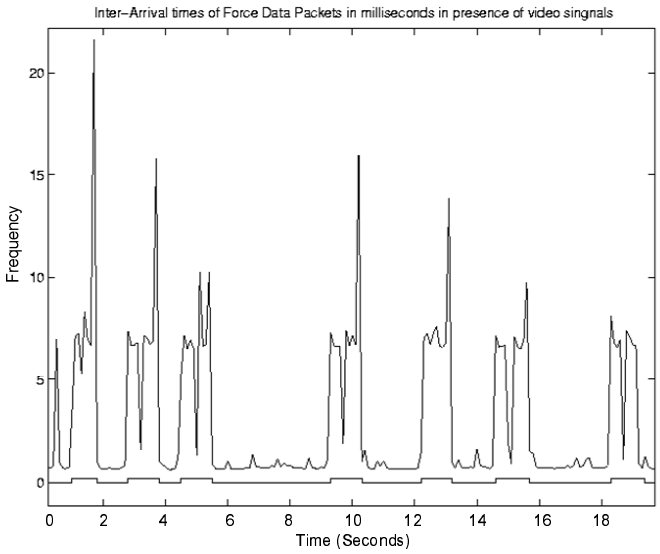


Figure 29: Magnified force packet inter-arrival times with active video

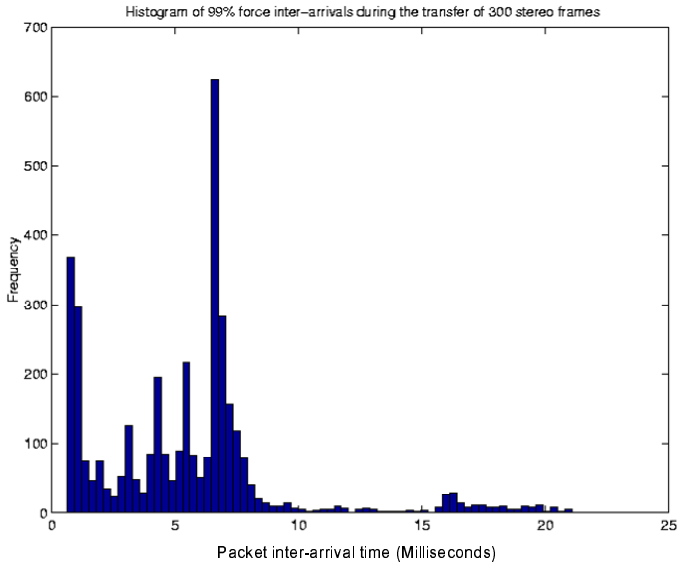


Figure 30: Distribution of inter-arrival times of force packets during the transfer of a video frame

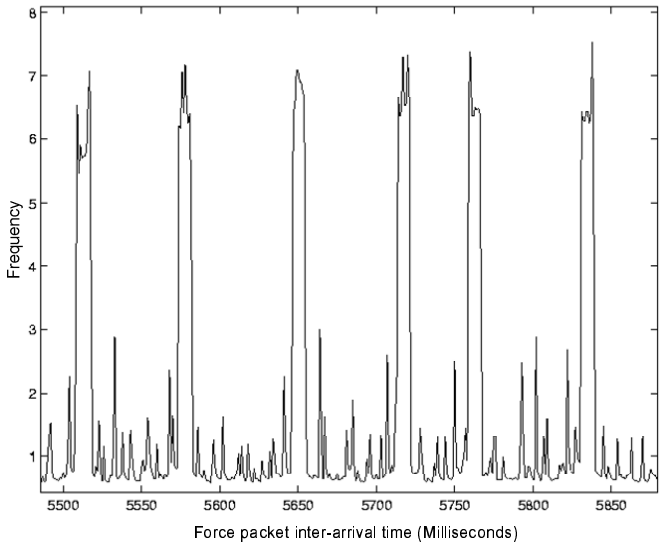


Figure 31: Magnified inter-arrival times of force packets in the presence of video and command threads

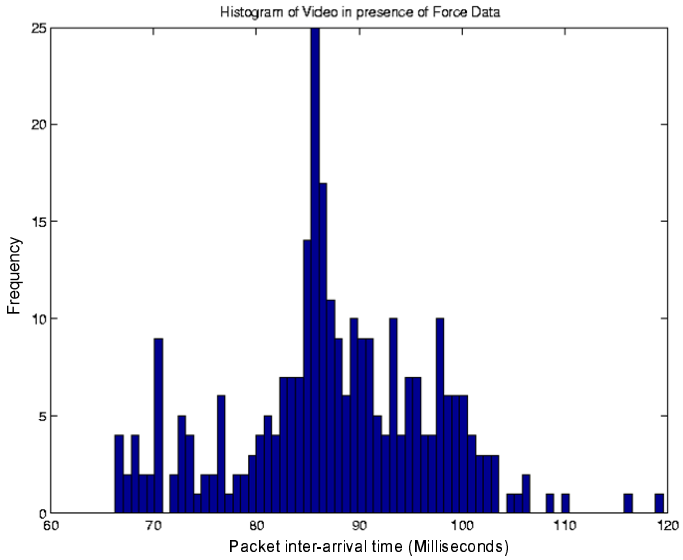


Figure 32: Distribution of inter-arrival times of video packets in the presence of force thread

Force, command, and video multistreaming

When all of the three i.e., force, command and video, threads are invoked simultaneously, we get a mean inter-arrival rate of 1.1 ms for the force packets, while 100% of the population remains under 8 ms. A magnified plot of the data against the force packet arrivals is given in Figure 31. Clearly the peaks in the plot show the effect of the transfer of video frames on the inter-arrival times of force packets.

Qualitative evaluation of force feedback

Force feedback [12, 64] is widely known to enhance performance in telerobotics. In telesurgery, providing force feedback display to the surgeon hand proved to reduce tissue trauma and to increase operation safety [11]. The proposed MTDF is used to provide bilateral coupling between a 6 dof PUMA 650 slave arm and a locally designed master arm with force display capabilities. The interconnection is studied within a LAN that is subject to other users traffic. The master arm is used to grab the operator hand motion as well as to display haptic feedback (force) due to its light and back-drivable structure.

The instrument held by the master arm interacts with the environment [2, 79]. The objective of displaying haptic information on the master arm is to let the operator feel the contact forces and react to them by generating motion corrections. Therefore the operator is part of a wide control loop including the remote environment as the target, the slave arm as a tool, the network as a mean of transmitting commands and feedback, the master arm as a tool to generate commands and display some feedback.

Our proposed MTDF is based on the force-reflecting teleoperation framework [10]. Analysis of contact forces as well as modelling issues based on damping and impact velocity is presented in [80]. To quantify the transparency of proposed MTDF we

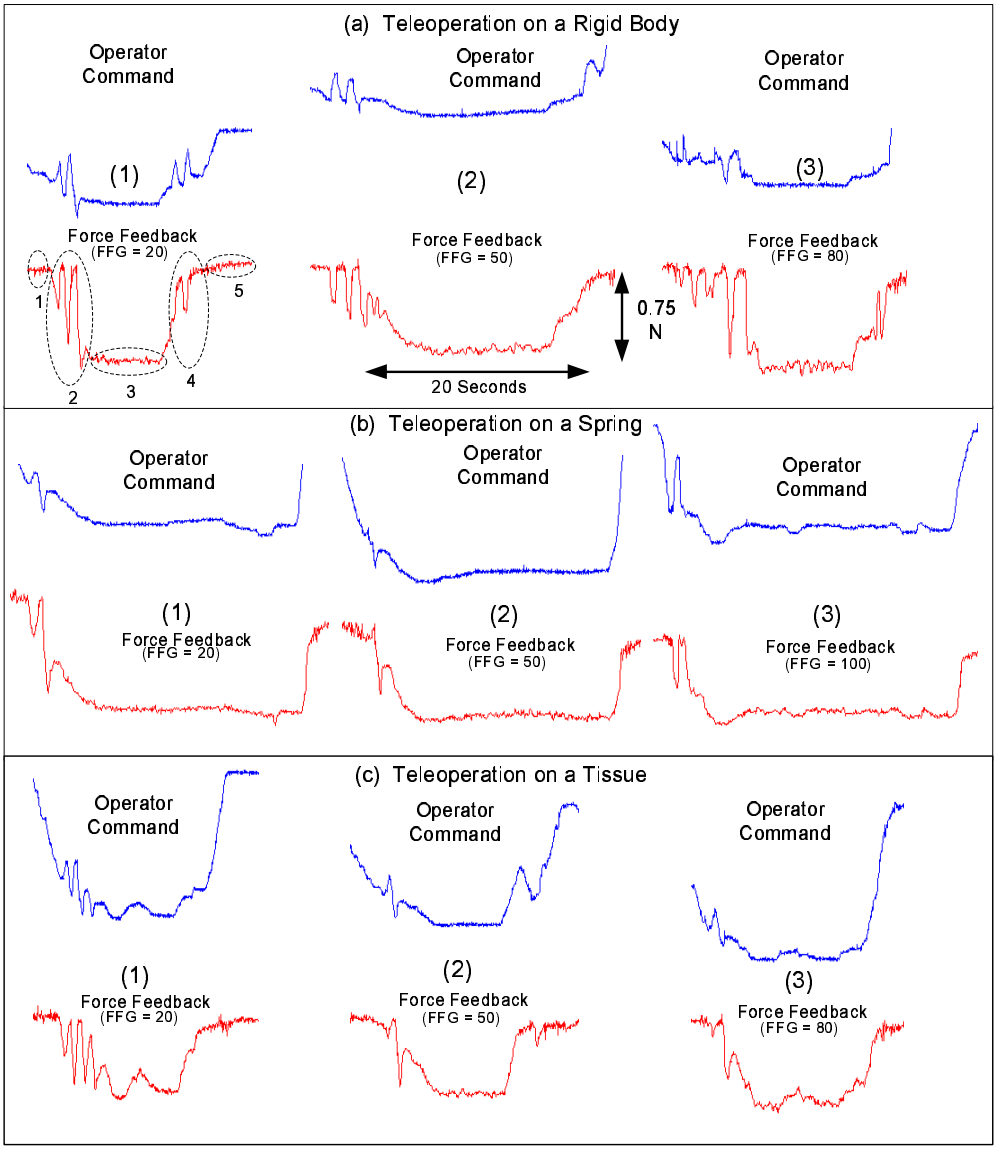


Figure 33: Operator commands and force feedback occurring during contact with a rigid body (a), a spring (b), and a tissue (c)

study the slave arm tool transition from free space to contact state. The objective is to minimize forces arising during contact between robot tool and objects.

The operator moves down the master arm along the vertical direction which causes similar motion in the slave arm tool. A real-time force data stream is transmitted from the server (slave robot) to the client (master arm). Figure 33 shows the force interaction occurring during contact between the slave arm tool and (1) a rigid body (case (a)), (2) a spring (case (b)), and (3) a human muscle tissue (case (c)). Following the contact, the operator was asked to maintain a constant force on the target. Each of the above three cases is further sub-divided into three experiments (from 1 to 3). Each experiment corresponds to a different setting of the force feedback gain (FFG) which is used to scale up and down the force feedback measured at the slave arm wrist. The force feedback received by the client is plotted before being scaled by the FFG coefficient. In each of the nine instances of Figure 33 the operator command is plotted in the upper part and the corresponding force feedback is plotted in the lower part. For a total duration of about 20 s the task of the operator is to bring the slave arm tool into contact with the target and to maintain on it a constant force of 0.75N.

Each contact operation has 5 phases which are (1) contact-free, (2) pre-contact, (3) contact, (4) pre-release, and (5) release. These are shown on Figure 33-(a)-1. For each given instance the operator receives no force feedback as far as the tool is still in free space. The pre-contact phase starts when the tool hits the target in the remote environment. We note that in both pre-contact and pre-release phases the teleoperation system is subject to vibrations which are displayed to the operator through the master arm after scaling it by factor FFG. The vibration frequency depends on the combination of two factors which are the stiffness of the target and the value of FFG. Stiff targets produce prompt bouncing contact forces and therefore produce higher vibration frequency. The vibrations for the rigid object are greater and faster than those of the spring or the tissue.

Similar effects are also observed for higher values of FFG as shown for each instance of figure parts (2) and (3). In the master arm both (1) the operator and (2) master arm motor can generate forces that control the dynamic of corresponding linkage (motor, wires, pulleys, and operator). The motor excitation is being the reflected force feedback.

Figure 34 shows an extended pre-contact periods for each of the above cases and material. The reason for the vibration is that when the operator is starting the pre-contact phase the first contact of the tool with target leads to (1) a force feedback that moves the motor in the opposite direction, (2) transmitting the force to operator through the dynamic of the linkage, and (3) producing a force bouncing (as the force feeling) from the operator hand back to the slave arm. This process continues to transmit contact forces from the scene and return a bouncing force from the operator until the elastic system between the target and operator hand is closed up by the operator engaging the slave closer and closer to target which completely amortize the above force bouncing. Note that the release phase is similar to the pre-contact phase. A high FFG gain may drive the telerobot out of control as shown in Figure 34-(a-1), (a-2), (b-1), and (b-2). Stable contact for the rigid and spring objects requires the use of lower FFG gains.

Note the difficulty caused by the visco-elastic nature of the tissue when the oper-

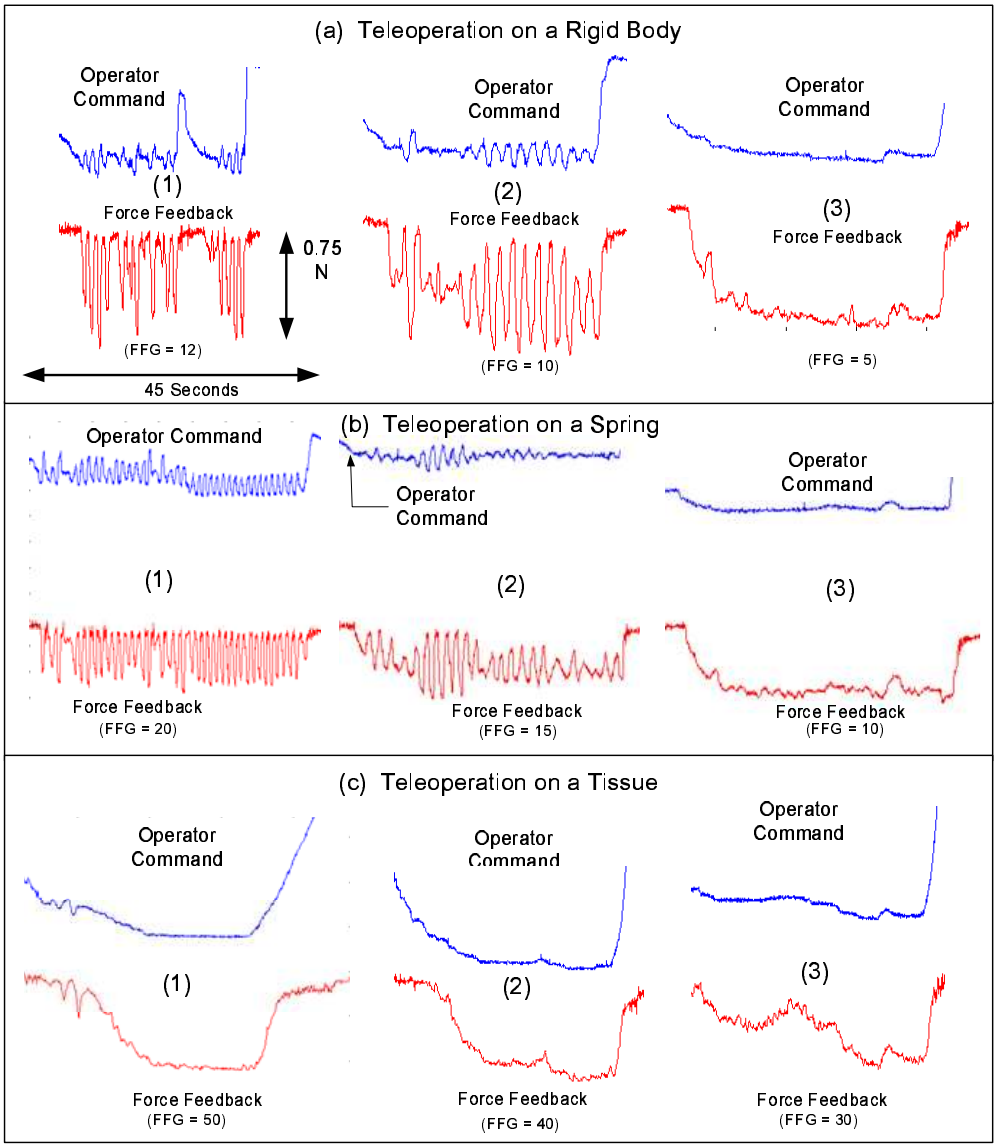


Figure 34: Extended command-force interactions of pre-contact for a rigid body (a), a spring (b), and a tissue (c)

ator attempted to maintain a constant force contact. The tissue shape deformations causes instabilities even in the contact state as shown on Figure 33-(c-1), -(c-3), and Figure 34-(c-2), and -(c-3)). The tissue is more subtil because its visco-elastic deformation may cause pre-contact phases to occur in the middle of a contact phase. Here the operator felt the reaction and attempted to retain back the contact state. There are other important reasons for instability in internet teleoperation which are due to transmission delays [64].

Generally the contact phase is characterized by a good fidelity of force feedback. The master arm can display a force of 20 N and the range of FFG values used are from 1 to 100. In other words, a force scalability factor of 1:100 has been observed in our setting. The operator has control of contact forces and succeeds in maintaining the target force for the desired time. The FFG can be controlled by the operator to reduce the possibility of teleoperation errors as well as to increase operation safety in telsurgery. However excessively high FFG values lead to unstable operations as described above. It was observed that at high force feedback gain (FFG) the elasticity feature of the spring was transmitted into physical constraints on the operator motion. Contact phases are all quite stable regardless of target stiffness when moderate FFG is used (1:20) except for tissue. For example pressing on the spring induced an opposing force on the operator hand giving the operator the feeling of a quasi spring. Teleoperation with weight can expose the operator to gravity effects if needed.

Througout the above experiments the proposed telerobotic framework proved to be effective in providing a reliable telerobotic system that can be used as a testbed to study man-machine-environment interaction as well as 3D visualtization.

Comparison to others

Teresa et al. [81] developed an internet based telerobotic system using JAVA and VRML. They used Java-based frame grabbing software to move an image from camera to DRAM as compared to our approach using DirectShow. The video transfer rate achieved by Teresa[81] is 1 frame every 3 seconds for a single image of 16 bit color depth over the internet. The Java-based frame grabbing software takes one second for an image to move from camera to DRAM as compared to a mean value of 24 ms obtained by our approach using DirectShow.

Yeuk et. al [60] developed a distributed infrastructure based on JAVA and DCOM. They used JAVA for database connectivity and path planner GUI(Graphic User Interface) and DCOM for network connectivity. MS VM(Microsoft Virual Machince) is proposed to bridge the gap between the two. However in our case, .NET framework is directly used for all GUI development as well as the core system components thus making it a unified solution. This frees us from using intermediatory services like MS VM within the framework.

Huosheng et. al.[82] proposed JAVA for network interfacing and video as well as the use of C++ for the robot controller for Internet-based telerobotic System. In a LAN setup, Huosheng et. al. quote a transfer rate of 9-12 fps with time delays less than 200 ms for a single image of size 200×150 pixels. This is to be noted that the images are not bitmap but are compressed using JPEG compression technique. In comparison to this, our stereo video client-server transfers *two uncompressed* images

(stereo frame) of size 288×360 pixels at a rate of 11.74 fps with a delay of around 87 ms only.

Al-Harthy[70, 83] implemented a client-server framework using VB 6.0 and TCP ActiveX controls. In the case of custom protocols like Al-Harthy's, the TCP read/write operations are very slow because of the many software layers involved such as Application, Custom protocol, TCP ActiveX control, and Windows Sockets etc. It takes 55 ms for a command signal (48 bytes) to reach from client to server. In our case a force packet consisting of 6 double values (6×8 bytes = 48 bytes, same size) took about 0.7 ms in the absence of stereo video data and 1.1 ms in the presence of video stream. While in a distributed setup, the components directly communicate with each other through windows sockets using .NET Remoting. This difference is achieved by using the distributed component based approach in place of TCP based custom protocols.

In a typical scenario when both client and server use .NET based components with TCP channels, highly optimized data transfer is obtained [84]. TCP Channel uses a default binary formatter which serializes the data in binary form and uses raw sockets to transmit data across the network. This method is ideal if the objects are deployed in a closed environment.

It is also worth noting that if the serialization of capturing and transferring-over-LAN threads is modified by thread manipulation on the server, an inter-arrival delay of around 55 ms can be achieved while utilizing nearly 90% of the bandwidth of a 100 Mbps LAN. By creating independent threads, instead of serialized processes in a single thread, in the process of capturing and data transfer, we are able to obtain a mean inter-arrival time of 58.57 ms.

6.4.5 Conclusion

In this section, a *Distributed Components based Telerobotic Framework* implementing a real-time interaction between a telerobotic client and server, is presented. To optimize delays in multi-streaming of force feedback, stereo data and master-slave commands, our approach uses tools that automatically handle the network resources and data transfer while isolating the components from network protocol issues. This liberates us from defining custom protocols for client-server interaction. Also this scheme provides flexible deployment environment in a sense that no pre-registration of components is required on the host machines which is a clear advantage over DCOM. In addition to providing a truly multi-threaded environment, the use of .NET components on both client and server sides guarantees fastest telerobotic interaction in a closed environment like a LAN. Statistical analysis of delays under different software scenarios enabled engineering the component distribution and promoting concurrency among service threads. The results of thread engineering and software optimization is an effective multistreaming running with a sampling rate of 17 Hz for stereo video, 76 Hz for force feedback, and 50 Hz for operator commands over a commodity 100 Mbps LAN. Thanks to multithreading for the graceful degradation of real-time force feedback data in periods of intensive video steaming. Using our telerobotic framework, we extensively analyzed the force feedback interaction of operator, master-slave arms, and environment. We qualitatively characterized teleoperated contacts with a rigid body, a spring, and a tissue.

Throughout the above experiments the proposed framework proved to be effective in providing a reliable telerobotic system that can be used as a testbed to enhance man-machine-environment interactions and 3D perception using advanced software and visualization techniques.

6.5 Real-time, multi-streaming, for stereo vision system

In this section we present performance evaluation of a real-time multi-streaming of Stereo and force data over a LAN. This system is concerned with the design of an effective Client-Server Framework for Stereo Image Acquisition, transfer over a LAN, and display at a remote station.

In a tele-operated environment, the operator needs to know the most recent situation at the server (or remote) side in order to make efficient manipulative decisions to control the robot. This information can be of more than one types, visualization being one of them. By this approach we provide the operator with a pictorial view of the remote side thus giving him a way to see the effect of his control commands.

Using stereo image techniques allows the operator to estimate the relative distances among the remote objects or to feel the depth of the scene. It has been shown in literature that these techniques greatly enhance the operator's efficiency during tele-manipulation. However, this allowance of stereo image on the client side imposes severe requirements in terms of bandwidth to transfer real-time stream of video data in a client-server environment. In addition it also requires the use of advanced technologies like DirectX and Windows Sockets to accomplish the capturing and relaying of video data over a LAN. Commercially available softwares like Microsoft NetMeeting are optimized for a low band-width network like internet so they show too poor display resolution to be used for stereo vision in a telerobotic setup.

Development of a highly optimized client-server framework for grabbing and relaying of a stereo video stream becomes inevitable keeping in view the above discussion. This framework must accomplish following tasks;

Server Side

1. Capture or grab stereo images from two cameras at the slave side simultaneously.
2. Establish a reliable client-server connection over a LAN, the slave side being the server.
3. Upon requests from the client send this stereo frame comprising of two pictures to the the client through windows sockets.

Client Side

1. Establish a highly optimized fast graphic display system to show the pictures received from the server.
2. Detect and establish the connection with server.
3. Display the pictures arrived from the server and continue in a loop each time asking a new stereo frame from the server.
4. Allow the viewer to adjust the alignment of the pictures on the output device, whatever it is, to compensate for the misalignment and non-linearities present in the stereo camera setup at server side.

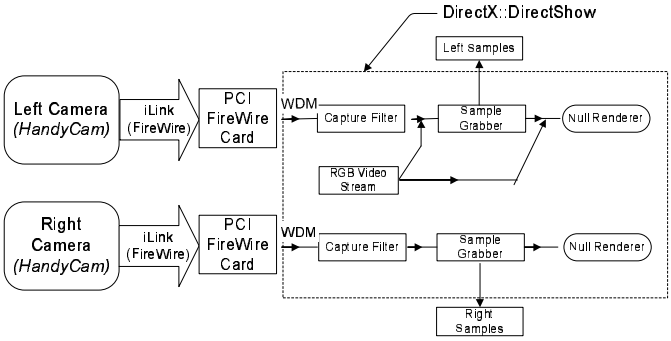


Figure 35: Block Diagram of Sample Grabber

A client-server framework fulfilling the above defined requirements is developed using the most advanced software development tools like Microsoft Visual C# and Microsoft DirectX. A detailed description of its functional and implementation details follows.

6.5.1 Functional details

The functional design of this distributed framework can be split into two parts: (1) server side, and (2) client side.

In the following we present the server implementation. Microsoft DirectX provides COM based interfaces for various graphice related functionalities. DirectShow is one of these services. DirectShow, further, provides efficient interfaces for the capturing and playback of video data. In our scheme we use a component of DirectShow named SampleGrabber to capture video frames coming through a stream from a stereo camera setup. A block diagram of the scheme used at the server side to grab stereo frames is shown in Figure 35.

Here the images from the left and right cameras are transferred to the PC using Sony's iLink interface which is based on IEEE 1394 serial bus standard (also known as FireWire) at a data rate reaching 400 Mbps. This stream is converted to WDM(Windows Digital Media) by a PCI card that hosts FireWire input ports for devices using FireWire standard. After that we hook capture filters provided by DirectShow to get hold of the video stream from the cameras. One we have video stream, the SampleGrabber is attached to capture the video samples from the stream. For termination purposes a null renderer is used to end the stream. If required, a renderer filter can be used to display the video on the primary output device.

In the following we present the client implementation. The graphical component of the Windows graphical environment is the graphics device interface (GDI). It communicates between the application and the device driver, which performs the hardware-specific functions that generate output. In order to show the received pictures from the server, we need to use GDI. A block diagram of the client side scheme to display the video is shown in Figure 36.

After receiving the video data from windows sockets, we use GDI functions to show the picture on the monitor screen.

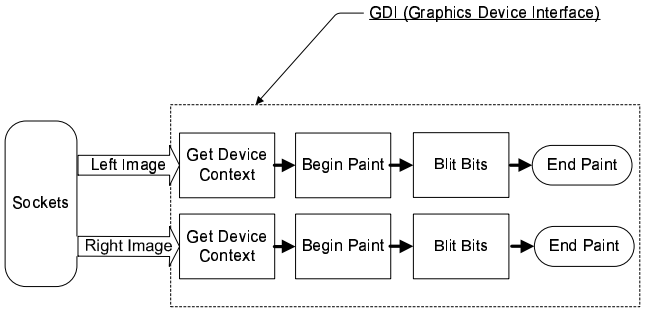


Figure 36: Displaying the Stereo Picture on Client Side

6.5.2 Implementation

In order to implement the above described client-server interface, we need a LAN to carry out the transfer of video data. In Windows environment, Sockets are used to program the network applications or in other words, we can use network services and send/receive data over a network using windows sockets.
Windows sockets are further subdivided into two major categories, known as (1) synchronous windows sockets and (2) asynchronous windows. Synchronous and asynchronous refer to whether a network call on the socket is blocking or non-blocking. The stereo video setup uses synchronous windows sockets as an interface between vision server and client. Two different schemes were implemented to transfer the video data. The schemes differ in the usage of multiple threads on the server side as well as some optimization steps to reduce the network traffic for the transfer of the data. Two implementations are presented: (1) single buffer with serialized transfer, and (2) double buffer, de-serialized transfer.

In the following we present the Single Buffer, Serialized Transfer, implementation. A detailed diagram of the implemented system for the transfer of stereo data is shown in Figure 37.

Both the client and the server side software are written in Visual C++ using MFC (Microsoft Foundation Classes). In the beginning the client as well as the server needs to be setup and each side has different steps to be taken in the startup phase.
On the server side the DirectShow environment is initialized and after that we connect the two video cameras to this environment by the scheme drawn in Figure 35. The SampleGrabber component of DirectShow uses a callback function to inform the completion of one video frame. In the stereo case we have two instances of SampleGrabber running at the same time to capture the video coming from two sources. Once the SampleGrabber executes this callback function, we can then copy this data supplied by SampleGrabber to some global memory buffer to be sent to the client through sockets. Microsoft does not recommend the sending of video data on to sockets directly from the callback function because it blocks the user interface of certain versions of Windows OS. After the hooking of callback function onto SampleGrabber, we initialize FilterGraph, another component of DirectShow, which starts the video capturing. The last step of server initialization is the setup of a server socket to send the video data over LAN. Once this initialization procedure

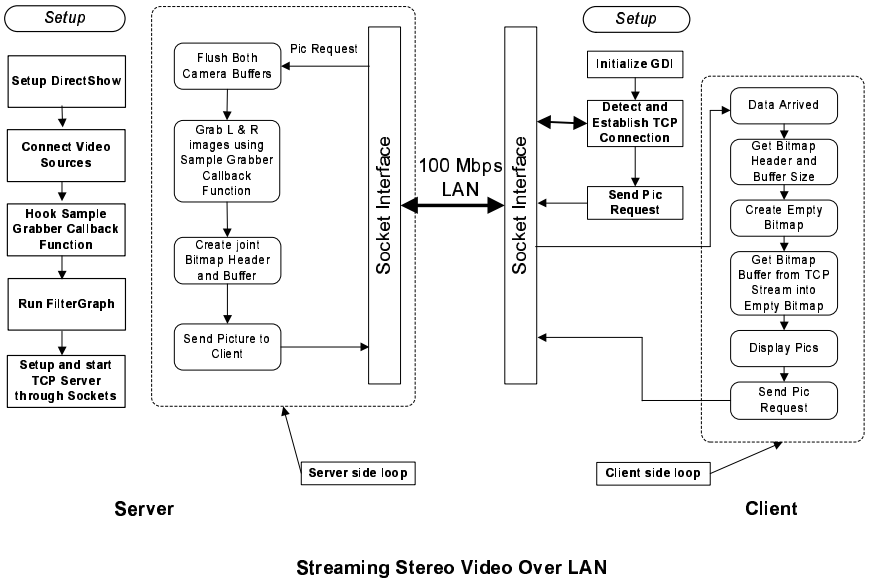


Figure 37: Streaming Stereo Video over LAN

is over , the server waits for a request of picture from client to initialize sending video data.
On the client side the initialization is a bit simple as we initialize GDI (Graphics Development Interface) to be able to draw the received pictures on the client screen. After GDI is initialized, the sockets are hooked to check the presence of the server on LAN, and if found, to issue a request for the picture to the server. This completes the initialization process on the client side.
After the client has sent a request for the picture to the server, both the client and the server enter respective local loops. The server side loop continues to receive the requests from the client, flush the previous bitmap buffers, grab left and right images using callback functions, create a Bitmap information header for these images and send it through the sockets over the LAN to the client.
The client side loop gets the buffer size from the TCP stream, prepares the bitmap buffer, receives the bitmap information header, copies the bitmap data from the sockets into the buffer, requests for new picture, draws the stereo picture on the screen to be viewed in 3D.

In the following we present the Double Buffer, De-Serialized Transfer, implementation. In this scheme, we try to optimize the transfer of video data over the LAN by using some thread manipulation on the server. Specifically speaking thread overlapping among capture and sending thread is achieved using double buffers on the server side. In this way, it is ensured that the thread responsible for sending the video data over the LAN will not wait after receiving a picture request from the client. A detailed diagram of the new scheme is shown in Figure 38.

By having a look at the figure, it is clear that the server side setup is not changed. Rather we have allocated two buffers, one for each stereo frame on the server. Every time a picture is received, the callback function of the respective camera is invoked. Once inside the callback function, it accesses a shared variable among multiple threads which indicates which buffer was copied to in the previous successful callback

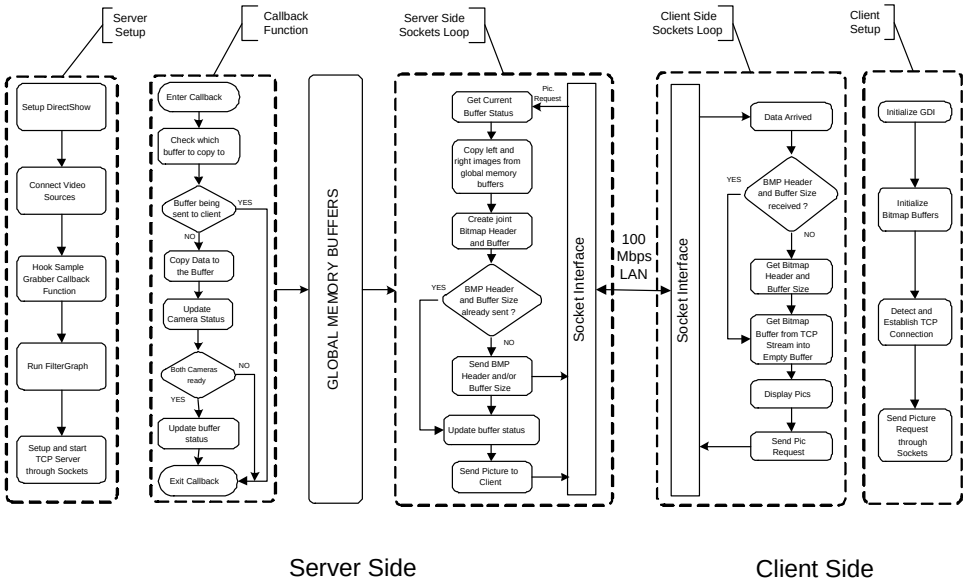


Figure 38: Streaming Stereo Video over LAN, Optimized Scheme

of this very camera. By the successful we mean that there are callback invocations in which no data will be copied to the memory buffer. An example of this case is the situation in which camera 1 copied data to buffer $b_{1,c1}$. Subscript $1, c1$ stands for 1st buffer out of double buffers and further that this portion of the buffer is related to camera 1. After the copying operation the sending thread accessed $b_{1,c1}$ and started sending data over LAN. In this duration, if camera 1 finished copying to the second buffer, i.e., $b_{2,c1}$, it will come back to $b_{1,c1}$ to write the next frame. But after accessing the buffer status variable, it will be denied access to this buffer as its transfer is still underway over the LAN. The camera will immediately return from the callback function. This will be attributed to an unsuccessful callback.
After copying the data to the buffer, it will further update the status of the camera. The status of the camera is required to synchronize the stereo frames for the left and right pictures. If both cameras are ready, it will update the buffer status which will enable the sending thread to send this buffer over to the client. In case the second camera has not finished copying the picture to the buffer, buffer status is not updated.
The sending thread is responsible for receiving requests from the client. After it receives a request, it will check the buffer status to determine which buffer should be sent. Once the proper stereo buffer is determined, it will create Bitmap headers and retrieved the buffer size. If these information have not already been sent to the client, they are sent. Otherwise, the server continues with the sending of buffer data only. The client proceeds in the same manner as with single buffer approach except that it does not receives the Bitmap information header and buffer size with each stereo frame. It retains the Bitmap Information Header and buffer size to properly display and read the required number of bytes from windows sockets.
This approach enables us to send higher number of stereo frames over the same LAN and hardware. The only overhead is the allocation of extra buffer in the server DRAM which not a real problem with available systems containing large memory.

6.5.3 3D visualization

There can be different methods to produce 3D effects on the client side once we have two stereo images of the remote scene. The following two methods are used extensively to accomplish this task: (1) Sync-Doubling, and (2) Page Flipping.

Sync-doubling does not requires any special device inside the computer. We only need to arrange the left and right eye images up and down on the computer screen. A sync-doubler sits between the display output from the PC and the monitor to insert an additional frame v-sync between the left and right frames (i.e. the top and bottom frames). This will allow the left and right eye images to appear in an interlaced pattern on screen. Using the frame v-sync as the shutter alternating sync allows us to synchronically transmit the right and left frames to respective left and right eyes, thus creating a three-dimensional image. This is the most effective 3D presentation method. It is not limited by the computer hardware specs as well as the capabilities of the monitor. However, sync-doubling is limited in a way that we get only half of the resolution of the screen for the 3D image.

Page-flipping means alternately showing the left and right eye images on the screen. Combining the 3D shuttering glasses with this type of 3D presentation only requires the application of frame v-sync as the shutter alternating sync to create a 3D image. Page-flipping requires higher hardware specifications.

- Since synchronized registration of left and right eye frames is necessary, the minimum capacity of its frame buffer is twice as usually required.
- In order to overcome the "flashing" problem of 3D imaging, frames provided should be at least 60 frames per second; hence v-scan frequency should be 120Hz or higher.
- As it involves hardware frame buffer and page-flipping synchronization, it often requires specially designed hardware for double-buffering the stereo image.

Page-flipping provides full resolution picture quality, hence it has the best visual effect among all available 3D display modes. But being highly dependent on software and hardware is the biggest drawback of this technique.

Because of the easy availability of sync-doubling shuttering glasses and minimal dependence on hardware, we have used sync-doubling technique as a provider of 3D visualization on the client side. A pair of stereo pictures is drawn on the client screen and the sync-doubler is hooked between the monitor and the VGA output. If the operator does not feel the 3D effects due to the camera calibration problems, a keyboard interface is given on the client side to move the pictures relative to each other to compensate for the camera setup problems until he gets a complete 3D view of the remote scene.

In the following we present the output devices used for 3D visualization. Mainly two types of devices are used for 3D vision systems, (1) shuttering glasses and (2) HMDs (Head Mounted Displays). The principle of the shuttering glasses is to show each eye a different images, i.e., left image to the left eye and right image to the right eye, by alternatively shuttering LCD glasses worn by the viewer. This way the human brain gets the illusion of viewing two different(stereo-scopic) views at the

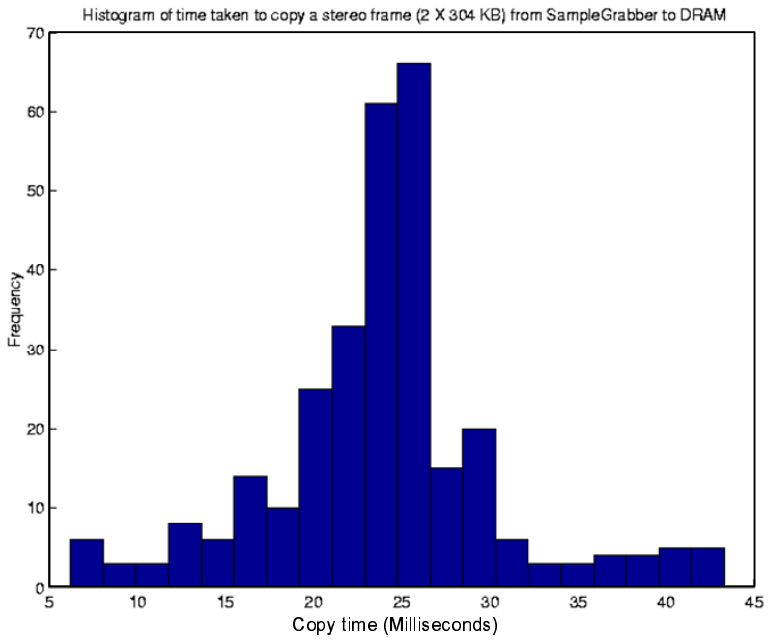


Figure 39: Histogram of copy times from SampleGrabber to DRAM

same time.
HMDs have a different approach. They are wearable displays and to each eye a separate LCD screen is shown. Some HMD display also come with head movement trackers to help aid in 3D orientation.
We use eye shuttering glasses with a display resolution of 1024 by 768 pixels and refresh rate of 85 hertz which is doubled to 170 hertz by the sync-doubler. The stereo image resolution attained is 288 by 360 pixels. A resolution of 384 by 512 pixels can be achieved with current monitor settings but this will introduce more load on network traffic thus decreasing the frames per second and in turn increasing the inter-arrival times.

6.5.4 Performance evaluation

Different experiments were conducted to test the visual quality of the client-server setup as well as find the time delays and other measures of the video data. The validity of the data obtained during the experiments was verified by conducting several experiments with same configuration and by checking the differences in the results.
The specifications of the stereo frame are as under:

Height of each picture = 288 pixels
Width of each picture = 360 pixels
Size = 304 KB (311040 Bytes) per picture
= 608 KB (622080 Bytes) per stereo frame

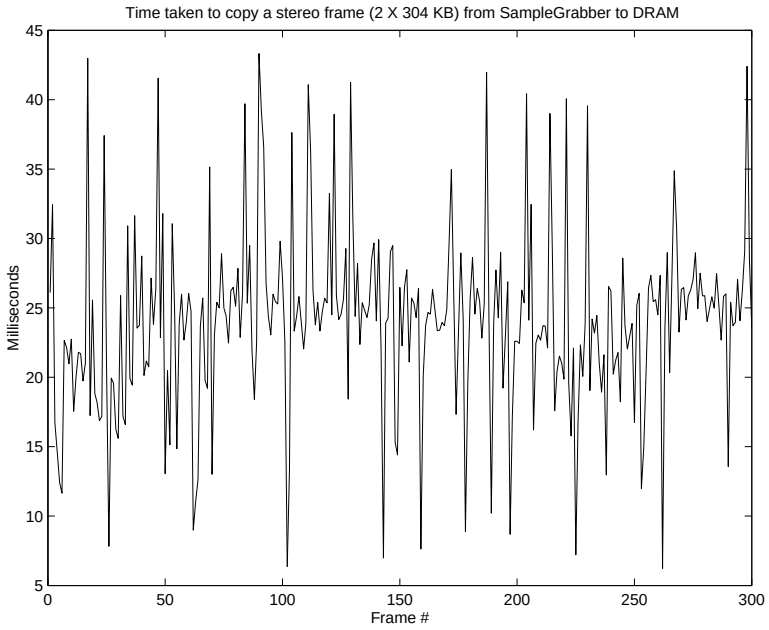


Figure 40: Plot of copy times from SampleGrabber to DRAM

So each stereo frame is of size 0.6 MB and requires a bandwidth of 5Mbps/Frame on the LAN. This simple calculation shows the limitation of the 100 Mbps LAN to transfer only 20 fps at the highest possible transfer rate.

Copying from SampleGrabber to DRAM

First we consider the server side to find out the time to copy one stereo frame from the SampleGrabber to the DRAM. A high precision counter was used to count the cpu ticks between the start and end of the frame copy.

Case 1: Copy times on server - Single Thread:
A histogram of the data obtained during the transfer of 300 stereo frames is shown in Figure 39. The mean value of 24.025 ms is clearly visible in the normal distribution of the data. 95% confidence interval falls between limits of 23.29 and 24.75 ms.

A plot of the time taken by each from for all 300 frames is shown in Figure 40. There are some disturbances in the beginning of the capture but soon it settles to a mean value. These disturbances could be attributed to the initialization routines at the startup of the capturing and allocation of memory buffers. During the experiments, the local display at the server is disabled which is more than 30 fps if enabled.

Case 2: Copy times on server - Two Threads:
In this case, we trigger another thread to read force information from the sensor. In this thread we try to read force as fast as possible. Each time a force packet is received from the sensor, an event is invoked. We do not transfer this force over LAN. Rather the we observe the effect of an additional thread on the copy times of video data from SampleGrabber to DRAM. This helps us to evaluate the performance of a multi-threaded environment.
The histogram in Figure 41 clearly shows the the data now follows a Beta distribution

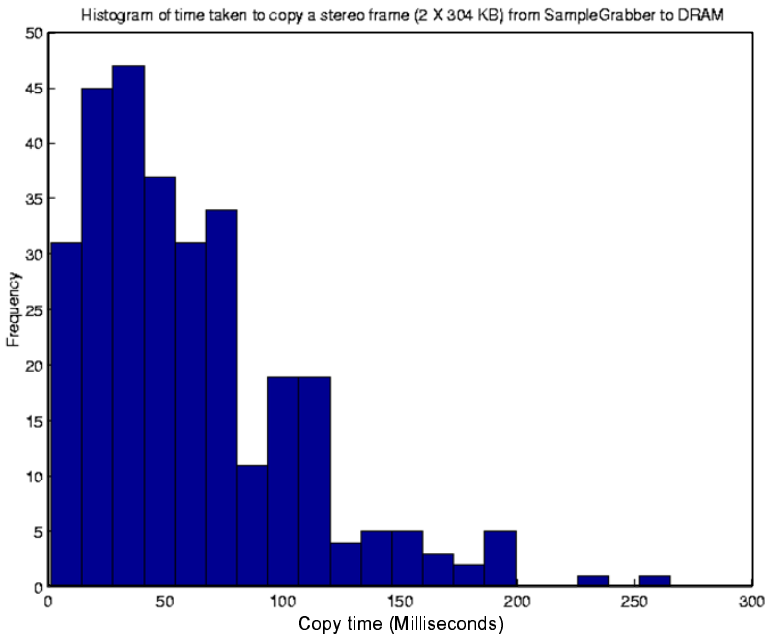


Figure 41: Histogram of copy times from SampleGrabber to DRAM in the presence of a force thread on the server

instead of a clear normal distribution in the previous case. Also the addition of a new force thread on the server has caused the mean value to jump to 60.48 ms from 24.025 ms in the previous case. Simple statistical data analysis shows that 90% of the data lies between 8 and 150 ms. So it is clear that the inclusion of the force thread to active threads on the server affects the process reasonably.

Simple plot of the data for two thread case is shown in Figure 42.

Case 3: Copy times on server with Force transfer over LAN:
In this setup, the force information is also transferred to the client side. So the server is running more than two threads and the socket function that transfers the force information from the client to the server is a blocking one, i.e., the force thread is blocked until the force data is sent completely to the client side.
The histogram in Figure 43 shows the results of this setup. This figure shows that the data follows Gumbel distribution (a special case of Weibull distribution) and the mean value for the data is 33.46 ms. This decrease in the copying time can be explained by the fact that the force transfer is a blocking operation so the time, during which force thread is blocked, is utilized by the video data copying routine thus decreasing the copy time from 60.48 ms to 33.46 ms. More clearly speaking, the addition of a force transfer thread causes an addition of 9.43 ms (33.46 - 24.025) of delay in copying time of a stereo frame from SampleGrabber to DRAM on server side.

Simple plot of the data for this case of video transfer in the presence of force transfer thread is shown in Figure 44.

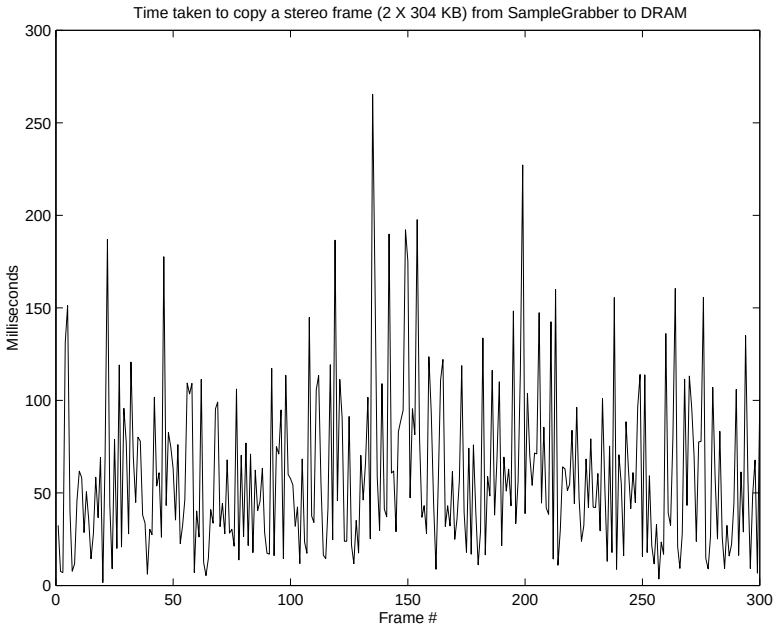


Figure 42: Plot of copy times from SampleGrabber to DRAM in the presence of a force thread on the server

Transferring over the LAN

In this part we deal with the performance issues related to the transfer of stereo image over a LAN. The experiments were carried out in a single lab on client and server PCs. They are connected by a 100 Mbps ethernet.

Case 1: Single Buffer, Serialized Transfer:
In this configuration we use the scheme shown in Figure 37 using single buffer on the server side. The sending thread waits for the two SampleGrabbers to write stereo frame data to global buffer in order to send it over the LAN.
Figure 45 shows the histogram of inter-arrival times of 300 stereo frames.

This clearly is a Gaussian distribution with a mean value of inter-arrival times equal to 86.5 ms which shows a stereo frame rate of 11.6 frames per second. A plot of inter-arrival times for the transfer of 300 stereo frames over LAN is shown in Figure 46.

Case 2: Double Buffer, De-Serialized Transfer:
In this case, the performance of the optimized client-server setup shown in Figure 38 is evaluated. Figure 47 shows the histogram of inter-arrival times of 50,000 stereo frames transferred between client and server while the display on both clients and server is disabled which is also applicable to the case with single buffer, serialized transfer experiments.

Statistically this is a Gumbel distribution with a mean value of 58.94 ms and 90% of the data lying between 56.0 and 64.8 ms. This gives us a transfer rate of 17 fps. The maximum delay observed is 1298.6 ms which obviously is coming from network congestion and the minimum value is 53.4 ms. A plot of inter-arrival times for the same data is shown in Figure 48.

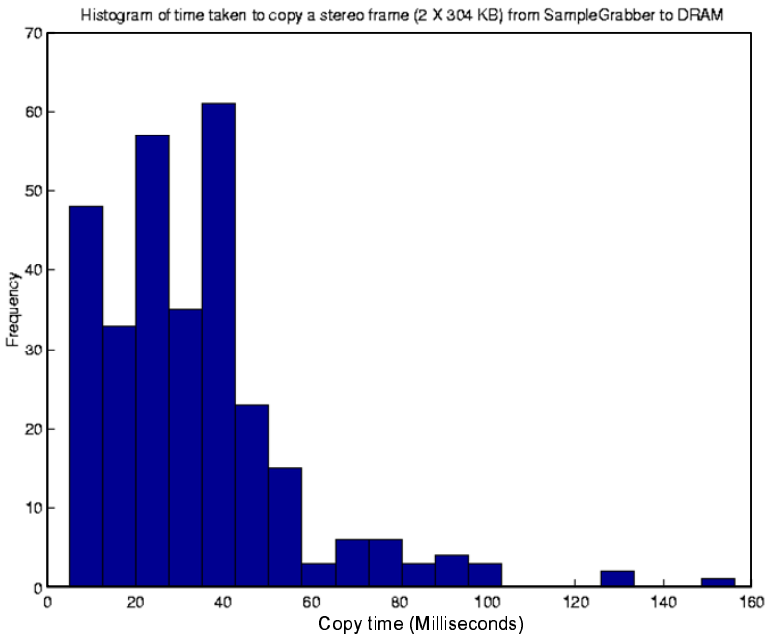


Figure 43: Histogram of copy times from SampleGrabber to DRAM in the presence of force transfer over LAN

Clearly this setup is giving much better results than the previous one with single buffer. The mean value has decreased from 86.5 ms to just 58.94 ms giving us a gain of 27 ms. This clearly is the copying time of one stereo frame on the server side (24 ms) plus additional time saved that was being used in activating the SampleGrabbers and receiving the buffer ready notification from them.
A frame rate greater than 10 fps gives good viewing experience and refresh rate of 85 hertz eliminates any flickering. The viewer never feels headache because of high refresh rate. Some simple manipulation experiments, to move objects by looking at 3D scene on the computer screen wearing shuttering glasses, showed good depth perception of the viewer.

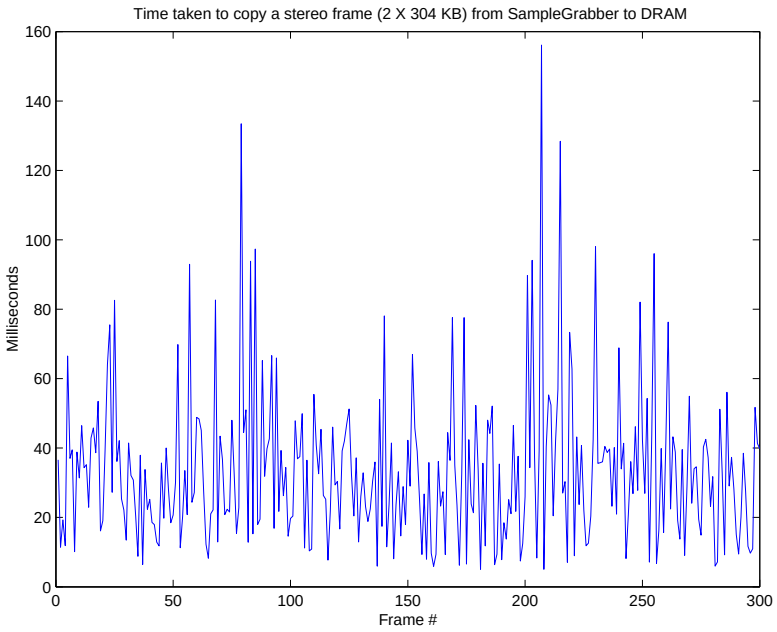


Figure 44: Plot of copy times from SampleGrabber to DRAM in the presence of force transfer over LAN

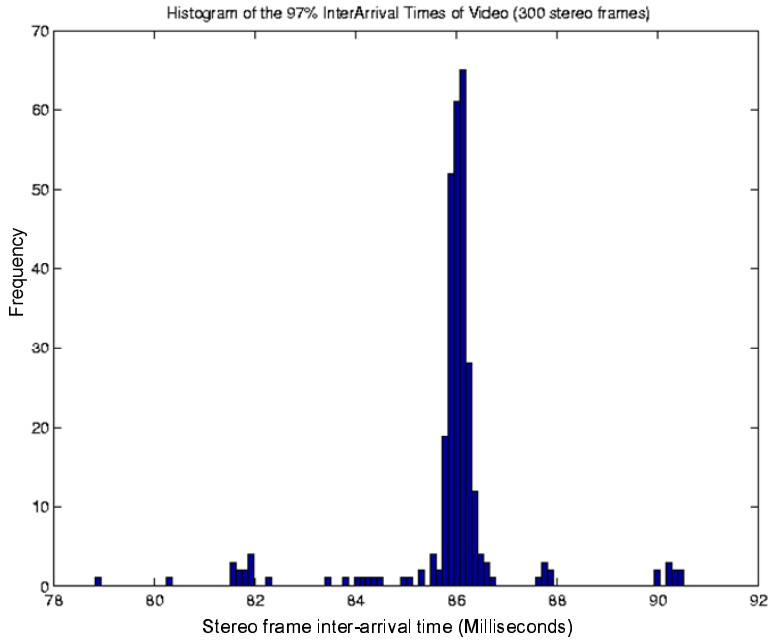


Figure 45: Histogram of inter-arrival times of stereo frames on client side

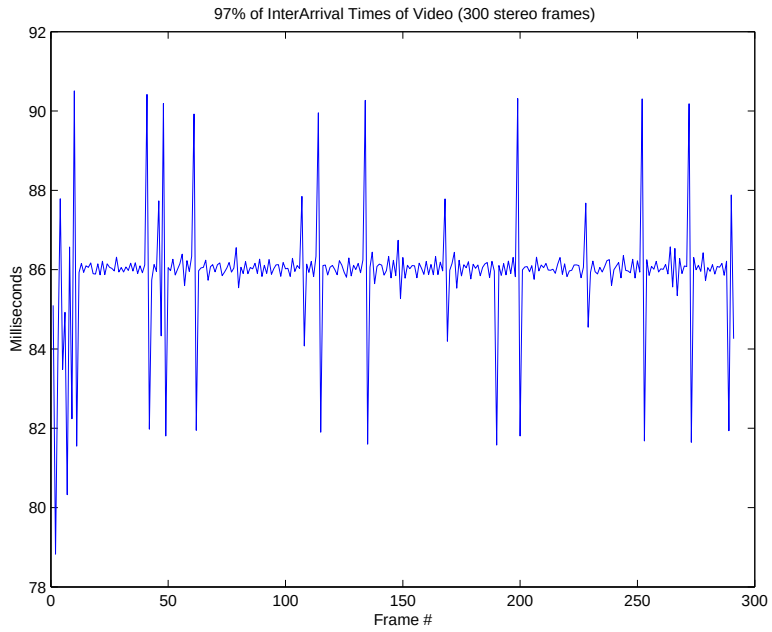


Figure 46: Plot of inter-arrival times of stereo frames on client side

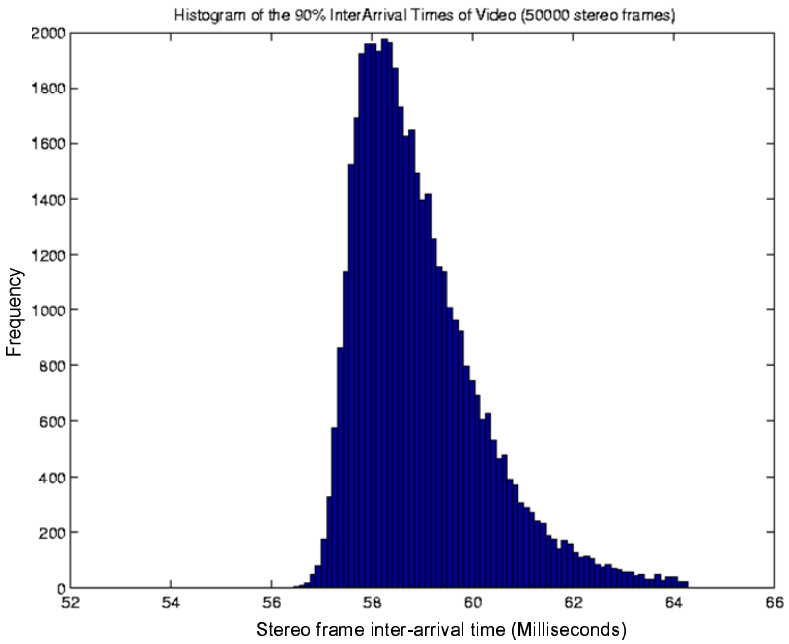


Figure 47: Histogram of inter-arrival times of stereo frames on client side

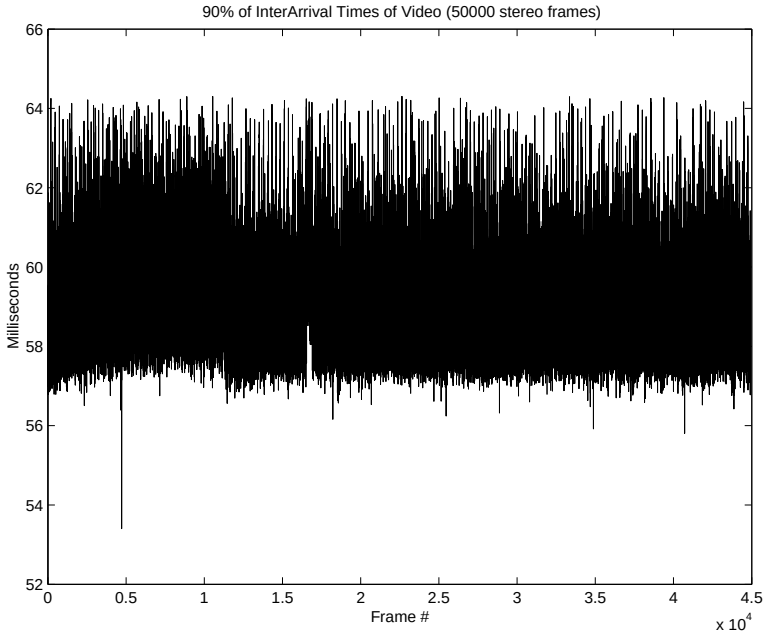


Figure 48: Plot of inter-arrival times of stereo frames on client side

6.6 An augmented reality system for telerobotics

Augmented Reality can be used as an effective way to overcome the effects of time delays in a telerobotic environment. The basic idea of an augmented reality system is to mix the real and virtual information in order to provide the operator an augmented view of the remote scene combined with a virtual representation of his own local actions. This allows the operator to see how his action would fit into the scene before being executed. The information that is added locally must fit seamlessly into the remote real data so as to avoid any perplexities for the tele-operator. The method that is generally adopted to augment a video stream uses overlaying virtual graphics over real images.

Milgram et. al. [68] state three primary purposes for overlaying graphics for teleoperation applications: 1) as a tool for probing the real remote environment visible on video, 2) for enhancing video images through real object overlays, thus compensating for image degradation due to occlusion of objects, and 3) for introducing realistic looking but non-existent graphic objects so that they appear to be a part of the video scene. The last approach will be followed in the present work with an aim to show the present location of the gripper point on the local video display in the absence of fresh video data. To accomplish this task, it is proposed that a small ball should be inserted in the most recent video scene at the position of the gripper which is calculated locally from the command data coming from master arm, using the direct geometric model of the robot. This should indicate the location of gripper one step ahead of time thus providing the operator a way to view the results of his commands before the arrival of relevant video data.

Overlaying the graphics on real video, however, requires that a bidirectional one-to-one mapping of coordinate spaces between the virtual world and the remote world viewed through the video is established. For a stereo video system, this requires the respective mappings of both right and left video frames. Simply stated, we must know where a point in virtual 3D world will be projected on real stereo video. This requires the knowledge of how some known fiducial points in 3D world are projected on 2D image plane (pixel array).

6.6.1 Notations

Following notations will be used in the text to follow:

f	=	Focal length of the camera
P_i	=	Any point in 3D space
P_0	=	Origin in 3D frame of reference
P_1 to P_3	=	Reference points constituting 3D frame of reference
P_{ix}, P_{iy}, P_{iz}	=	X, Y and Z coordinates of a point P_i in 3D space, using world coordinates unless specified otherwise
p_i	=	Projection of a point P_i in pixel coordinates
p_{ix}, p_{iy}	=	Pixel coordinates of the projection of a point P_i
p_{ij}	=	Elements of the projection matrix
M	=	Overall projection matrix
M_l	=	Left projection matrix
M_r	=	Right projection matrix

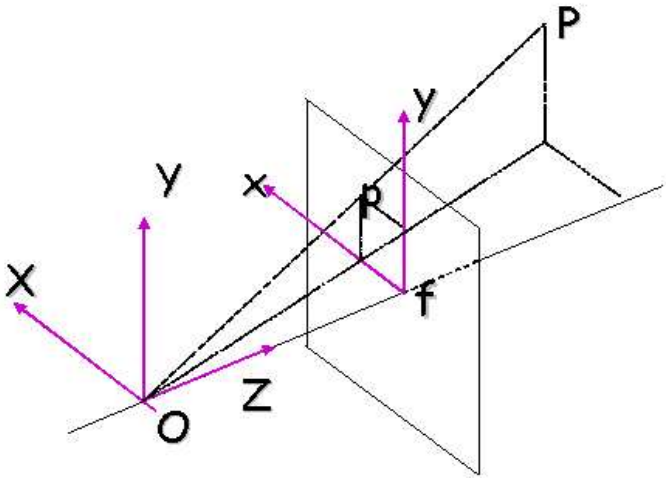


Figure 49: Pinhole camera

6.6.2 Camera model

A camera model is used to project 3D points on 2D image plane. The full perspective transformation between world and image coordinates is conventionally analyzed using the pinhole camera model with the following non-linear equations for a point $P_i(X, Y, Z)$ in world coordinates where camera is placed at the origin of world reference frame. The relative positions of camera, point P and image plane are shown in the figure.

$$\begin{bmatrix} x_{cam} \\ y_{cam} \end{bmatrix} = \begin{bmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{bmatrix} \tag{294}$$

In homogeneous coordinates the pinhole projection is given as:

$$\begin{bmatrix} x'_{cam} \\ y'_{cam} \\ z'_{cam} \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{295}$$

where

$$x_{cam} = \frac{x'_{cam}}{z'_{cam}}$$
$$y_{cam} = \frac{y'_{cam}}{z'_{cam}}$$

Or,

$$\begin{bmatrix} x_{cam} \\ y_{cam} \\ f \end{bmatrix} = f/Z \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{296}$$

The term f/Z is generally known as scale factor.

If the camera and world coordinate frames are different, which is generally the case, a transformation from world to camera coordinates is also needed. The general

form of the projection from 3D world to camera surface is given as:

$$\begin{bmatrix} x_{cam} \\ y_{cam} \\ 1 \end{bmatrix} = \begin{bmatrix} f/Z & 0 & 0 & 0 \\ 0 & f/Z & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{297}$$

Or,

$$p_{cam} = K * T * P_i \tag{298}$$

where P_i is a point in world coordinates, T is a transformation matrix from world to camera coordinates and K projects the points from camera coordinates to image plane using a scale factor of f/Z. t_x, t_y and t_z form the translation vector while r_{xx} are the elements of a rotation matrix, both from world to camera coordinates.

Equation 297 can be written in more compact form as:

$$\begin{bmatrix} x_{cam} \\ y_{cam} \end{bmatrix} = f/Z * \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{299}$$

In practical computer vision application we find environments where the depth of scene is comparably small as compared to average distance of camera from the objects i.e. $\delta Z << Z_0$ where δZ is the depth of scene and Z_0 is average distance of camera from the objects.

In such cases a linear approximation to the model given in equation 299, is used that is called 'weak perspective projection'. In this setting we assume that all the points in the scene are at an average depth from the camera. Weak perspective projection is given as:

$$\begin{bmatrix} x_{cam} \\ y_{cam} \end{bmatrix} = f/Z_0 * \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{300}$$

When small objects are viewed from more than one meter distance, as in normal laboratory experiments, weak perspective projection gives reasonable results.

The relationship between image plane coordinates (x_{cam}, y_{cam}) and their pixel addresses (u, v) can be modelled by an affine transformation representing offsets, scaling, etc. [85], and the entire projection, in homogeneous form, can be written as a linear mapping:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{301}$$

If we describe the origin offsets separately, the projection of a point P_i in 3D space to a point p_i onto the pixel surface is given as:

$$\begin{bmatrix} p_{ix} \\ p_{iy} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \end{bmatrix} \begin{bmatrix} P_{ix} \\ P_{iy} \\ P_{iz} \end{bmatrix} + \begin{bmatrix} p_{0x} \\ p_{0y} \end{bmatrix} \tag{302}$$

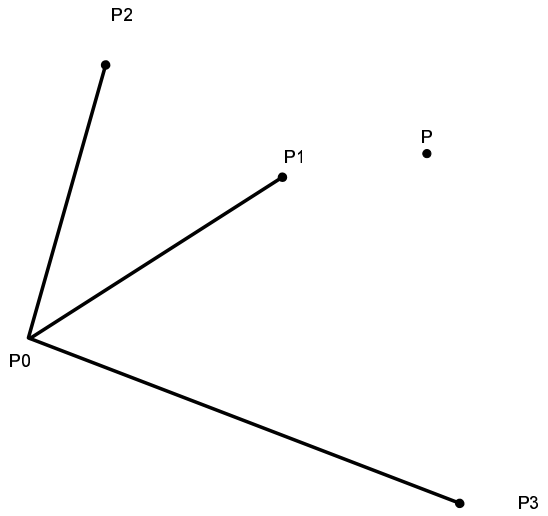


Figure 50: Affine reference frame

This will be discussed in more detail in section 6.6.3. Equation 302 can also be written as:

$$p_i = M * P_i + p_0 \tag{303}$$

For a stereo computer vision system, we need two projection matrices, one for each left and right images. Throughout our augmented reality applications, we will use the weak perspective camera model given in equation 302.

6.6.3 Camera identification

Accurate projection of virtual objects onto a video stream requires the knowledge of camera that is used to capture the video of real environment. In order to use the model given in equation 302, we must find the projection matrix M of the camera.

Projection matrix M can be calculated by finding the projections of four non-coplanar points in the pixel coordinates. These four points constitute the affine frame of reference that can serve as a basis for all other points present in the scene. For illustration see figure 50.

In this figure points P_1, P_2 and P_3 constitute the basis vector with P_0 as origin. Any point $P_i(X, Y, Z)$ can be described with respect to this frame of reference. Because the weak perspective projection is an affine transformation, the same relationship will remain valid in the projected basis vector and any other scene points.

If the projections of points P_0 to P_4 are known as well as their 3D coordinates, we can generate the following set of six linear equations using expression 302.

$$\begin{aligned}
p_{11}P_{1x} + p_{12}P_{1y} + p_{13}P_{1z} + p_{ox} &= p_{1x} && (304)\\
p_{21}P_{1x} + p_{22}P_{1y} + p_{23}P_{1z} + p_{oy} &= p_{1y} && (305)\\
p_{11}P_{2x} + p_{12}P_{2y} + p_{13}P_{2z} + p_{ox} &= p_{2x} && (306)\\
p_{21}P_{2x} + p_{22}P_{2y} + p_{23}P_{2z} + p_{oy} &= p_{2y} && (307)\\
p_{11}P_{3x} + p_{12}P_{3y} + p_{13}P_{3z} + p_{ox} &= p_{3x} && (308)\\
p_{21}P_{3x} + p_{22}P_{3y} + p_{23}P_{3z} + p_{oy} &= p_{3y} && (309)
\end{aligned}$$

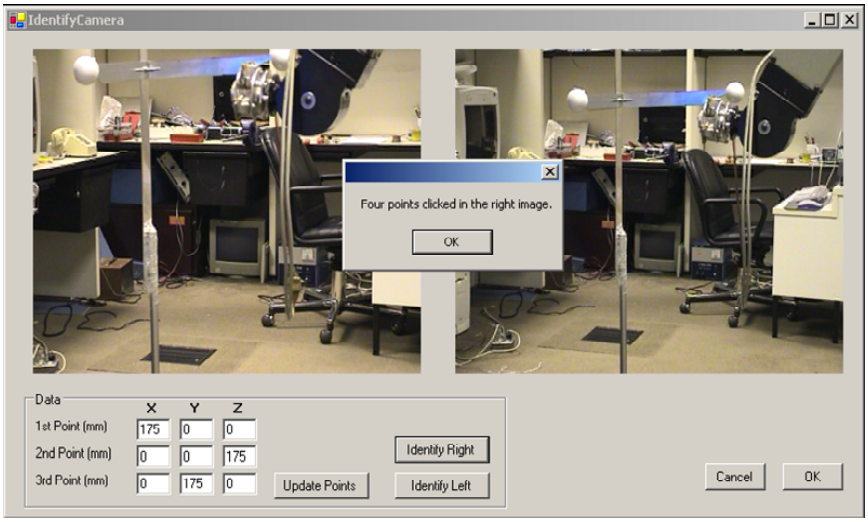


Figure 51: Camera identification GUI

where P_0 is the origin of the affine frame of reference. In matrix form, this system can be written as:

$$\begin{bmatrix} P_{1x} & P_{1y} & P_{1z} & 0 & 0 & 0 \\ 0 & 0 & 0 & P_{1x} & P_{1y} & P_{1z} \\ P_{2x} & P_{2y} & P_{2z} & 0 & 0 & 0 \\ 0 & 0 & 0 & P_{2x} & P_{2y} & P_{2z} \\ P_{3x} & P_{3y} & P_{3z} & 0 & 0 & 0 \\ 0 & 0 & 0 & P_{3x} & P_{3y} & P_{3z} \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{21} \\ p_{22} \\ p_{23} \end{bmatrix} = \begin{bmatrix} p_{1x} \\ p_{1y} \\ p_{2x} \\ p_{2y} \\ p_{3x} \\ p_{3y} \end{bmatrix} \tag{310}$$

or,

$$AX = B \tag{311}$$

Solving this system of linear equations $X = A^{-1}B$ can give us the projection matrix M of the camera. The inverse of A must exist if the 4 reference points are non coplanar. For the stereo projections, two sets of equation 310 must be required to be solved for left and right projection matrices.

A graphic user interface was designed to help the user select the projections of the mentioned points by clicking with mouse on the respective pixels. A snapshot of the GUI is given in figure 51. The user can either choose the default locations of the fiducial points, or he can enter the new 3D locations of the same if they have changed since last setup.

Once the 3D positions of the points are entered in the appropriate text boxes, the user can start the camera identification by pressing either of the *Identify Left* and *Identify Right* buttons. After he clicks either of the button, the system asks him to click the four points forming the basis in the respective image following a certain order while clicking. The locations of these points are stored as the pixel coordinates of the projections of fiducial points. Both the right as well as left projections of these points are recorded before closing this identification form.

After the provision of all necessary data, the program solves the matrix equation 310 to find out M_l and M_r matrices. In order to speed up the process of solving the

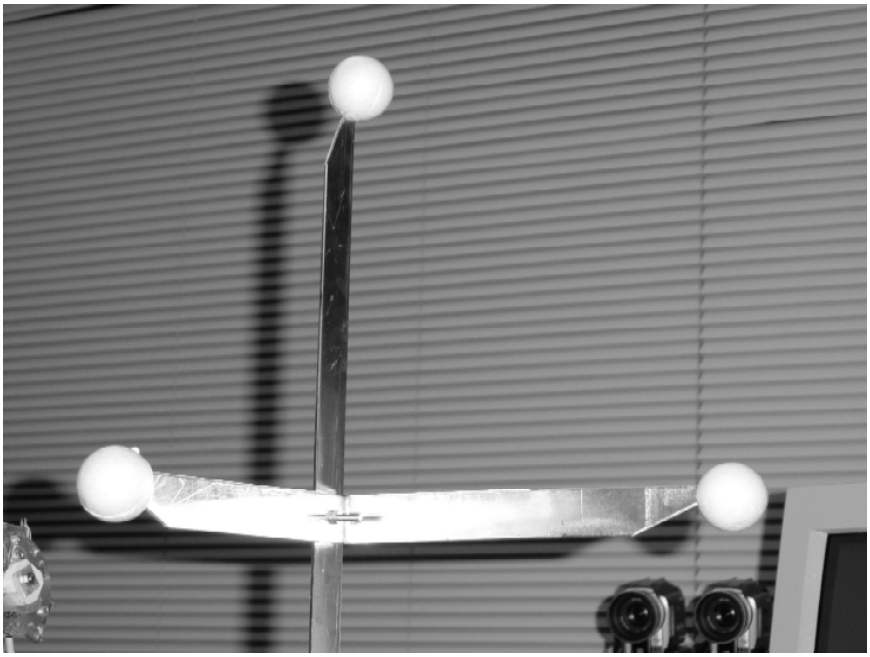


Figure 52: Reference frame

system of linear equations, an analytical solution of the system was developed and is evaluated by just substituting the values recorded by the user.

Setting-up server side

Server side setup requires the positioning of a reference frame with four points in space whose 3D locations are known. This reference frame should be as close to the object of interest as possible in order to avoid any non-linear behavior of the weak-perspective projection. There is no need for these points to be always present in the scene. They are required only during the identification phase. Once the identification is done, the physical reference frame can be removed from the workspace.

Any point in the 3D scene will be described with respect to the origin of this frame of reference. Because there is a down scaling of the objects on image plane, the fiducial points should be dispersed at a reasonable distance otherwise their projections on the pixel surface will be too close to yield any good values for the projection matrix. In our experiment we have used a distance of 20 cm for each point from the origin. A view of the frame is given in figure 52. Similarly the camera should be placed as far from the scene as possible, usually more than 1.5 meters for reasonable approximation to perspective camera model.

6.6.4 DirectX API

The image data retrieved from the *StereoSocketClient* component comes in a memory stream according to bitmap format. This stereo image is then displayed to the tele-operator using the *DXInterface* component, and HMD(Head Mounted Display) controller. Because the *DXInterface* heavily depends on *DirectX* API (Application

Programming Interface), so a brief overview of it will be helpful in understanding the subject matter.

Microsoft DirectX is a set of low-level application programming interfaces (APIs) for creating high-performance multimedia applications. It includes support for two-dimensional (2D) and three-dimensional (3D) graphics, sound effects, input devices, and networked applications [84], [86].

Surfaces

The *DirectX Surface* can be thought of as a piece of paper that you can draw on. You must specify the dimensions, and color pellet while creating a surface. By default *DirectX* will try to create the surface in accelerated video memory on video cared but if there is not enough room to create the surface in this memory, it creates the surface in system memory. The primary surface is the the pixel array that is visible on the output video device. This is always on the video card if it has enough memory.

There is only one primary surface per *DirectX* device. However you can create off-screen surfaces for other purposes, like drawing and blitting, etc. Again, the off-screen surfaces should ideally be created in the accelerated graphics memory for minimum system delays. A pointer to the primary surface can be attained by asking for a *BackBuffer* from the *DirectX Device*. It is typical to have a *BackBuffer* for the image on primary surface. The *BackBuffer* can be switched easily with the current displayed frame. The purpose of this framework is to allow maximum flexibility while drawing 3D objects onto the screen. The frame data that is to be displayed next on the screen in generally manipulated on the off-screen surfaces.

Page Flipping, HAL (Hardware Abstraction Layer)

Once every video frame, the back buffer is updated from one or more off-screen surfaces and then the back buffer is presented to the display screen. This process is called page flipping. During this process, the graphics microprocessor flips the addresses of front and back buffers and the next image drawn on the screen comes from the previous back buffer. While the previous front buffer is now back buffer and is ready to be used for the coming video frame. Ideally this process takes place in video hardware and is extremely fast not allowing any shearing or tearing of the image while changing from one video frame to the next.

In our case, during each flipping operation a complete stereo image will be sent down to the HMD. This image will be acquired from the network video stream while the drawing of the current image on graphics screen is in progress. A stereo snapshot that is just to be flipped to the HMD is shown in figure 53. In short, the stereo video is updated on local display in a page-by-page format and not pixel-by-pixel which delivers great benefits in terms of reducing time delays.

Direct3D, a component of *DirectX*, delivers real-time full 3D rendering and transparent access to hardware graphics acceleration boards. In other words, it allows Windows to make use of the advanced graphics capabilities found in 3D hardware graphics boards. However in doing so it utilizes the *HAL (Hardware Abstraction Layer)*. The use of *HAL* guarantees increased stability and portability of *DirectX* application. *HAL* serves as a thin wrapper around the *DDI(Device Driver Interface)*.

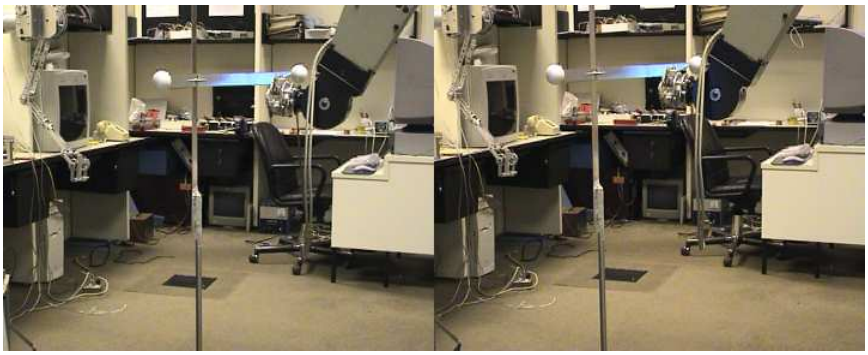


Figure 53: A stereo snapshot ready to be displayed on HMD

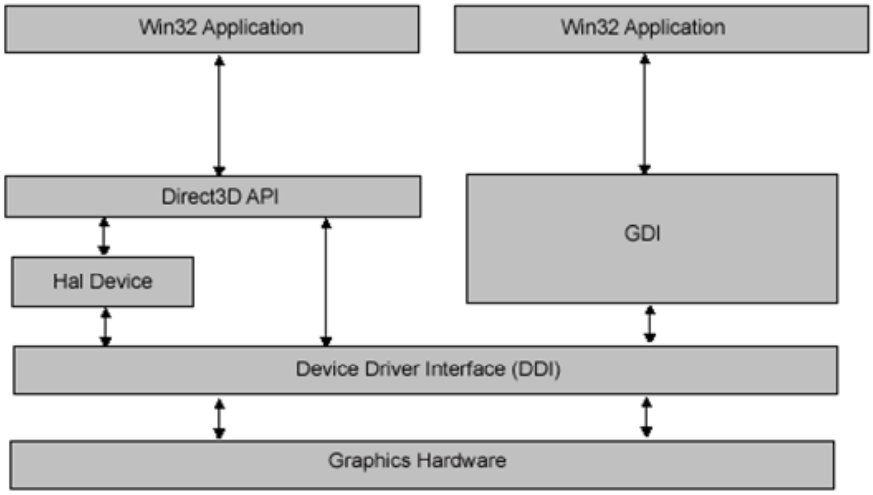


Figure 54: HAL Device overview

Let us try to understand the need for a *HAL* wrapper. If we want to draw a circle on the video display, instead of drawing it pixel-by-pixel, we would like to create it with a single *circle(x,y, radius)* command, where x, y indicate the origin of the circle. Usually this command should be supplied by the graphics hardware vendor through *DDI*. Each hardware vendor will supply a different implementation for the *circle* method. So our *DirectX* application will be restricted to only one type of graphics board. *HAL* is the solution to this problem. It supplies us with a generic *circle* method along with many other useful graphics commands which are implemented by each hardware vendor at *DDI* level. The position of *HAL* in the whole graphics pipeline can be understood by having a look at figure 54.

6.6.5 Component framework

This augmented reality system was realized using component based software development keeping in view the ease of extensibility, reusability and compactness. After development, these components were made part of the already existing distributed telerobotic framework described in section 6.4. A detailed description of client and server side components related to augmented reality is given in the following sections.

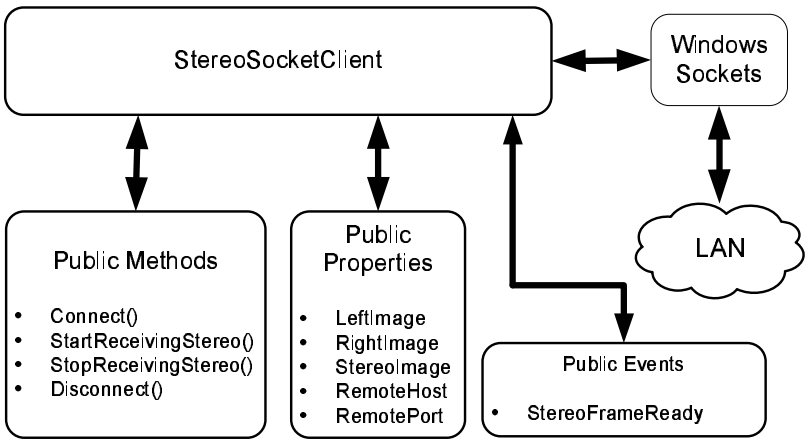


Figure 55: StereoSocketClient Component

6.6.6 Client side components

Listed below are the components providing augmented reality functionality on the client side:

1. StereoSocketClient Component
2. IdentifyCamera Component
3. RobotModel Component
4. DXInterface Component

A brief description of all these components follows:

StereoSocketClient component

The stereo video server on the remote side sends binary video data stream to the client side through windows sockets. This stream consists of *BITMAPINFO-HEADER* carrying the information header of the bitmap data, the bitmap buffer size, and the bitmap data itself. A socket interface must be used on the client side to retrieve the binary data. And after the byte data has been retrieved from the socket stream, we need some mechanism to regenerate compatible bitmaps from this data. StereoSocketClient component provides this very functionality. A block diagram of StereoSocketClient is shown in figure 55.

The public methods exposed by the StereoSocketClient component are:

bool *Connect()* : Used to connect the client socket to the remote vision server.

bool *Disconnet()* : Disconnects the client socket from the remote server.

bool *StartReceivingStereo()* : Starts receiving stereo images from the remote side.

bool *StopReceivingStereo()* : Stops receiving stereo images from the remote side.

Similarly a description of the public properties is given below:

Bitmap *LeftImage* : A get only property that returns a copy of the freshly received left image.

Bitmap *RightImage* : A get only property that returns a copy of the freshly received right image.

Bitmap *StereoImage* : Returns a copy of current stereo image in bitmap format combining left and right images in side by side fashion.

String *RemoteHost* : A get/set property specifying the DNS name of the computer running vision server.

int *RemotePort* : Get/set property indicating the port address of *RemoteHost*.

In order to use the component, first the calling thread creates an instance of the StereoClientComponent. *RemoteHost* and *RemotePort* properties are set properly for the computer running the vision server. Then *StartReceivingStereo()* method of the component is called. This call creates a separate thread for receiving the images. Whenever it receives fresh copies of both left and right images, a *StereoFrameReady* event is fired by the image receiving thread and the fresh copies of the images are available immediately through *LeftImage* and *RightImage* properties. This event is synchronous which means that until the called thread has read the image data, the event stops the execution of calling thread which in our case is the image receiving thread. By this, we ensure that the image data is not overwritten during the copy operation. If the container of the StereoSocketClient component needs the stereo image instead, it can read the *StereoImage* property of the component that returns a single stereo image in left/right format.

IdentifyCamera component

As explained in section 6.6.3, camera identification must be done to accurately position 3D objects on the pixel plane. For a given set of four non-coplanar points constituting the basis vector, IdentifyCamera component can be used to find out the camera projection matrices for both left and right cameras.

The component is initialized with the default positions of fiducial points which can be changed. ***UpdatePics()*** method of IdentifyCamera component can be used to update the left and right images whenever they are available through the video stream provided by StereoSocketClient component in the form of bitmap images. These pictures are updated on the GUI provided by the component as shown in figure 51.

Public properties of the component are the following:

LeftProjectionMatrix : Projection matrix for the left camera.

RightProjectionMatrix : Projection matrix for the right camera.

LeftOriginCorrection : Returns left origin correction in number of pixels to be added to 3D projections of the virtual points.

RightOriginCorrection : Right origin correction in number of pixels.

On the closure of the GUI, both the left and right projection matrices M_l and M_r, respectively, are available to the calling thread based on the mouse clicks of the user.

RobotModel component

This component plays an important role in the realization of the augmented reality system. This acts as a local proxy of the PUMA robot which is also available in the form of *IDecisionServer* interface. The difference between the two is that *IDecisionServer* is an active proxy of the *DecisionServer* component. Any call to the public methods or properties of *IDecisionServer* interface will be directed to the active instance of *DecisionServer* component on the server side through .NET remoting. While the *RobotModel* component is a passive proxy which is, in no way, connected to the instance of *DecisionServer*. This setup requiring the use of *RobotModel* is needed to locate the future position of the robot gripper based on the current command that is being sent to the robot through *IDecisionServer* interface.

RobotModel component provides following public methods:

bool *InitializeModel()* : Initializes the RobotModel to the default reference position. This position is the same as used to initialize the PUMA component on the server side. However the method is overloaded and can initialize the model to any given set of joint angles as well.

bool *MoveModel(double[] incAngleData)* : Moves the model using incremental joint space values. The parameter *incAngleData* specifies the incremental values of joint space variables.

bool *MoveModel(pVec, oMat)* : Incrementally moves the robot model in cartesian space. The parameters *pVec* and *oMat* specify the incremental values of position vector and orientation matrices.

The public properties exposed by the component are:

***PositionVector* :** Retrieves the current position vector of the model (PUMA gripper).

***OrientationMatrix* :** Current orientation matrix of the robot gripper.

***RobotAngles* :** A write only property for setting the current joint space variables of the *RobotModel*.

RobotModel component can be thought of as a thin copy of *PUMA* component removing the robot hardware related functionality. It has complete inverse geometric model of the robot that it uses to move the robot when its *MoveModel* method is invoked with incremental position vector and orientation matrix. While moving the model, the component also takes care of the *RobotFrameMode* of PUMA robot. Upon moving it using the incremental joint space variables, it uses the direct geometric model to calculate the current position vector and orientation matrix.

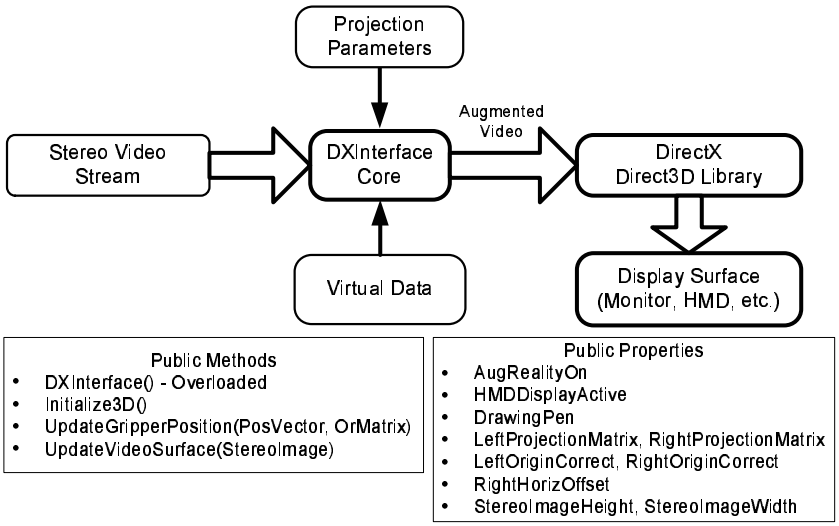


Figure 56: An overview of DXInterface Component

DXInterface component

This is the central component of augmented reality framework. All the video related tasks such as 1) augmentation of real video, 2) synchronization of real and virtual data, 3) projection on video surface, 4) page flipping for HMD stereo visualization, are handled by *DXInterface*. An illustrative overview of the component is given in figure 56.

The component receives video stream in the form of stereo bitmap images from the *StereoSocketClient* component. Two other inputs of the *DXInterface* component are the projection matrices for the two cameras as well as the virtual data to be augmented with the real video stream. Before using the *DirectX* libraries for video manipulation, a *DirectX Device* must be initialized. This device will server as an interface to the video manipulation functions of *DirectX*. In our case, a full-screen device is created keeping in view that the data is to be sent to an HMD for 3D viewing. The image is so adjusted that a complete stereo images is precisely divided into two subsets, each for left and right eye, when displayed on HMD. Figure 57 shows the HMD used in the experiments.

The *Device* then creates a *DirectX* surface, *frontSurf*, for the real video data storage. This surface also serves the purpose of the backup of real data. Then a copy of the surface, named *augSurface* is made for the augmentation purposes. This copy is then used throughout all the projection and rendering pipeline. Whenever a new image from stereo video stream arrives, the *fronSurf* is updated and again a copy is given to the *augSurf*.

The virtual data in our case is the 3D gripper position. We need to draw a small ball at the supplied gripper position. DXInterface applies the camera model given in equation 310 utilizing the supplied left and right projection matrices. The component while projecting the point to the real data surface must also take into account the horizontal offset for the right image because the stereo image is saved as a single image in the video memory. If we need to write something for the right frame of stereo image, a horizontal offset equal to the image width of a single image

Figure 57: HMD and its controller

must be added to any point being projected to the right hand side.

Whenever a new gripper position is received to be displayed, *DXInterface* uses *augSurf* to write virtual data to the video surface. After the augmentation, the data is rendered using *Present()* method of *DirectX* interface. This operation of presenting the the data to the display screen is accomplished using page flipping. The video memory address of front buffer is flipped to the back buffer and vice versa. All the information on the previous front surface is discarded during flipping operation.

DXInterface provides following public methods:

DXInterface() : An overloaded constructor that accepts left, right projection matrices, screen size and other parameters to be used in 3D *Device* initialization.

Initialize3D() : Initializes the 3D *Device* in full screen mode.

UpdateGripperPosition(posVector, orMatrix) : Augments the real display with virtual ball on the supplied location specified by the parameters posVector(Position Vector) and orMatrix(Orientation Matrix). Although we do not use the orMatrix in placing the virtual ball in 3D space, it may be useful in possible future work in placing complex 3D objects.

UpdateVideoSurface(StereoImage) : This method updates the current video surface when a new stereo frame arrives from the network video stream. In drawing the new real data, the virtual data is preserved.

The public properties exposed by the component are as under:

bool AugRealityOn : Can be used to toggle the augmented reality on stereo display.

bool HMDDisplayActive : A read only property indicating whether 3D *Device* is sending data to HMD or not.

DrawingPen : The color to be used to draw augmented data.

LeftProjectionMatrix : Left camera projection matrix.

RightProjectionMatrix : Right camera projection matrix.

LeftOriginCorrect : Origin correction in number of pixels for left image.

RightOriginCorrect : Origin correction in number of pixels for right image.

RightHorizOffset : Horizontal offset in pixels for drawing virtual data to right image, with respect to left image.

StereoImageHeight : Stereo image height.

StereoImageWidth : Stereo image width.

Server side

Server side acquires and sends the stereo image data through windows network sockets. However only the client side is responsible for major augmented reality business. In section 6.5, we discussed the server side for the stereo video client-server framework. The same vision server is used in the AR framework. *StereoSocketClient* is intelligent enough to understand the socket stream sent by the vision server developed for MFC(Microsoft Foundation Classes) client-server setup.

6.6.7 The complete augmented reality system

All of these components have been combined together to form a complete augmented reality system on the client side. The system provides the augmented reality functionality through the following steps:

1. Input from the user is taken through the *MasterArm* component.
2. *MasterArm* provides incremental position vector and orientation matrix to *IDecisionServer* and *RobotModel* components.
3. *IDecisionServer* executes the incremental move command on remote *DecisionServer*.
4. *RobotModel* component provides the new 3D position of gripper to *DXInterface* component.
5. *DXInterface* has already acquired a stereo frame of remote scene from *StereoSocketClient* component as well as left and right projection matrices from *IdentifyCamera* component at the system initialization.
6. *DXInterface* projects a virtual ball at the gripper position in 2D stereo image and sends the stereo image to HMD controller in order to display it to user.

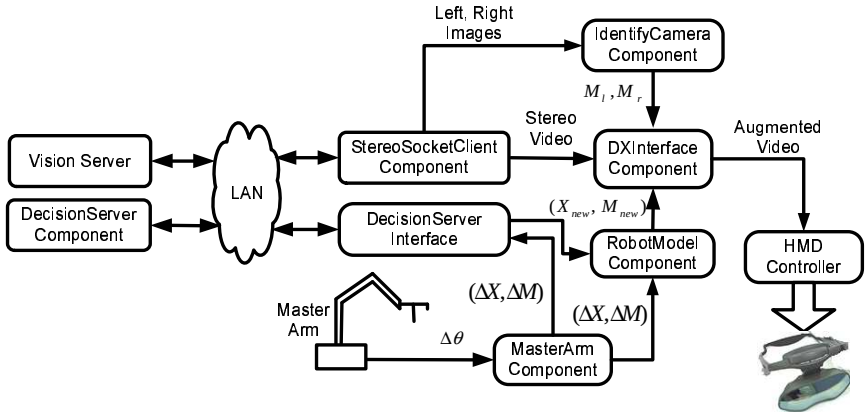


Figure 58: Block diagram of complete AR system on client side

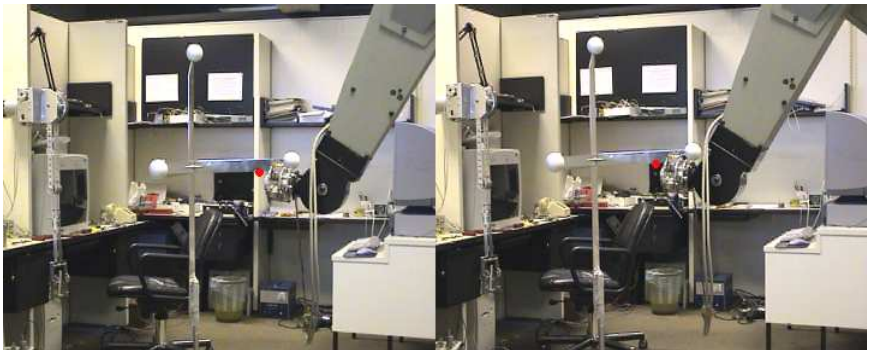


Figure 59: A real scene augmented with a red ball

7. When the *IDecisionServer* receives the *OnMove* event from the remote side, the current angular position of the robot are sent to the *RobotModel* to update the local model.

An architectural overview of the augmented reality system present on the client side, is given in figure 58.

It is important to update the local robot angles upon the invocation of *OnMove* event from the server side because there may be some differences between the move command arguments and the current position of robot due to mechanical and mathematical roundoff errors. Also it is to be noted that the *IDecisionServer* uses *.NET Remoting* for network streaming of component data and force data while Vision Server and *StereoSocketClient* use raw windows sockets to transfer binary video data. This setting makes it a true multi-stream distributed framework.

The accuracy of the augmented ball at the gripper location depends on the position of cameras from the robot gripper and the distance between the reference frame and robot itself. As we increase the distance between the cameras and robot, projection becomes more and more accurate. A real scene augmented with a red ball at the projected gripper position is shown in figure 59.

The ball seems to be a bit farther than the tip of force sensor because the length of the gripper used in 3D projection is longer than the force sensor.

The system has the ability to remember the identification of cameras and other projection related data across different runs by preserving these values to the per-

manent memory in a special format. So, the identification is required only when the cameras or the objects have been moved from their previous locations.

6.7 A 3D Vision-Based Man-Machine Interface For Telerobotics

This study presents a telerobotic system [87, 88, 16] that consists of a vision-based station (client) and a slave robot (server) which are interconnected through a 100 Mbps Ethernet LAN. The client-server system is implemented using a robust Visual Basic (VB) programming environment together with TCP/IP socket programming to provide real-time connectivity through the LAN. A real-time vision system consisting of two digital cameras monitors the operator hand motion to control a tele-robot. To view the robot scene, the operator uses eyes shuttering glasses with display of stereo views at the client station. The generation of stereo views [39, 38] lead to display of left and right images of the robot which produces the stereo effects after proper synchronization.

Two digital cameras are used to monitor a four-point feature frame that is held by the operator hand. The frame consists of four balls having distinct colors. The cameras monitor the trajectory of each ball. Using un-calibrated stereo vision the multiple-view affine invariance property is used to build a 3D interpretation for the feature frame which is considered as a reference of the operator hand frame. The frame is represented by twelve parameters, three for the cartesian coordinates of its origin and nine for its orientation matrix. These are considered are reference parameters for the position of operator hand. The position and orientation of the operator hand frame are transmitted in a differential manner to the slave robot by using the client-server network interface. Visual feedbacks are also forwarded from the slave station to master station to provide the operator with stereo-views of the slave arm. The operator can see the effects of the previous motion which enables making the necessary motion corrections through repetitive operator hand-eye interactions.

Fukumoto et al. [89] proposed a stereo vision system where the user can point a place on the computer's screen by his hand and give some commands by hand gesture. However, such conventional vision systems still impose some restrictions. They require camera calibration which limits the user motion. The absolute position and orientation of the hand in the space are used for gesture recognition, which means that the operator should not move his body from the initial position at the calibration. However, the operator cannot expect how the system will work when he moves his body from the initial position.

Estimating 3D position of abject is also used in model-based telerobotics that allows obtaining and maintaining accurate models of the remote site. To overcome the problems of time-delay and bandwidth limitation the operator directly interacts with a model of the remote site instead of delayed remote site. For this the operator points to a known object feature in a video image of the remote site and use 2D images of these features to solve for the 3D position of the object. Using one single camera Lloyd [90] used the pin-hole camera model with off-line calibrated focal-length and radial distortion for one single camera. Using simple camera calibration the geometry of affine stereo vision is used [91] to estimate the positions and surface orientations needed to locate and reach for objects by sight. The advantage of this system is its immunity to unexpected translations and rotations of the cameras and changes of focal length. Un-calibrated stereo vision [92] is also used in a pointing-

based interface for robot guidance based on the use of active contours to track the position and pointing direction of a hand in real time.

An interactive human-robot interface [93] is proposed to track a hand pointer using a constrained perspective transform. The real-time tracking system visually tracks the operator's pointing hand and projects a mark at the indicated position using an LCD projector. The mark is visually observed by the operator, thus making possible correction of the indicated position.

Kuno et al. [94, 95] proposed an interfacing method by using un-calibrated stereo vision. Their system is based on the multiple view affine invariance theory. It calculates the hand positions as invariant coordinates in the basis derived from four points located on the user's body in a user-centered frame so that the operator can move his hand forward and backward relative to his body regardless of possible body motion. In this case, only the hand motion with respect to the body is measured by the camera system.

We propose a system that avoids taking the operator's position into account. Our system recognizes the position and orientation of the feature frame regardless of the position of the operator. For this we use the multiple view affine invariance theory. The position and orientation of the feature frame is evaluated with respect to its previous configuration. The position of the balls are calculated as invariant coordinates in the coordinate system with the three basis vectors defined by the four points. For this we evaluate the position of the feature frame with respect to its position in the previous frame which allows evaluation of an incremental motion that can be directly mapped to the tool frame of the slave arm. The operator can control the slave arm by moving the feature frame regardless of the position of his body. This approach needs no camera calibration because the camera parameters do not affect the affine invariance feature.

This system is based on robust boundary detection, fast tracking strategy, and a simple mechanism for partial occlusion. The gravity center of each ball should be computed precisely. The use of color information to improve discrimination and enables the use of a wide range of color selection compared to the gray image based techniques. In the computer vision, the detection of a colored object is known to play essential role in extracting specific information from the scene therefore we have selected the RGB color space for the image processing and color detection. Ideally an increase in the communication delay is translated by a graceful degradation in the task execution time, i.e. resilient system. For this the control strategy is based on a coarse-control of the slave arm which leads the operator to assign a coarse-trajectory to the slave arm leaving the generation and fine trajectory control to a local slave controller.

This section is organized as follows. Section 2 presents some background on the color spaces. Section 3 presents a metric to measure color matching. Section 4 presents our the tracking algorithm. Section 5 presents the 3D position matching module. Section 6 presents the evaluation. Section 7 concludes about this work.

6.7.1 Background

Due to the structure of human eye all colors are seen as variable combination of the three primary colors: Red, Green, and Blue. There are different format for

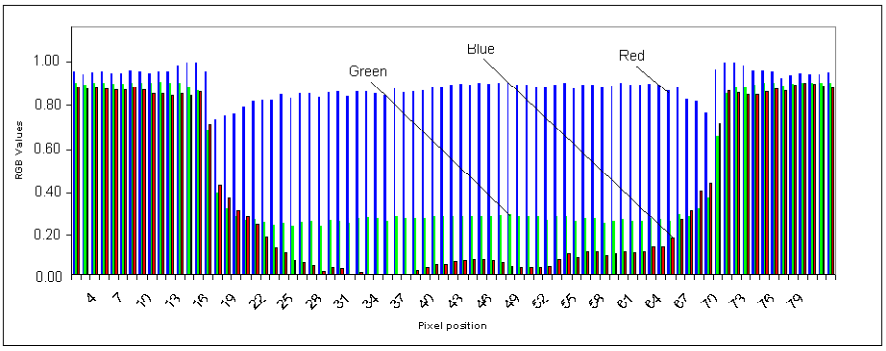


Figure 60: The RGB for a scan of a red ball with white background

colors in video cameras, such as YUV, HIS, etc. We have used the 24 bits RGB format in which each color is assigned one byte. Ideally any primary color like red, green, or blue should consist of one component only. In practice the colors that appear in different images taken by different cameras are not stable under different conditions. For example, a horizontal scan of a red ball with white background produces the RGB shown in Figure 60. The middle part (ball) of the figure indicate that there are some components for the green and blue that cannot be omitted for a specific red color. There are various reasons for the problem of variation in the values of RGB components, like light reflection, color saturation, camera sensitivity and configuration, external noise,..etc.

Besides the RGB color space, there are some other 3D color coding, like YUV, YCBCR, HIS, etc., but in general using of the above coding other than RGB converting from one coding to another which increases the processing delays. For a high-speed image application with real-time control, the extra computational overhead and delay may not be acceptable. Therefore, processing such as edge detection based on the raw RGB coordinates will have an advantage. The existence of a color edge implies changes in terms of chromaticity or luminance or both. However monochrome (i.e. gray scale) edges only take into account changes in luminance. Therefore, the edges detected in a monochrome image may not correspond to the set of edges existing in a color image, because of regions that are indistinguishable in terms of their luminance but differ in terms of their chromaticity or vice versa. In the past, edge features were generally obtained by applying special filters and gradient operators to gray level images with the aim of minimizing false detection and enhancing the noise rejection capability. The use of color information has been suggested and proved to perform better than techniques that operate on gray image alone.

Our approach is based on using two digital cameras tracking a feature frame that consists of four balls (red, green, blue, and yellow) located at the edges of three orthogonal vectors. The objective is to accurately evaluate the 3D position of the above moving frame, that is due to operator hand, and to assign the identified motion to a slave robot in an attempt to create a mechanism that replicates the operator hand motion. The vision system has the following components: (1) a metric to measure color matching (Section 6.7.2), (2) a tracking algorithm running on two

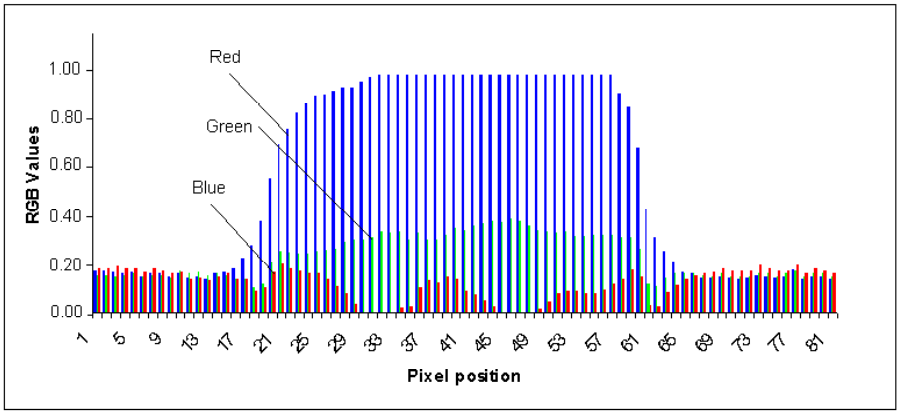


Figure 61: The RGB for a scan of a red ball with black background

computers where each computer is interfaced to a digital camera (Section 6.7.3), (3) each computer tracking the motion of the feature frame, with respect to each camera, determines the coordinate of each ball, and (4) a stereo matching approach is used to compute the 3D position of each ball which enables finding the position and orientation of the feature frame Section 6.7.4.

The tracking algorithm tracks the ball motion, carry out color edge deletion by using a color matching function , and evaluate the coordinate of ball centers by using the 2D images captured by the digital cameras. One computer collects the ball positions from the other computer prior to running the 3D stereo matching module. The stereo matching module evaluates the 3D position and orientation of the feature frame based on the ball positions provided by each tracking algorithm running on a computer-camera subsystem. The operator interface allows carrying out supervised learning of color features for each ball prior to running of the tracking system. It also displays (1) the current feature frame image with pointers to detected positions, (2) the RGB plot along a horizontal or vertical scan of a ball, (3) the color information, and (4) the 3D position and orientation of feature frame. In Section 6.7.6 we present performance evaluation of (1) ball position tracking and associated errors under static and dynamic conditions for each camera, and (2) stereo tracking of the feature frame position and associated errors under static and dynamic conditions.

6.7.2 A metric to measure color matching

Every pixel on the image is represented by three bytes, which means that every primary component will be stored as a byte of information. This property of the RGB color space help us to explore the color features, and consequently to detect them.

A color pixel p is represented by its RGB components $p = (c_1, c_2, c_3)$, where c_1, c_2, and c_3 are the luminance of the red, green and blue of the RGB components. Although each of R, G, and B is represented by one byte (256 levels) we assume normalized RGB components, i.e. $0 \leq c_j \leq 1$ for $j = 1, 2$, or 3. For example the

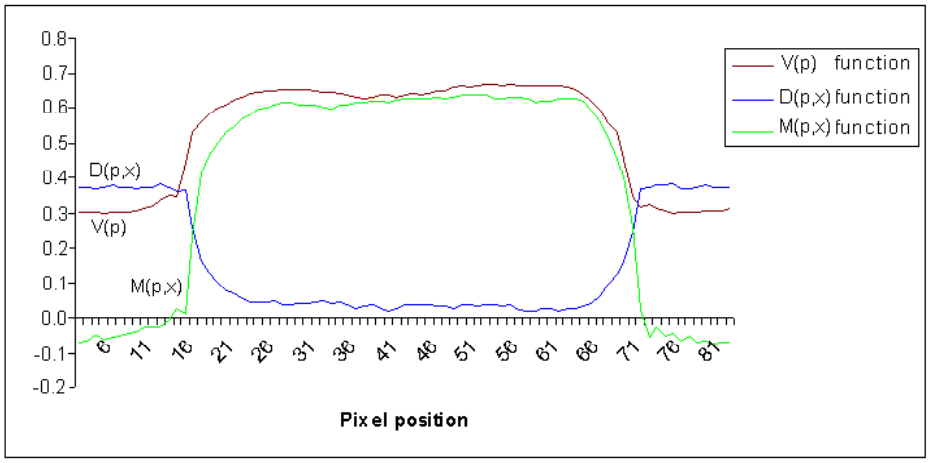


Figure 62: Metric functions for a scan of red ball with white background

horizontal scan of a red ball with white background shown in Figure 60 produces an RGB plot versus the pixel location. Note that the white background has similar c_1 component to the red ball which has non-zero c_2 (green) and c_3 (blue) components. In this case, the red ball can mainly be discriminated from its background based on its c_2 and c_3 components. Note that the white background has similar c_1 component to the red ball which has non-zero c_2 (green) and c_3 (blue) components. A black background significantly reduces the c_1 components which improve the discrimination with red ball but the c_2 and c_3 components are still present as shown in Figure 61.

A monochromatic color has only one luminance component. A reference color i is represented by its reference RGB parameters (c_1^i, c_2^i, c_3^i). We define a function $V_i(p)$ to measure how close a color pixel $p = (c_1, c_2, c_3)$ is to a reference color i, where $1 \leq i \leq 6$. The reference color i can be a monochromatic color (one color) like the red, green, or blue or a combination of two colors like the red-green, red-blue, or green-blue. We define a color luminance function $V_i(p)$ as follows:

$$V_i(p) = \frac{1}{3}w(p) + \frac{1}{3}\begin{cases} 2(c_i - 1) & \text{for } 1 \leq i \leq 3 \text{ (one color)} \\ \sum_{j=1,j\neq k}^{3}(2c_j - 1) & \text{for } 4 \leq i \leq 6 \text{ (two colors)} \end{cases} \tag{312}$$

where $w = \sum_{i=1}^{3}(1 - c_i)$. Note that use of normalized color components leads to $0 \leq V_i(p) \leq 1$. Using Equation312, for each ball of the feature frame we define a set of four colors for which $V_i(p)$ is given by the expressions:

$$V_i(p) = \frac{1}{3}\begin{cases} c_1 + (1 - c_2) + (1 - c_3) & \text{for the primary red color } (i = 1) \\ (1 - c_1) + c_2 + (1 - c_3) & \text{for the primary green color } (i = 2) \\ (1 - c_1) + (1 - c_2) + c_3 & \text{for the primary blue color } (i = 3) \\ c_1 + c_2 + (1 - c_3) & \text{for the red-green } (i = 4) \\ c_1 + c_3 + (1 - c_2) & \text{for the red-blue } (i = 4) \\ c_2 + c_3 + (1 - c_1) & \text{for the green-blue } (i = 4) \end{cases}$$

The first three colors $(1 \leq i \leq 3)$ are being the primary RGB colors which are the red, green, and blue. The function $V_i(p)$ is maximal if p has full component $c_i = 1$

on a primary color i and zero component on the remaining two. The complement of component c_i is being $1 - c_i$ for normalized references but in practice each primary color component occupies one byte, i.e. 256 levels. The last three colors ($4 \leq i \leq 6$) represent any combination of two primary colors such as the red-green, red-blue, and green-blue. In this case the function $V_i(p)$ is maximal if p has full component on two primary colors and zero component of the third color. Currently we are using four balls colored with red ($i = 1$), green ($i = 2$), blue ($i = 3$), and yellow ($i = 4$).

Function $V_i(p)$ represents the luminance of primary color i at pixel p. Other functions can also be used. Another implementation [96] is $V_i(p) = 1 - \prod_{j=1}^{3}(c_j - c_j^i)$, where $p = (c_1, c_2, c_3)$ and c_j^i is jth component of the reference color i. In this case $V_i(p)$ will be maximal if at least one component of p matches the corresponding reference color component. For example, the best combination of red color is $r_1 = 1$ (red), $r_2 = 0$ (green), and $r_3 = 0$ (blue) which gives $V_1(p) = 1$.

The function $V_i(p)$ measure the similarity between a color pixel p and a single or combined primary color. Ideal colors are difficult to design. In addition they may have have different values for their primary components under different conditions of lighting and image quality. This explains the need to use the RGB components of a reference color, like those of a selected colored ball, as reference parameters. These references are useful for matching with those of a color pixel to prevent the detection of color pixels having high $V_i(p)$ values when the searched ball is occluded or out of range. Notice that with a black background the c_2 and c_3 components are similar for those of the red ball. In the case of black background 61 there is a good preservation of relative composition of the RGB components as compared to the case of the white background 60. The selection of threshold value for $V_i(p)$ is not governed by any rule and can be affected by the changes in the image and lighting conditions. This shows that the color luminance function $V_i(p)$ may lead to processing of many exceptions derived from detection errors.

Another metric to measure the color matching can be selected as the normalized distance $D(p, p_{ref}^i)$ between the a reference color $p_{ref}^i = (c_1^i, c_2^i, c_3^i)$ and a given color pixel $p = (c_1, c_2, c_3)$. The normalized distance color matching is defined as:

$$D(p, p_{ref}^i) = \left(\frac{1}{3} \sum_{j=1}^{3} (c_j - c_j^i)^2 \right)^{1/2}$$

The distance depends on the feature parameters of a specific reference color which are generally determined during the supervised learning period and can be dynamically updated at run time if the referenced ball is detected with high confidence. Although the distance metric gives useful results in general it may fail because many sporadic scene pixels may have components that are quite similar to those of the reference. This situations can occur under poor lighting conditions or in noisy environment. Then the detection of the correct color in a narrow range will be more difficult. For the above reasons the distance matching may sometimes give poor results. The objective is to have one single metric that maximizes the discrimination of ball colors from a wide spectrum of realistic background colors. One approach to preserve the benefit of the distance matching $D(p, p_{ref}^i)$ and the color luminance function $V_i(p)$ is to combine them into one single color matching function $M(p, p_{ref}^i)$ defined by:

$$M(p, p^i_{ref}) = V_i(p) - D(p, p^i_{ref})$$

where the subscript i denotes the color of one of the four balls and p^i_{ref} represents its normalized RGB reference parameters. The color matching function satisfies $-1 \leq M(p, p^i_{ref}) \leq +1$ which gives 1 and -1 for a maximally and a minimally matched color pixels, respectively. Figures 62 show the plot of functions $V(p)$, $D(p, x)$, $M(p, x)$ where p is pixel position and x is the reference to the red color. Both plots show that $M(p, x)$ maximizes discrimination between the red ball and its background as compared each of $V(p)$ and $D(p, x)$.

This approach enables the use of realistic colored balls while providing good color discrimination if the selected colors are as close as possible to the primary RGB components. The color discrimination is improved compared to use either of the above metrics. The color luminance function $V_i(p)$ contributes in improving discrimination in poor lighting conditions and under noisy background. For this, we use the color matching function $M(p, p^i_{ref})$ in the remainder of this section as the main technique for color detection and matching.

6.7.3 The tracking algorithm

In this section we first present the color matching function, then our tracking algorithm that monitor the position and orientation of the feature frame. A too small search area may lead to missing the searched object. A too large search area leads to slowing down the tracking algorithm. For this we use a uniform and a spiral searching techniques (Section 6.7.3) with backtracking to minimize the number of visited pixels while covering a relatively large area. This algorithm allows tracking the motion of the feature frame by identifying its instantaneous position and orientation (Section 6.7.3). For this each camera tracks the feature frame, shown in Figure 63-(a), and identifies the coordinates of each ball with respect to its camera frame. Using information from both cameras the 3D position and orientation of the feature frame can then be evaluated. The metric for color matching (Section 6.7.2) allows boundary detection of each ball which enables finding its position in the camera frame. The algorithm also handles the case of partial or total occlusion (Section 6.7.3) among the balls and determine reasonable solution for each case. To consolidate the detection our algorithm validates the detected balls through shape and geometric matching 6.7.3 prior to dynamically updating the ball reference colors.

The tracking mode represent the normal operation, where all the balls were detected successfully in the previous frame. The flow chart of the tracking algorithm is shown in Figure 64 which describes the algorithm structure and its major functions. In this mode, there are different approaches are used to assist tracking the ball and to measure the center and the diameter of each ball. In the following, we present the major functions of the tracking algorithm.

Supervised learning of colors

In supervised learning allows finding the typical RGB parameters for each color ball. For this the user points to each ball in the image of the feature frame. The reference

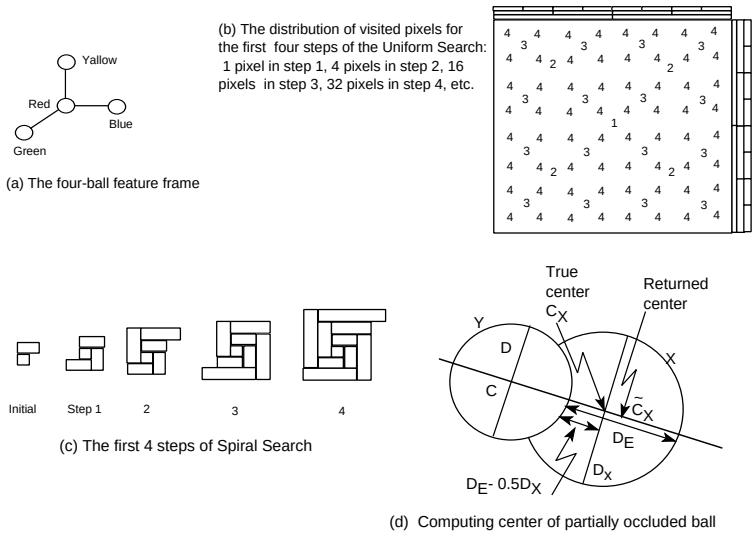


Figure 63: Feature frame, Uniform and Spiral search, and the case of occluding

RGB parameters for each color are determined by using the histogram technique which cluster the color pixel population against the immediate neighborhood of the ball. The display of horizontal scans over the selected ball allows checking the retained parameters and the detection borders.

Boundary detection and searching area limits

We noticed that a fast refreshing rate with relatively simple tracking algorithm, of the feature frame, and a narrower searching area is more effective than a slower algorithm with wider search. One of the main issue is the processing and memory access times of a large number of pixels associated with the use of commercial processors. A slow refreshing rate constrains the speed of operator hand in addition to increasing the probability of exception occurrence for which one of balls is not detected and a costly search of a wider image area becomes the only solution. For this an effective position prediction of the feature frame combined with a narrower searching area makes faster memory access time of cached data for two reasons. First, we only need to process a small volume of data associated with fast memory access because the locality of the searching area is easily captured in the processor cache memory, i.e. better re-use of cache data. For this we search a square area centered at a point that is linearly predicted based on the previous positions of each ball in each camera frame. Second, given the refreshing frequency of the tracking mode it is found that the side of the searching area needs not to exceed three times the most recent diameter of the corresponding ball. For boundary detection, the use of the color matching function $M(p, p^i_{ref})$ for a scene color pixel p enables matching to the ith ball reference color p^i_{ref} whenever the following condition is met:

$$M(p, p^i_{ref}) \geq M(p_{bkg}, p^i_{ref}) + \alpha \times (M(p_{typ-i}, p^i_{ref}) - M(p_{bkg}, p^i_{ref})) \qquad (313)$$

where $M(p_{bkg}, p^i_{ref})$ and $M(p_{typ-i}, p^i_{ref})$ is the color metric value at a neighboring background pixel p_{bkg} and at a typical ball pixel p_{typ-i} both taken from the previous

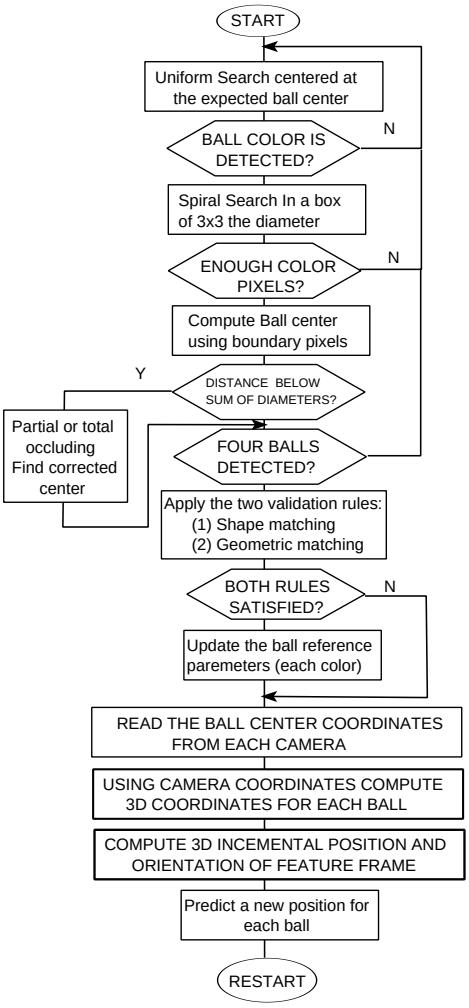


Figure 64: Flow chart of the tracking algorithm and its major functions.

tracking iteration, and α is a constant satisfying $0 < \alpha \leq 1$. Note that p^i_{ref} is a reference for the ith color (ball) that is determined from (1) the supervised learning phase, or (2) the most recent validation and reference update. Note that the right hand side of Equation 313 is recomputed once following each successful ball detection.

To minimize the searching time, the search of a ball within a predicted area is based on alternating between a *uniform search* (US) and a *spiral-shaped* (SS) search within the searched area as shown in Figure 63-(b) and (c), respectively. Initially, no information is available and US is started. When one pixel matches the searched color we switch to SS and search around the detected pixel and continues until complete detection of the ball or abandon the SS search if enough inconsistent evidence are accumulated. An inconsistent ball detection occurs when the number of matched pixels is a small fraction of the total number of visited pixels by the SS search. The algorithm backtracks to the US search, at the previous state, when the SS search is abandoned. However, if the predicted area is visited without detecting the ball the algorithm restarts with a larger search area. Note that the coordinate of each searched pixel that meets the condition stated in Equation 313 will be

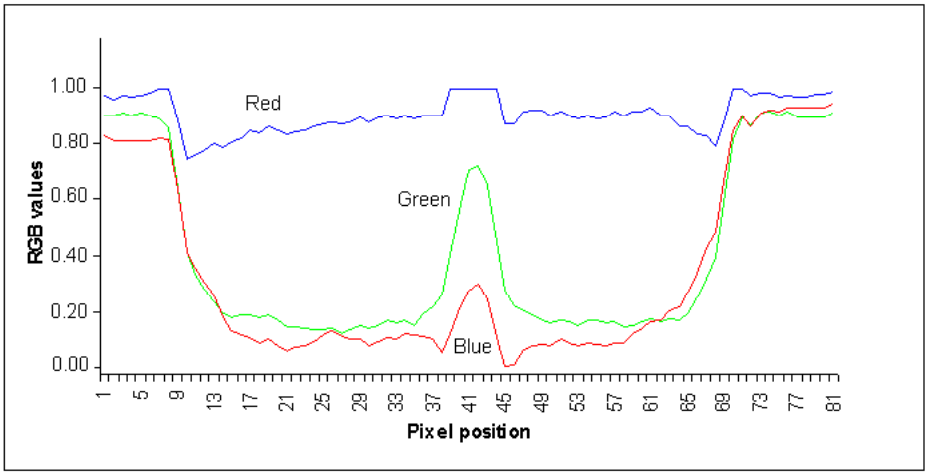


Figure 65: The RGB of a red ball, white background, and a reflection area

considered in the evaluation of the ball gravity center in a later step.

The SS search is based on a 2D square-shaped spiral algorithm that starts at a predicted pixel. The algorithm is based on repeatedly alternating the direction while moving in a bi-cyclic fashion over an increasing number of pixels. The direction of motion is fixed for n pixels. In each step the spiral is created by scanning n pixels along a given direction, i.e. the row or the column. Next the motion direction is rotated by $\pi/2$ in the plan and another set of n pixels is scanned in the new direction. The next step starts after incrementing n. A segment size of n pixels indicates that the total number of scanned pixels is $n(n+1)+1$. Thus the spiral search is ended when the total number of scanned pixels exceeds a square whose side is three times the most recent value of the ball diameter (d in pixels). In other words is search is ended when $n(n+1)+1$ exceeds $9 \times d^2$.

The US search consists of uniformly carrying out color matching within a rectangular area (RA) $L_x \times L_y$, where L_x and L_y represent the length and the width. This procedure allows computing the matching function over a set of 2^k uniformly distributed locations (boxes) within RA after k iterations. In each iteration, RA is partition into a number of equally sized boxes and the matching function is computed at the center of each box. At start we initialize the box size to that of RA. In each iteration the box is divided into 4 smaller boxes. If there is a match the ball is detected and we abandon the US search. Otherwise, the algorithm continues after subdividing the box size. In each iteration the number of visited points is four times that of the previous iteration. Thus after the kth iteration the algorithm visited a total of $\sum_{i=0}^{i=k} 4^i$ uniformly distributed pixels within the predicted searching area. The search is ended if the box size becomes smaller than half the expected diameter of the searched ball. In this case the searched ball is not present in RA and a global search must be activated like at the initialization.

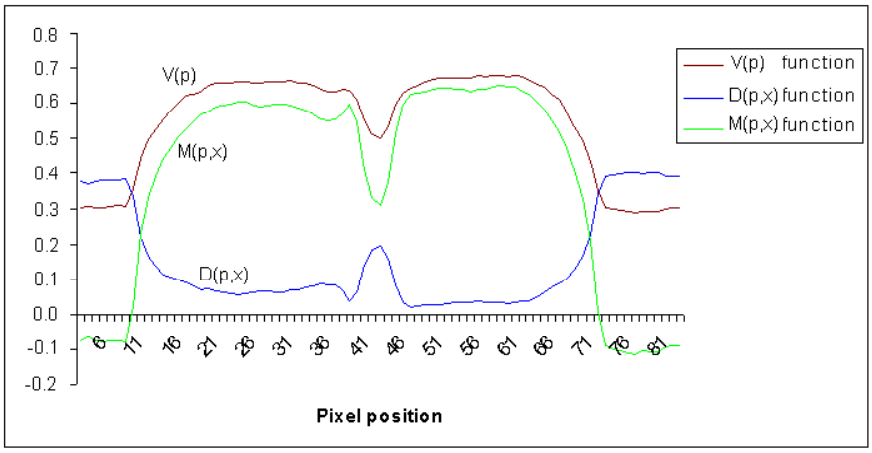


Figure 66: The luminance, distance, and matching functions for red ball a reflection area

Computing the coordinate of ball center

The search is successfully completed when each ball is detected in the predicted area. The ball is generally subject to many sources of noise like the light reflection which produces white regions within the ball boundary as shown in Figure 65 and 66. During the spiral search procedure, in each row i we only record the detected edge pixels located at the most left (j_{min}) and most right (j_{max}) position of a given row that intersects the outer corona of the ball. The edge pixels represent the ball boundary pixels with the background. To significantly reduce the effects of the potential of white regions only the edge pixels contribute in the evaluation of the ball center coordinates. The center of each row is computed using the edge pixels with the assumption that all the pixels between the two edges are matched to the ball color. With this approach the coordinates of the row center (i_r, j_r) are computed by averaging the coordinates of $N_r = (j_{max} - j_{min} + 1)$ pixels within edges (i, j_{min}) and (i, j_{max}). This simplifies to:

$$(i_r, j_r) = \left(i, \frac{j_{max} \times (j_{max} + 1) - j_{min} \times (j_{min} + 1)}{2(j_{max} - j_{min} + 1)} \right) \tag{314}$$

The matched rows contribute to the evaluation of the ball center by summing up the row center coordinates (i_r, j_r) with N_r being their weight. Note that the true metric of the ball diameter D need not be identified. The largest value of N_r (in pixels) for a given ball is considered as the current ball diameter. We also count the total number of matched pixels for a given ball that is used later as an indicator of the detected ball area. This is useful for validating or invalidating the ball shape.

Partial and total occluding

Partial and total occluding may occur among the balls for each camera. The strategy is to monitor the distance between the ball centers and detect a partial occluding

situation which implies that the computation of ball center must be modified to account for the hidden part.

As shown in Figure 63-(d), given two Balls X and Y, the technique of computing the coordinate of the ball center by using Equation 314 is valid when the distance $d(X,Y)$ between the identified centers of two balls (C_x) and (C_y) exceeds $(D_x + D_y)/2$, where (D_x) and (D_y) are the most recently evaluated values of the diameter just before the detection of the partial occluding situation. Otherwise, a ball is considered under partial occluding due to another. The former ball (X) is identified by comparing the values of its currently computed diameter D_x and area A_x to previously stored values of the same parameters that were evaluated in the most recent pass without partial occluding for each ball. The later ball is fully visible and its computed center (C) and diameter (D) are valid. Since ball X is detected under partial occluding the above algorithm returns $(\tilde{C}_X)$ as its center. However the true center (C_X) must be located on the direction U of the unit vector $\frac{\tilde{C}_X - C}{|\tilde{C}_X - C|}$. For ball X, let's assume (D_E) is the computed distance between the edge pixels along the direction U. Then C_X is just $D/2 + (D_E - D_X/2)$ away from C on the direction U. The center C_X can be evaluated by using the following vector equation:

$$C_X = C + \frac{\tilde{C}_X - C}{|\tilde{C}_X - C|}(D_E + 0.5(D - D_X)) \tag{315}$$

Note that if $|\tilde{C}_X - C|$ becomes very small we may assume that $C_X = C$ where ball X is partially or totally occluded.

Validation rules

The RGB references of the ball may change depending on the location of the feature frame. For this we need to update the color references if there is enough confidence in detection of all the balls in the current tracking iteration. The confidence function used here consists of meeting three validation rules [97] for each camera frame which are: (1) the *shape matching*, and (2) the *geometric matching*.

The shape matching measures for each camera how circular are the balls that are detected without partial occluding. In this case the ratio of ball area (in pixels) to the to the square of the current ball diameter (d in pixels) must be close to $\pi/4$. To avoid degradation due to the digitization effect this rule can only be used when d is large enough.

The geometric matching consists of matching the currently detected position of each ball with respect to each other ball to that of the same ball in a scene obtained by extrapolating the previously identified 3D feature frame. In each camera frame, the expected position of the ball center is evaluated by (1) extrapolating its previously detected 3D position, and (2) projecting the expected position over each camera frame. Denote by $X(t+1)$ the expected 3D position of a ball center with respect to the previously detected feature frame $R(t)$. $R(t)$ is defined by its origin $X_0(t)$ and its orthogonal vectors $E_i(t)$, where $1 \leq i \leq 3$. In 3D the ball center vector $X(t+1)$ is defined by $X(t+1) = X_0(t) + \sum_{i=1}^{3} \alpha_i \times E_i(t)$, where α_i, for $1 \leq i \leq 3$, is the ith component of point $X(t+1)$ over $R(t)$. The affine invariant projections of $X(t+1)$ over $R(t)$ allow writing $x(t+1) = x_0(t) + \sum_{i=1}^{3} \alpha_i \times e_i(t)$, where $x(t+1)$,

$x_0(t)$, $e_i(t)$ are 2×1 vectors that represent the projections of $X(t+1)$, $X_0(t)$, and $E_i(t)$ over the camera frame.

The function $d(x_k, y_l)$ represents the distance between the ball centers x_k and y_l as measured by the tracking algorithm in a camera frame. Using the predicted feature frame and its projection in each camera frame we compute the distance $d(x_k^*, y_l^*)$ for the expected ball centers x_k^* and y_l^*. A relative distance error $d(Measured, Predicted)$ that accumulates the mismatches between all pairs of distance errors:

$$d(Measured, Predicted) = \sum_{k}^{3} \sum_{l}^{3} \frac{|d(x_k, y_l) - d(x_k, y_l)|}{Min\{d(x_k, y_l), d(x_k^*, y_l^*)\}}$$

where $Min(.,.)$ is used to normalize the distance error. A small value of $d(Measured, Predicted)$ indicates that the current geometric distribution of the balls with respect to each other in the current scene is similar to that of a predicted scene.

6.7.4 The 3D position matching module

The camera model is based on weak perspective projection. Since the dimension of the balls is a small fraction of the distance between the camera and the ball the weak perspective projection can be considered as a valid approximation of the general projective transformation. This approach uses un-calibrated stereo vision based multiple views generated by two cameras which enable computing the 3D invariant position of a point with respect to our four-ball feature frame. This approach needs no camera calibration nor knowledge of camera position because the multiple views invariant are independent from the camera parameters. Therefore the cameras can be set in an arbitrary position or even move to keep centering on the observed feature frame.

Here, we compute the position and orientation of the 3D object from the two images captured simultaneously by the two digital cameras. There are six parameters, three for position and three for orientation. Our main objective is to enable the human operator to move the robot arm by moving the feature frame. We have used a set of four colored balls: red, green, blue and yellow. The red ball is at the origin of that structure as shown in Figure 63-(a). The algorithm gives the 3D position of the object with respect to the initial coordinate system composed of the basis vectors.

6.7.5 The affine invariant from multiple views

We use multiple views generated by two un-calibrated cameras to compute the 3D invariant position of a small ball. For this we define a 3D frame of reference R formed by three mutually orthogonal axes using basis vectors $\{E_i : 1 \leq i \leq 3\}$ and origin O. We assume frame R is observed with respect to a fixed frame R_0. The mapping of R to our feature frame is as follows: (1) the origin O of R is the position X_0 of red ball center, (2) the position of the edge of each basis vector E_i, for $1 \leq i \leq 3$, is the center X_i of the green ($i = 1$), blue ($i = 2$), and yellow ($i = 3$) balls, respectively. In other words, if X_i is being the 3D coordinates of the ith ball the basis vector E_i can then be evaluated as:

$$E_i = \frac{X_i - X_0}{\|X_i - X_0\|^{\frac{1}{2}}} \tag{316}$$

The position and orientation of frame R can be determined at time t by using the 3D coordinates of each ball. This gives $R(t) = \{X_0(t), E_i(t) : 1 \leq i \leq 3\}$. At time $t+1$ frame R moves (operator) to a new position and orientation defined by $R(t+1)$ which is observed with respect to the previously identified frame $R(t)$. The position of $X(t+1)$ of a ball center of $R(t+1)$ becomes:

$$X(t+1) = X_0(t) + \sum_{i=1}^{3} \alpha_i \times E_i(t) \tag{317}$$

where α_i is being the ith components of a ball of frame $R(t+1)$ that is observed with respect to $R(t)$. Parameters α_i, for $1 \leq i \leq 3$, are also the affine invariant projections of edge $X(t+1)$ over the basis vectors of $R(t)$. In other words, the coordinate of the same ball of $R(t+1)$, with respect to $R(t)$, in the 2D frame of first camera are given by:

$$\begin{pmatrix} x^1 \\ y^1 \end{pmatrix} = \begin{pmatrix} x_0^1 \\ y_0^1 \end{pmatrix} + \begin{pmatrix} e_{1,1}^1(t) & e_{2,1}^1(t) & e_{3,1}^1(t) \\ e_{1,2}^1(t) & e_{2,2}^1(t) & e_{3,2}^1(t) \end{pmatrix} . \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \tag{318}$$

where $(x^1, y^1)^t$, $(x_0^1, y_0^1)^t$, $(e_{1,1}^1(t), e_{1,2}^1(t))$, $(e_{2,1}^1(t), e_{2,2}^1(t))$, and $(e_{3,1}^1(t), e_{3,2}^1(t))$ are the projections on first camera (superscript) of $X(t+1)$, $X_0(t)$, $E_1(t)$, $E_2(t)$, and $E_3(t)$, respectively. This means that we can derive two equations with three unknowns for any point location in a single 2D camera frame. The problem of finding α_i is under-determined. A second view with known correspondence to the first view can, however, give an over-determined set of equations. Thus we have four equations with three unknowns. In matrix notation we have:

$$X = M.\alpha$$

where M ia a matrix formed by the projections $E_1(t)$, $E_2(t)$, and $E_3(t)$ over the first (upper two rows) and second camera (lower two rows), respectively. We have:

$$\begin{pmatrix} x_0^1 \\ y_0^1 \\ x_0^2 \\ y_0^2 \end{pmatrix} = \begin{pmatrix} e_{1,1}^1(t) & e_{2,1}^1(t) & e_{3,1}^1(t) \\ e_{1,2}^1(t) & e_{2,2}^1(t) & e_{3,2}^1(t) \\ e_{1,1}^2(t) & e_{2,1}^2(t) & e_{3,1}^2(t) \\ e_{1,2}^2(t) & e_{2,2}^2(t) & e_{3,2}^2(t) \end{pmatrix} . \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \tag{319}$$

The least squares solution $\hat{\alpha}$ is given by:

$$\hat{\alpha} = (M^t M)^{-1} M^t X \tag{320}$$

Thus $\hat{\alpha}$ provides an estimate of the 3D position of any given ball. We may obtain an estimate of the 3D position for each of the four balls that represent the feature frame $R(t+1)$ by simply repeating the computation of Equation 320 for each ball. Therefore the position of the feature frame $R(t+1)$ can be fully identified by the position of its origin $X_0(t+1)$ and its orientation matrix $\Phi(t+1) = \{E_1(t+1), E_2(t+1), E_3(t+1)\}$ which derive from Equation 316. Both $X_0(t+1)$ and Φ represent

differential information on position and orientation of hand motion because they measure changes with respect to previous frame $R(t)$.

In addition to 3D positional information, we can obtain 3D orientation matrix $R_{3\times3}$ information based on the 3D position information as following:

$$R_{33} = \begin{pmatrix} \frac{1}{l_1} & 0 & 0 \\ 0 & \frac{1}{l_2} & 0 \\ 0 & 0 & \frac{1}{l_3} \end{pmatrix} . \begin{pmatrix} x_g - x_r & x_b - x_r & x_w - x_r \\ y_g - y_r & y_b - y_r & y_w - y_r \\ z_g - z_r & z_b - z_r & z_w - z_r \end{pmatrix}$$

For teleoperation we need to guarantee that the identified feature frame has normalized orthogonal vectors. For this we use Gram-Schmidt orthogonalization process to produce an orthogonal set of function from the set we have computed. The Gram-Schmidt process for computing an orthonormal basis $T = \{Z_1, Z_2, .., Z_m\}$ for a m dimensional subspace W of R^n with basis $S = \{X_1, X_2, ..., X_m\}$ through the following steps. In Step 1, we Let $Y_1 = X_1$. In Step 2, we Compute the vectors $Y_2, Y_3, ...Y_m$ successively, one at a time, as follows:

$$Y_i = X_i - (\frac{X_i.Y_1}{Y_1.Y_1}).Y_1 - (\frac{X_i.Y_2}{Y_2.Y_2}).Y_2 - ... - (\frac{X_i.Y_{i-1}}{Y_{i-1}.Y_{i-1}}).Y_{i-1}$$

Note that the set of vectors $T^* = \{Y_1, Y_2, ...Y_m\}$ is an orthogonal set. In Step 3, we Let

$$Z_i = \frac{Y_i}{\|Y_i\|}, 1 \leq i \leq m$$

Then $T = \{Z_1, Z_2, ..., Z_m\}$ is an orthonormal basis for subspace W.

6.7.6 Performance evaluation

Our telerobotic system consists of a master station (client) and a slave robot station (server) which are interconnected through a 100 Mbps Ethernet LAN. The client-server system is implemented using a robust Visual Basic interface. Both client and server programs use TCP/IP sockets and API to provide the needed real-time connectivity through the LAN. The system also includes a stereo-vision system consisting implemented by using two Handycam Sony cameras as the slave robot station. We use eye shuttering glasses at the client station which allows the operator to have a real-time stereo views of the slave station. The technique used is to synch-doubling that consists of successively displaying the right and left images on a PC screen at a rate of 120 Hz. The shuttering attached hardware provides wireless infrared control of the shuttering glasses which causes the stereo effects to the operator. To preserve the quality of stereo images and their high resolution we use the LAN file sharing service as the mean to transport the visual information from the client to the server.

The vision system configuration used for the experiments consists of two PCs with a direct connection to each other. Each PC is equipped with a Firewire card that interfaces to a digital camera. The two PCs run in parallel the basic tracking algorithm. However, for the 3D part one PC transfers its local data to the other PC where the 3D affine invariant computation is performed. The result is a set of twelve differential parameters, of which three for the position and nine for the

orientation matrix. The parameters are transmitted to the slave site through the LAN to control the motion of the robot. Ideally this results in a slave robot motion that is a replica of the operator hand motion. The stereo vision allows the operator to see the slave scene and to make the necessary corrections.

Mainly our system can track the feature frame and calculate its 3D information at about 9Hz. The main reason for the delay is the time spent for (1) vision data transfer, (2) the tracking algorithm, and (3) computing the 3D affine invariant transformations.

The performance of the tracking algorithm is based on (1) evaluation of static and dynamic positioning errors for each camera, and (2) evaluation of dynamic errors in the 3D transformation.

The measurement of static errors with respect to each camera consists of evaluating the ball center position in the camera frame versus change in the ball diameter (12 mm) measured in pixels. The ball was set at a distance of 1.5 m from the camera. We used the zoom function to vary the ball diameter in the range of 5 to 65 pixels. For each camera, the average position error was 3×10^{-4} (0.1 pixels) and its upper bound was 6×10^{-4} (0.2 pixels) of the camera range. For example a workspace of 1 m leads to a static error of 0.3 mm with an upper bound of 0.6 mm.

The measurement of dynamic errors with respect to one camera consists of evaluating the ball position errors in the camera frame while moving the feature frame by the robot along a linear and a circular trajectories. The ball diameter was approximately 1.2 cm or 15 pixels. Our algorithm was implemented on 2 PCs, one for each camera, and dynamically tracking in parallel the balls. The camera frame acquisition timer is an indicator of the availability of new camera frame and used to trigger the acquisition and iterative processing of the tracking algorithm. As such the refreshing rate was 10 iterations per second. For each camera, the average position error was 3×10^{-3} (1 pixel) and its upper bound was 5×10^{-3} (1.6 pixels) of the camera range. For example a workspace of 1 m leads to a dynamic error of 3 mm with an upper bound of 5 mm. The above results are only valid when the ball speed is below 500 mm/s. The average dynamic errors increase significantly with increase in the ball speed. for example at a speed of 700 mm/s the average position error becomes 1.2×10^{-2} (3 pixels) and its upper bound was 2.8×10^{-2} (7 pixels) of the camera range.

For the 3D position measurements each of the two digital cameras is interfaced to a separate PC. Both computers continuously acquire images, and run the tracking algorithm in parallel on two computers. After computing the coordinate of the balls in its camera frame the first computer forwards its data position to the other computer where the 3D position calculation is carried out. The task of the second computer is (1) compute the affine invariant transformation and find the changes in the position (vector α) and orientation (matrix Φ) of the operator hand frame, and (2) send a real-time packet to the server (slave) station with (α, Φ) as payload.

The measurement of dynamic errors consists of computing the affine invariant transformation while moving the feature frame by the robot along a known linear and a circular 3D trajectories. The identified trajectory are shown on Figures 67 and 68. The setting is similar to the previous experiment. The upper bound on the measured error was a box of $6 \times 6 \times 6$ mm in a slave arm work of 1 m^3 when the speed on motion was at most 0.5 mps and the feature frame is about 1 m away from

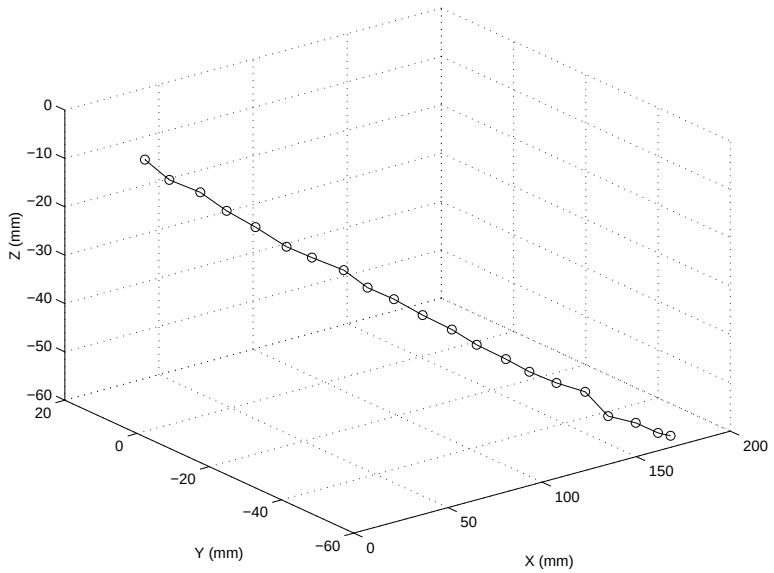


Figure 67: Two cameras tracking of a linear 3D trajectory with single-pass and lines

the cameras.

The server station continuously reads the joint position (vector $\theta_{Puma}(t)$) of the PUMA 560 slave arm, and compute the slave arm tool position and orientation $\{X(t), M(t)\} = G(\theta_{Puma}(t))$, where $X(t)$ and $M(t)$ are the position vector (3×1) and orientation matrix (3×3) of the tool frame at time t, and $G(.)$ is the direct Kinematic model of the slave arm. The Kinematic model allows localized control of multiple solution and proper processing of each slave arm singular configuration. Whenever the slave station receives from the client station a position control packet with (α, Φ) as payload it (1) computes the new slave position and orientation as $X(t+1) = X(t) + \alpha$ and $M(t+1) = M(t)\Phi$, and (2) evaluate the slave joint position $\theta_{Puma}(t+1) = G^{-1}(X(t+1), M(t+1))$ which corresponds to the new tool frame $\{X(t+1), M(t+1)\}$, and (3) sends to the slave robot the new joint position $\theta_{Puma}(t+1)$. The local servo-controller in the slave robot moves the arm from its previous position $\theta_{Puma}(t)$ to the new equilibrium position $\theta_{Puma}(t+1)$. The above scheme maps the operator incremental motion onto increments on current slave position and orientation.

The timing of the various components are presented here for the telerobotic system during a motion controlled by the operator. In this case the robot speed is fixed at 0.5 mps and the motion time varies depending on its magnitude. It is noted that computer processing time does not exceed 1 ms in all cases. The packet synchronization delay between master and slave system is about a few mms. The dominant delay component is about 0.2-0.3 seconds is due to motion of the slave robot and the processing at the client computer. The variation in these parameters is due to the magnitude of motion requested and the fact that the slave arm is controlled by a constant speed criterion.

6.7.7 Conclusion

In this section we presented a telerobotic system that consists of a real-time vision-based tracking algorithm (client) and a slave robot (server) which are interconnected

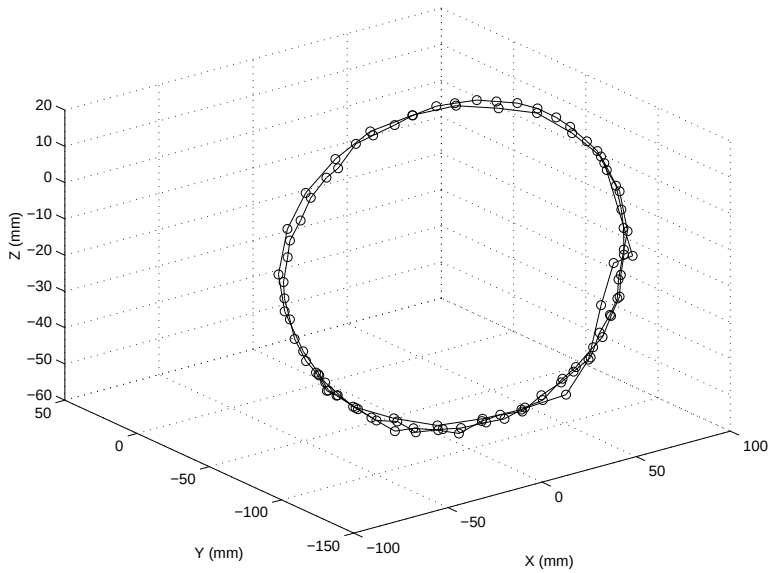


Figure 68: Two cameras tracking of a circular 3D trajectory with multi-passes and lines

by using a LAN. The tracking system monitors a feature frame that is held by the operator hand. The algorithm determines the 3D position of operator hand by using uncalibrated cameras together with the affine invariant property. The features of the proposed systems are (1) a metric for color matching to discriminate the balls from their background, (2) a uniform and spiral search approaches to speedup the detection, (3) tracking in the presence of partial occluding, (4) consolidate detection by using shape and geometric matching, and (5) dynamic update of the reference colors. We presented a telerobotic system based on a complete kinematic mapping from operator hand motion to slave robot joint space. In the evaluation the experimental analysis indicated that the average static error in a workspace of 1 m is 0.3 mm (0.6 mm upper bound), that of the dynamic errors is 3 mm (5 mm upper bound), and 3D errors were contained in a box of $6 \times 6 \times 6$ mm if motion speed is below 0.5 mps. Analysis of delays in the proposed telerobotic real-time control scheme indicated that the dominant delays are due to the mechanical system and the network.

6.8 Design of a 6-dof master arm

A telerobotic system is a tool allowing sensorial transfer (telepresence) of a human operator into a remote working site that can be a hostile environment. The main objective of this task is to design a six dof master arm which is light, back-drivable, easy to maintain and with low friction. The arm will be interfaced through a LAN with a PUMA 560 slave arm. It is expected to allow an operator handling of its terminal part to control three positions, three orientations, three forces, and three moments, i.e. a set of twelve independent parameters.

6.8.1 Design criteria

The following are the main criteria considered during the design of the master arm.

1. Low impedance: the arm must have a small enough overall impedance (inertia) to allow the operator to freely and ergonomically move his arm at natural speeds approaching 1 mps with an acceleration of 1 g.

2. Dexterity: the arm must permit a man-machine interface (dexterity) that is as transparent as possible to ease the tele-manipulation task and its feedbacks. A dexterous master arm needs to have back-drivable transmission system that is activated by using electrical DC motors to ensure fine overall control while providing a reasonable level of force transmission and display to the operator hand. The back-drivability is based on the use of stiff wire-based transmission system, low friction, and adequate distribution of the motion reduction wheels. Ideally it is only limited by the equilibrium defects. The distribution and arrangement of the arm dofs and links should facilitate the spontaneous association of an effective combination of movements and their corresponding force display. Structure efficiency should be guided by a high degree of mechanical transparency, or iso-impedance with respect to the motion directions, felt by a user operating the master arm.

3. Maintainability: components prone to wear or failure, such as the steel wires, must be easily visible and accessible. Covers, panels, housings, must be easy to remove and replace. Components that must be serviced must be located in accessible locations.

4. Dynamic transparency: systems should be designed to improve dynamic transparency and effectiveness of kinesthetic mapping from operator hand motion to slave motion.

5. Light weight: the arm must be made from materials such as Aluminum alloys and all standard parts should be selected for their light weight.

6. Strength: the elements making up the structure must be able to withstand the applied static and dynamic loads without yielding or buckling.

7. Ergonomics and Aesthetics: human arm characteristics should be applied to the design. The operator should feel comfortable when dealing with the different functions of the arm. The visual appeal of the arm should be pleasing to the operator.

6.8.2 Conceptual design

The first steps of the design consisted on evaluating existing master arms and their means of motion transmission. The experimental model consisting of a 6 dof simple articulated structure (SAS), described earlier, was utilized to gain some experience on control and architecture of master arms. Several other existing design configurations were developed and evaluated. This lead to the selection of an arm with a serial structure comprising six revolute axes. The serial structure with 6 dof will have two independent sub-structures consisting of (1) 3 dofs (first) positioning mechanism for operator hand, and (2) 3 dofs (last) orientation mechanism. All dofs are revolute. The assembly of the master arm and its parts are illustrated in the drawings of Appendix A.

Decoupling the translation from rotation is aimed at improving the kinesthetic mapping from operator hand motion to slave motion as well as reducing arm impedance transparency and effectiveness. Several power and motion transmission systems were considered and two systems were retained: (1) a hybrid system and (2) a completely flexible system. The following gives a description of the setting for the above transmission systems:

1. Hybrid (flexible and rigid) motion transmission systems were considered first. In this design, the last two degrees of freedom (dofs 5 and 6) were to be driven by a bevel gear differential system. The transmission of motion from the motors was achieved by timing belts and the intermediate degrees of freedom were driven by flexible single strand wire ropes. A model consisting of a three bevel gear differential system was built and tested. It was found that the friction in the gears and the need for perfect adjustment of the three-gear system prevented this solution from being adopted.

2. Flexible wire drive system. The second alternative was that of replacing the gears with single strand flexible steel rope driven systems. This is expected to result in a lighter mechanism with lower friction and inertia improving the quality of operator hand motion measurements as well as display of force feedback.

Before the adoption of the flexible steel rope system solution the possibility of wire slippage during motion at high torque was investigated. An experimental single dof model was designed and built for the purpose. The system consisted of two motion transmission loops and an arm as shown on Figure 69. In the first loop (primary loop), the power from the motor is transmitted through a timing belt drive allowing a tenfold speed reduction and torque multiplication. The second loop (intermediate or final loop) consists of the wire driven mechanism to be tested. Initially, the single strand flexible steel rope was wrapped three times around two threaded wheels, one at each end of the loop. Testing involved lifting a weight fixed at the end of the 500-mm long arm at different speeds. It was observed that some slippage occurred between the wire and the threaded wheel at high torques. To prevent this, the wire is introduced in the middle thread, through the wheel and a jam screw is used to lock it with the wheel. Evaluation of the modified design proved that no slippage occurred even with torques of the order of 10 N.m.

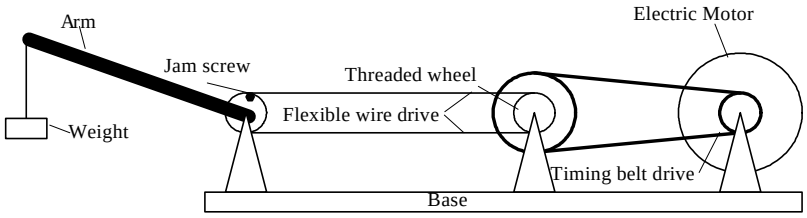


Figure 69: Schematic representation of experimental single-loop wire model

The above tested closed-loop system was thus retained as the solution of choice for torque transmission. The design for maintainability was addressed next. It was decided to exclude all complicated designs in which motion transmission systems utilize one single closed-loop wire connecting the motor shaft to the link. This type of system is difficult to maintain and presents a single point of fault.

The transmission mechanism based on multiple independent closed-loops will reduce the maintenance effort to a localized area and will improve system reliability. To ensure back-drivability and minimize inherent impedance at least the last three dofs of the master arm must have a small gear ratio from the motor shaft to link manipulated by the operator through a few closed-loop flexible transmission wires. All intermediary and final motion transmission loops will be wire driven while the primary loops will consist of timing belt drives. Torque multiplication and speed reduction is obtained at the primary loop level.

The arm mechanism must have intrinsically low impedance (low inertia) and be compliant (or elastic) to eliminate the need for contact sensing and to guarantee good force display. To further improve the systems' low impedance a dynamic model will be used. This model is needed to (1) hide the imperfection of the mechanical system from being sensed by the operator, (2) improve the quality of interaction between operator, master arm, and slave arm, and (3) minimize overall contact impedance and limit the need for sensing.

6.8.3 Embodiment design of the master arm

The conceptual design steps led to the selection of an arm architecture made of a serial structure with 6 dof having two independent sub-structures as described above. Taking into account the above required functions and criteria; a preliminary design of the master arm was performed. The activities at this stage were devoted to: (1) establishing and finalizing the arm architecture, (2) determining the configuration of parts that will satisfy the required functions, and (3) quantifying the important design parameters.

Arm architecture

Once identified, the physical elements (links and joints) were arranged in such a way to carry out the desired functions described above. Several iterations were performed. The modular architecture style was selected. To achieve this, each block

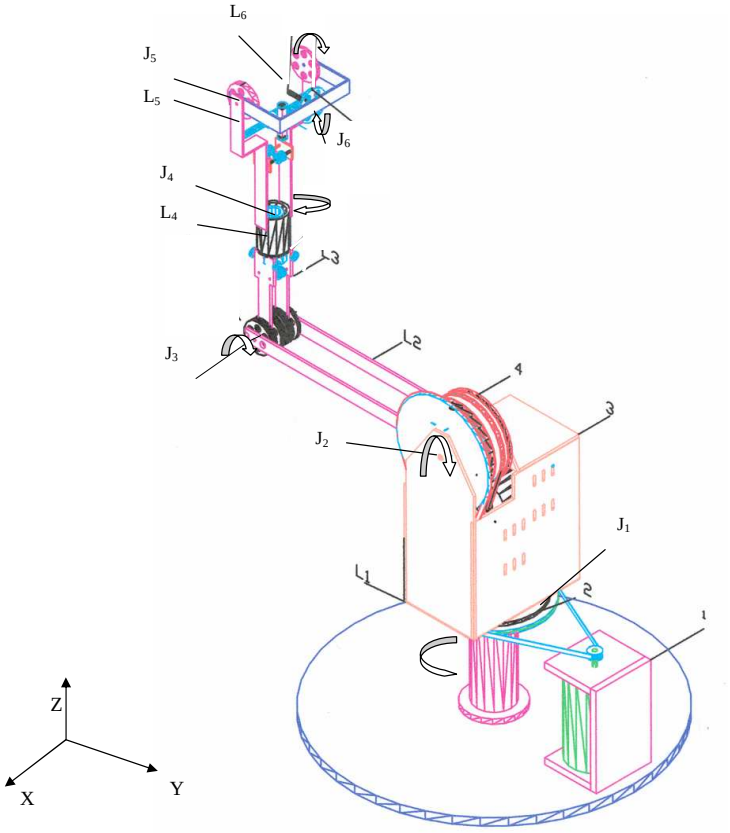


Figure 70: General Architecture of Designed 6 dof Master Arm

is associated with one degree of freedom and each degree of freedom is driven by an independent motion transmission system. Furthermore, as mentioned above, the first 3 degrees of freedom are for translation and the last 3 are for operator hand orientation. Decoupling translation from rotation is aimed at improving arm impedance, dynamic transparency and effectiveness of kinesthetic mapping from operator hand motion to slave motion.

The drawing of Figure 70 depicts the different components of the basic structure of the designed master arm. It is composed of 6 main serial links, indicated by letter L, with six revolute joints, designated by letter J. All joint axes are driven by high-performance DC motors through flexible transmission systems. The degrees of freedom are indicated by the arrows on the above figure. In this figure and all the other figures the wire loops are not shown.

All axes of joints 3, 4, 5, and 6 will be wire-driven. Each simple independent loop will consist of two grooved (threaded) wheels and a thin flexible steel cable. Each wire will be wrapped one and a half turns around the corresponding wheel to minimize slippage. In the final wrap, the flexible cable will be introduced through a specially designed inclined through-hole to be completely restrained from any slippage by a tightening screw device. Each of the driving threaded wheels will be tightly mounted on the corresponding toothed wheel (sprocket) which is driven by a brushed DC motor through an extra-light timing belt.

The belt drive systems for dof 3, 4, 5, and 6 are designed to allow a tenfold

multiplication of the driving torque. This is achieved by using a 150 mm pitch-diameter sprocket with a 15 mm pitch-diameter small sprocket. The small driving sprocket is mounted directly on the shaft of the motor using set screws. Given the higher required torque for dof 1 and 2 a larger sprocket is selected for driving the corresponding links. For these two dof the belt drive is designed to deliver a torque 7.5 times the motor torque. It is worth mentioning that these two links are directly fixed to their corresponding sprockets, i.e. only the primary loop is used. For dofs 4, 5 and 6 a complex system of threaded wheels, cylinders, cables and guiding pulley systems are designed to allow for quarter-twist drive motions and to minimize cable elongation during rotation of links 4, 5 and 6.

Because steel wire ropes are prone to failure, the threaded wheels and the sprockets in joints 2 and 3 are designed to allow access to all the cables in each loop for easy maintenance. Thus the cables can be independently replaced without having to disassemble the rest of the elements.

All the five DC motors (for dofs 2, 3, 4, 5 and 6) are mounted directly below the large sprockets. The positioning of the motors is made in such a way to minimize inertia. The supports for the motors can be adjusted in both horizontal and vertical directions. This allows alignment of sprockets and tensioning of belts. The five motors, timing belts and motor supports are enclosed in an aluminum housing (3) which will be mounted on the top of the large sprocket (2) of dof 1, driven by the sixth motor. The aluminum box (3) is designed to allow for easy access to motors and belts for maintenance and adjustment. The sixth motor will be mounted in bracket (1) on the platform as shown in Figure 70. Each motor will be equipped with an encoding system for control and force sensing.

The sprockets and pulleys in joint 3 are mounted on a solid steel shaft of 5 mm diameter. The strength analysis of this and all other shafts were performed under the conditions of static and fatigue loading. The analysis involved the bending stresses introduced by transverse loads and the shear stress induced by applied torques. In static conditions and using the distortion energy theory, the safe diameter of the shaft is calculated as follows:

$$d = \left[\frac{16n}{\pi S_y} \left(4M^2 + 3T^2 \right)^{0.5} \right]^{1/3} \tag{321}$$

where d is the tentative diameter at the critical region, n is the safety factor, S_y is the yield stress of the material, M and T are the resulting bending moment and torque at the critical point. The value of d obtained from this analysis is used as a first trial to estimate the endurance limit for fatigue analysis.

Considering that all the shafts will be working under cyclic loading. The minimum safe diameter should thus be determined under fatigue loading conditions. One of the most useful relationships used for this purpose is that combining the distortion energy for stress and the Goodman line for fatigue strength. Considering completely reversed bending and torsion, the minimum diameter can be estimated from the reduced form of the above mentioned theory [98, 99]:

$$d = \left[\frac{32n}{\pi} \left[(\frac{K_f M_a}{S_e})^2 + \frac{3}{4} (\frac{K_{fs} T_a}{S_{ut}})^2 \right]^{0.5} \right]^{1/3} \tag{322}$$

where K_f and K_{fs} are respectively the bending and torsional fatigue stress concentration factors, M_a and T_a are respectively the resulting alternating components of the bending moment and torque at the critical point, S_e is the real endurance limit of the shaft, and S_{ut} is the ultimate strength of the material.

Following strength analysis the lateral deflection analysis of shafts is also performed. The shaft is supported at both ends by deep groove ball bearings. Furthermore, and in order to minimize friction each sprocket and corresponding threaded wheel system is press-fit on two adjacent sealed ball bearings. The two bearings are positioned in a way to improve the stability (minimize lateral motion) of the wheels.

Completely sealed self-lubricated deep groove ball bearings are employed for all rotating elements. The bearings are selected from commercial catalogs [100] to have the required load capacity, satisfactory fatigue life and mainly the desired lightweight. The material to be used must be made of pure or light Aluminum alloy. All unnecessary material is removed to reduce weight and avoid increasing in the inertia.

Components configuration and design parameters

The following section deals with the description of the three main sub-assemblies that make up the designed master arm. It includes details concerning the selected and designed mechanical elements as well as their integration taking into account the objectives, functions and criteria cited above. The description of the sub-systems is done assuming that the motion occurs in the forward direction where the motors are receiving input from the force-sensing element in the wrist of PUMA slave arm. The reader should bear in mind that the system is back-drivable; i.e. the operator will be controlling the motion.

Structure for DOF-1

The designed mechanism allows for -180^o +180^o rotation of the whole arm structure around Z-axis as shown in Figure 70. However the useful range for the master arm is [-90^o,+90^o]. In the present design of the prototype it has been decided that the whole arm will be a tabletop type. Figure 71 illustrates the assembly drawing of the structure of DOF-1 and the master arm base (5). The motor (7) with a torque of 3 N-m and a speed of 2000 rpm will be supplying the rotational motion for the whole structure of the arm. The selection of a driving sprocket (2) with a pitch diameter of 15 mm and a driven sprocket of 225 mm (3) allows a gear ratio of 15 and an amplification of the torque by the same factor. Power transmission is ensured by an extra light (XL) rubberized fabric timing belt (4) with a standard pitch of 5 mm (0.2 in). The choice of the XL timing belt is based on the criterion of back drivability. Moreover, this type of belt does not stretch nor slip which will result in the transmission of power with a constant angular velocity. Timing belts are also known to have efficiencies varying from 97 to 99%. The circumference of the belt and the drive center distance were estimated such that at least three teeth of the small sprocket will be engaged. The wrap angle of small sprocket, which allows for this condition is given by:

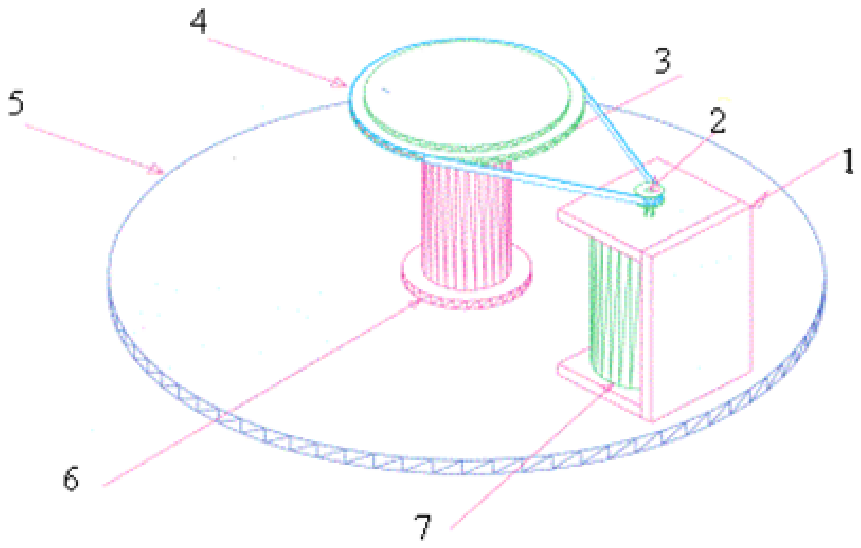


Figure 71: Structure of DOF 1

$$\theta_d = \pi - 2sin^{-1}(\frac{D-d}{2C}) \tag{323}$$

where D is the diameter of the large sprocket, d is the diameter of the small sprocket, C is the center distance, and θ is the contact angle.

The driving sprocket selected from a commercial catalog, was bored and mounted on the shaft motor using two setscrews. The motor shaft diameter was also reduced to fit in the sprocket bore. The teeth of the large sprocket were machined. A delicate procedure was established to arrive at an integral number of teeth. This procedure was later applied to all of the sprockets manufactured in the Mechanical Engineering Department Workshop and used in this design. This sprocket is mounted on a vertical shaft that is supported by two deep-groove ball bearings, one at the top and another at the bottom. These bearings were selected to withstand both radial and thrust load and to give a satisfactory life. These bearings are press-fit on the shaft and push fit in the cylindrical housing (6) which is bolted to the circular base (5). The DOF-1 driving motor (7) is also mounted on the same base using bracket (1).

Structure for DOF-2

This mechanism is designed to allow for an out of plane rotation (almost -150^o + 150^o) of link 2 (L2 in Figure 70) around the X axis. However, the useful range for the master arm is [0^o,-150^o]. The elements that help realize this motion are contained in the frame indicated by the number 3 in the figure. Driving is ensured by a brushed DC motor with a rated peak torque of 3 Nm and a speed of 1700 rpm. This motor drives link 2 by a timing belt system similar to that described above. The sizes of the small and large sprockets are the same as those for dof 1.

As shown in Figure 70 and 72, link 2 (L2) consists of two $50 \times 5mm^2$-aluminum alloy rectangular plates. The plates of total length 465 mm have been emptied by

reducing most of the width from 50 mm to 28 mm and by drilling holes along the plate length. This resulted in weight reduction and gave the link an appealing look. The detailed drawings of the plates design are provided in appendix B. The lower end of the left hand plate is directly bolted to the driven sprocket using two M3 flat head cap screws. To minimize friction and provide lateral rigidity for the structure, the plate-sprocket assembly is mounted on two sealed deep-groove precision ball bearings push-fit on the steel shaft of joint 2 (Figure 70). The same type of bearing is used to mount the second plate of the link. The distance between the two plates of L2 is dictated by the elements of joints 2 and 3 (Figure 70 and Figure 72)

Structure for DOF-3 and 4

A closer view of the structures for DOF 3 and 4 is illustrated in Figure 72 and 73. The range of angular rotation of link 3 (L3) is about -150^o + 150^o around the X-axis as shown in Figure 72 and 73. However the useful range for DOF 3 of the master arm is [0^o,150^o]. The rotation is realized by two loops of flexible power transmitting elements. The primary loop consists of a timing belt drive system similar to that described above for DOF 2.

In order to transmit the motion of the driven sprocket of joint 2 to link 3, a flexible stainless steel wire rope (cable) of 1 mm^2 section area is selected. This cable is wrapped over two 60-mm pitch diameter threaded wheels. As indicated above, each wire will be wrapped two times around the corresponding wheel to minimize slippage. In the middle wrap, the flexible cable is introduced through a specially designed inclined through-hole to be completely restrained from any slippage by a locking screw device. In order to completely imbed the cables three rounded thread types of depth equal to rope diameter i.e.1.2 mm were machined. This required a thread pitch of 2 mm and a wheel width of 6 mm. The detailed drawings of the threaded wheel is provided in appendix B.

The driving threaded wheel in joint 2 is bolted to the driven sprocket by 3 M3 flat head cap screws. This assembly is again mounted on the shaft of joint 2 on two sealed deep-groove precision ball bearings. The driven wheel is bolted to the right hand side plate of link 3 by 2 M3 flat head cap screws. This wheel and the plate rotate around the shaft of joint 3 (around X-axis) via two sealed deep-groove precision ball bearings. The left hand plate of link 3 is also mounted on the same shaft with a bearing. The upper end of the plates that make up link 3 are designed in a way to accommodate the guiding pulleys for dof 4, 5 and 6. The inner surface of the upper part of the plates is made round to fit on the inner cylinder C1 of dof 4 (Figure 73). These plates are fixed to the cylinder by 4 screws to the cylinder. The dimensions and shape of link 3 are given in the detailed drawings in appendix B.

DOF 4 is a -150^o + 150^o twist around Z-axis of hollow cylinder (C2) and the 2 plates that are fixed to it at one end and to link 5 at the other around cylinder (C1); it is the first mechanism for orientation. However the useful range for DOF 4 of the master arm is [-100^o,100^o]. This motion is realized through three loops. The primary loop is composed of a timing belt drive which transmits the power from the motor to the corresponding large sprocket in joint 2 with a gear ratio of 10. The second loop is a wire rope drive with a gear ratio of 1. In joint 3 the wire is wrapped around a 60-mm pitch diameter right hand threaded wheel that is solidly

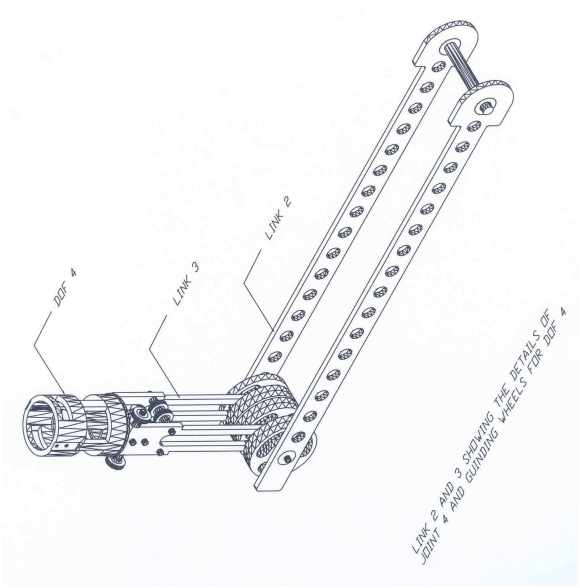


Figure 72: Structure for dof 3 and 4

joined to another threaded wheel with a left hand thread as illustrated in Figure 73. The third loop is also a wire transmission loop starting at this wheel and driving cylinder (C2) of Figure 73. Given that the axes of of joints 3 and 4 are at right angle a quarter-twist drive was designed and implemented. To achieve this, the driving wire is run through a small guiding pulley system where the pulleys are mounted at an angle such that the wire at the exit is tangent to the cylinder (C2). Cylinder C2 is designed to rotate around cylinder (C1) through two deep-groove precision ball bearings. Both these cylinders are manufactured from aluminum alloy rods and all unnecessary material was removed to reduce the links weight. The details of geometry and dimensions of cylinders C1 and C2 are illustrated in the drawings of Appendix B.

Structure for DOF-5 and 6

Figure 74 is an assembly drawing for the mechanism that allows the realization of degrees of freedom 5 and 6. The plates that link this mechanism to the preceding are part of link 4 that are connected to cylinder C2 by a specially designed bolt joint. Link 4 is 274 mm long. It is made of plates $20 \times 3\ mm^2$. The plates are bent to accommodate the mechanisms for DOF 5 and 6. The detailed drawings for these plates are given in Appendix B.

Dof 5 is the rotation of L5 around the X-axis. This link which has a C-Shape is made of an aluminum plate 20x3 mm^2 is mounted on Link 4 as shown in Figure 74

and allows the operator's hand to oscillate around the wrist's transverse axis by an angle of 180 . Dof 5 is realized using 4 loops with the first being a timing belt drive similar to that described above and the last three are wire rope drives. The 4th loop allows the transmission of motion from the left hand side of the 20-mm pitch diameter threaded roller to the 60-mm threaded wheel that is fixed to link 5 by two flat head screws.

To realize dof 6 a fifth loop is added; allowing the handle (L6) to rotate 180^o around the Y-axis as shown in Figure 70. The operator's hand controls the whole arm by manipulating the sixth link. The design is made such that the axes of rotation of links 4, 5 and 6 are concurrent at the center of the operator's hand. The 6th degree of freedom is driven by 5 loops. The first of these loops is the timing belt drive and the last four are wire rope transmission systems. The second system transmits the rotational movement from joint 2 to 3, the third conveys the motion from joint 3 to joint 4. Both these loops use 60-mm threaded wheel system. In joint 4 two opposite hand threaded 20-mm rollers are mounted on a single shaft. The wires transmitting the rotation of joint 3 to joint 4 are guided through the hollow cylinder (C2 of Figure 73) using two pairs of small pulleys. The distance between the pairs of guided wires is calculated such that the wire elongation resulting from a full revolution of the mechanism is minimal. The optimum separation distance was found to be 10 mm. The fourth loop is designed to transmit the roller motion to the threaded wheel on the right. The fifth loop consists of two threaded 60-mm wheels and two guiding pulleys. The detailed drawings of the rollers, shaft and pulleys are given in appendix B.

To minimize manufacturing time and cost all of the guiding pulleys have the same dimensions and details concerning the grooves and other shapes. All the threaded wheels of joints 3, 4, 5 and 6 have been designed to have the same dimensions and the thread geometry is kept constant. In manufacturing most of the parts close tolerances were respected.

6.8.4 Assembly of the master arm

Given the limited time for design, manufacturing and implementation of the system, several tasks were performed at the same time. Prototypes of each subassembly are manufactured and tested. Later the complete prototype was assembled and evaluated. It should be mentioned that except for wiring operation which requires some time, the rest of the components were easily assembled. The assembly of the complete master arm prototype is shown in Figure 70.

In order to test the functions of the complete arm, the base is mounted above the operator's shoulder allowing him to control it while sitting in an ergonomic way. All the links were designed according to lengths of arm and hand of an adult.

The implementation consisted on using this arm as a master arm that controls the motion of the slave arm (PUMA 560 slave arm) to transmit the signal through a number of position sensors that were selected and mounted on each motor shaft.

The first evaluation operation consisted of testing force sensing feedback, pouring a liquid in a container, and peg-in-hole insertion, assemble of a water-pump, etc. These evaluations were discussed in detail in the Sub-Section 5.11.

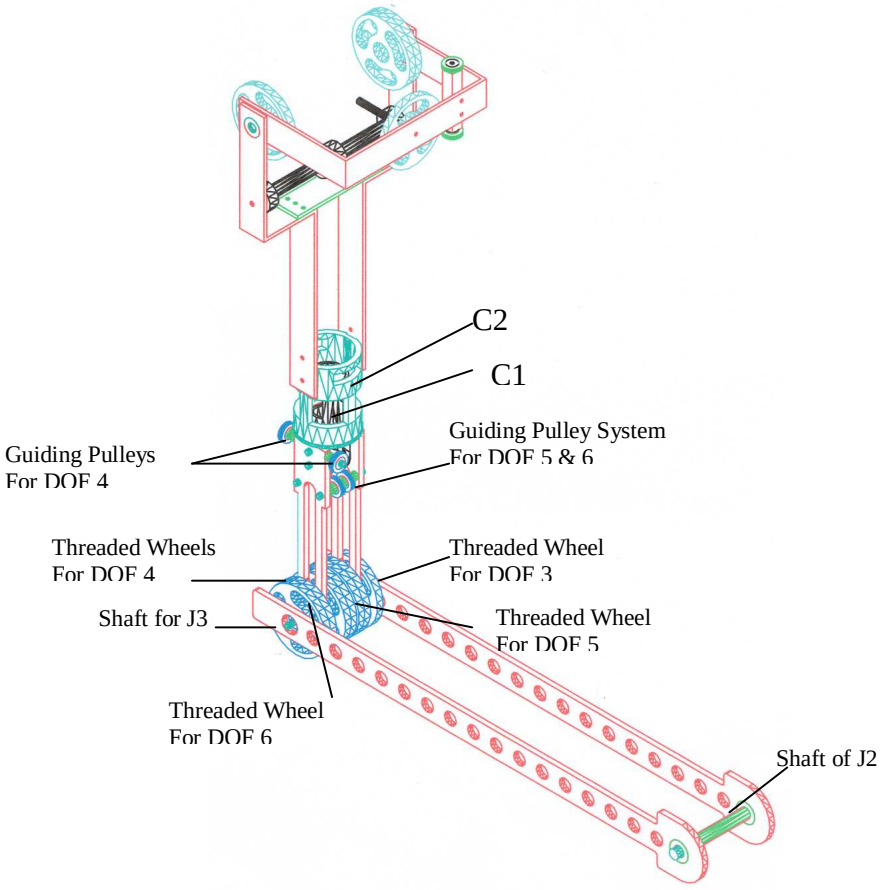


Figure 73: Assembly Drawing of dof 3, 4, 5 and 6

6.8.5 Alternative design for DOF 1 and Joint 2

Difficulties were encountered during a number of operations of the system and patching design and manufacturing were performed. Tension in transmission wire ropes was adjusted several times. The problem of the motors rotating with link 2 made system inertia high violating the low impedance criterion. To overcome this it was decided to come up with a design configuration where all the motors will be anchored at the base. Here two major problems are encountered. The first being the transmission of power from motors to the different links. This difficulty was overcome by using cables instead of belts. This was expected to have a drawback associated to the excess elongation of wires during transmission of high torque especially for dofs 1 and 2. To study this effect an experimental model was manufactured and tests were run proving that the prediction was correct. The 1 mm diameter wire was replaced by a thicker wire (1.5 mm diameter) resulting in lower elongation and less sagging of steel wire rope. The wire elongation was reduced by more than half. This thicker rope was thus selected to drive links 1 and 2. In this model the driving sprocket was replaced by a threaded roller and the driven sprocket by a large threaded wheel. The same procedure, described above, was used to fix the wire rope to the threaded wheel.

The second difficulty of the design is associated with the positioning of the electric

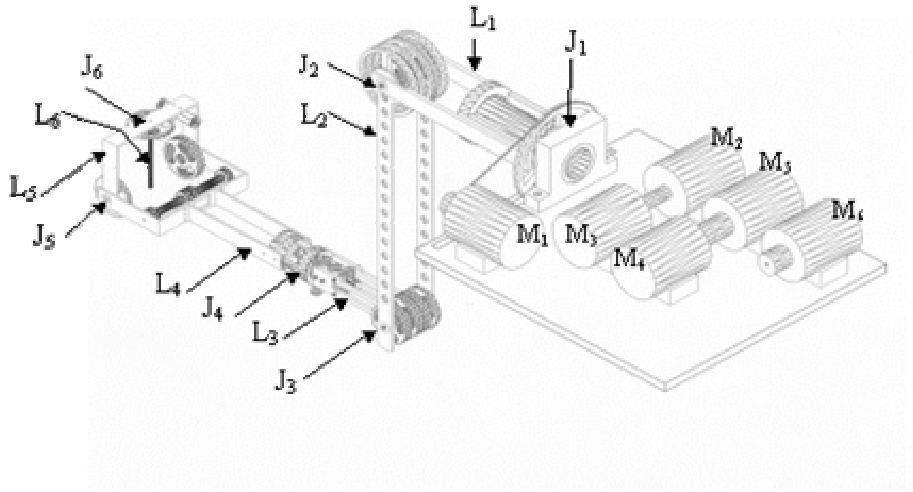


Figure 74: New Master Arm Design

Feature	Link 1	Link 2	Link 3	Link 4	Link 5	Link 6
Link length (mm)	180	420	370	95	0	0
Angular Range	[-90^o,90^o]	[0^o,-150^o]	[0^o,150^o]	[-100^o,100^o]	[-90^o,90^o]	[-100^o,100^o]
Inertia of Link Body (Nm^2)	0.0192	0.543	0.063	0.009	0.006	0.003

Table 2: Some mechanical and geometric features of the master arm.

motors. The positioning of the motors for dofs 2, 3, 4, 5 and 6 and the choice of driving roller sizes was done in way such that the wires will be parallel with spacing that allows for system rotation without contact between neighboring wires. The smaller diameter rollers are mounted in front and are used to dive links 2 and 3; requiring higher torque. A system of pulleys is designed to guide these wires at the exit of the cylinder. DOF 1 is composed of a single loop as shown in Figure 74. In this figure it can be seen that the motors M1 through M6 are mounted on a stationary platform. All small driving sprockets are replaced by threaded rollers and all large sprockets are replaced by threaded wheels allowing similar torque multiplication. The rest of the joints and links remain unchanged.

6.8.6 Mechanical features of the master arm

Table 2 shows some mechanical and geometrical features of each of the six links of the master arm. These are: (1) the link lengths, (2) the range of angular rotation of each link, and (3) the moment of inertia of the link body computed at the corresponding joint. The inertia at the link body includes the link body, the attached wheels at end of link, and the transmission wires traversing the link. The sum of weights of link bodies 3, 4, 5, and 6 is about 800 gs and that of link body 2 is 1.2 kgs

including all attached transmission wheels. Depending on arm configuration the operator hand needs to balance at most 400 grs. To balance the above gravity a spring was used between joint 2 and a selected point on link 3. Other alternatives were also considered. For example, using active gravity compensation in which the computer evaluates the gravity forces for a given configuration and generates the compensation forces on the attached motors.

At joint 5, the width and length of the master arm handle is 160 mm and 95 mm, respectively. Joint 5 is linked to joint 6 using a segment that is 85 mm long and 150 mm large. The operator handle itself is 150 mm long.

The arm is basically anthropomorphic with a wide workspace. The master arm workspace does not constrain the operator arm in any direction. The operator who is holding the master arm can freely move his arm by about ± 400 mm to the left and to the right, ±300 mm forward and backward, and ± 400 mm up and down. The arm workspace is about $0.8 \times 0.6 \times 0.8 = 0.384\ m^3$. This is considered as a large workspace when compared to existing commercial master arms.

6.8.7 Master arm installation

The master arm structure was designed at the Robotics Lab, Computer Engineering Department, KFUPM. However, all the necessary assembly and detailed drawings for this structure as well as its manufacturing were made in the Mechanical Engineering Workshop.

Following its manufacturing the master arm was installed in the Robotics Lab, Computer Engineering Department, where the following interfacing and testing tasks were performed:

1. Flexible rope wiring of all master arm DOFs between motors and links,
2. Operator handle design, installation, and operation,
3. Electronic control using installed position sensors, power control, overall motor and sensor wiring, and computer interfacing using specialized PCI I/O cards,
4. Interfacing to the Distributed Component Client-Server System,
5. Development of its direct and inverse kinematic models, programming, testing, and debugging,
6. Development of the client program, testing, and debugging,
7. Testing of the master-slave motion system, debugging, and tuning overall system.

Currently, the master arm is properly operating and was extensively used as an effective man-machine interface in carrying out a number of telerobotic tasks like the peg-in-hole insertion, pouring water, assembly of a water pump, and wire-wrapping of an electronic circuit.

6.9 Comparison to others

6.9.1 Comparing software system architecture

Even et. al [101] describes a computer aided telerobotic system as an integrated planning scheme including interactive 3D modelling, programming, and execution. The proposed telerobotic framework may provide the above system a pervasive network connectivity and mobility as well as a structured software framework for implementing real-time interactions.

In [83] a client-server framework is designed using VB 6.0 and custom protocol TCP ActiveX controls. This exposes TCP read/write operations to application, custom protocol, TCP ActiveX control, and Windows Sockets etc. While in a distributed setup, the components directly communicate with each other through windows sockets using .NET Remoting. This difference is achieved by using the distributed component based approach in place of TCP based custom protocols.

An object oriented distributed application (OODA) could also use JAVA that provides RMI as remote calling function, graphics services, and 2D and 3D support. Hu et. al.[82] proposed JAVA for network interfacing and video as well as the use of C++ for the robot controller for Internet-based telerobotic System. Teresa[81] used Java-based for frame grabbing as compared to the use of DirectShow in our scheme.

Yeuk et. al [60] developed a distributed infrastructure based on JAVA and DCOM. They used JAVA for database connectivity and path planner GUI(Graphic User Interface) and DCOM for network connectivity. MS VM(Microsoft Virual Machince) is proposed to bridge the gap between the two. Compared to [60], .NET framework is directly used for all GUI development as well as the core system components making it a unified solution. This avoid the use of middleware services like MS VM within the framework. JAVA and CORBA are intended to be cross-platform environments thus requiring lot of JIT(just-in-time) compilation and virtual machines to interpret code on different operating systems. In addition they provide no support for hardware-accelerated graphics APIs that are critical for live video visualization on PCs.

We proposed an object-oriented distributed component framework and NET remoting for (1) remote procedure call, (2) hiding details of network interface, (3) an automatic notification and data messaging mechanism between client and server. .NET does not require components registration thus breaking the interdependency in the development phase. **To develop off-the-shelf components and programs we used Windows 2000 because of support provided for hardware accelerated graphics, .Net remoting, and thread scheduling and priority needed for multi-threaded execution. .NET may be used for off the shelf real-time applications as it casts off every possible overhead. .NET has embedded type signatures which allow component debugging across different languages, a missing feature in JAVA and CORBA. Process variables like real-time sensor data and robot-states are relayed to the client-side using implicit inter-component communication**.

6.9.2 Comparing telerobotic system delays

Compared to [102, 101] the proposed telerobotic framework provides the above CAT a network connectivity software framework for commands, stereo vision, and force feedback which augments its pervasive aspects. Here force feedback is mapped to current orientation of operator hand [103] which is easily integrated by the human brain in achieving mechanical tasks [104] at large and small scales.

The video transfer rate achieved by Teresa[81] is 1 frame every 3 seconds for a single image of 16 bit color depth over the internet. The Java-based frame grabbing software takes one second for an image to move from camera to DRAM as compared to a mean value of 24 ms obtained by our approach using DirectShow. Compared to [60], .NET framework is directly used for all GUI development as well as the core system components making it a unified solution. This frees us from using intermediatory services like MS VM within the framework.

Huosheng et. al.[82] proposed JAVA for network interfacing and video as well as the use of C++ for the robot controller for Internet-based telerobotic System. In a LAN setup, Huosheng et. al. quote a transfer rate of 9-12 fps with time delays less than 200 ms for a single image of size 200×150 pixels. This is to be noted that the images are not bitmap but are compressed using JPEG compression technique. In comparison to this, our stereo video client-server transfers *two uncompressed* images *(stereo frame)* of size 288×360 pixels at a rate of 11.74 fps with a delay of around 87 ms only.

Al-Harthy[70, 83] implemented a client-server framework using VB 6.0 and TCP ActiveX controls. In the case of custom protocols like Al-Harthy's, the TCP read/write operations are very slow because of the many software layers involved such as Application, Custom protocol, TCP ActiveX control, and Windows Sockets etc. It takes 55 ms for a command signal (48 bytes) to reach from client to server. In our case a force packet consisting of 6 double values (6×8 bytes = 48 bytes, same size) took about 0.7 ms in the absence of stereo video data and 1.1 ms in the presence of video stream. While in a distributed setup, the components directly communicate with each other through windows sockets using .NET Remoting. This difference is achieved by using the distributed component based approach in place of TCP based custom protocols.

In a typical scenario when both client and server use .NET based components with TCP channels, highly optimized data transfer is obtained [84]. TCP Channel uses a default binary formatter which serializes the data in binary form and uses raw sockets to transmit data across the network. This method is ideal if the objects are deployed in a closed environment.

It is also worth noting that if the serialization of capturing and transferring-over-LAN threads is modified by thread manipulation on the server, an inter-arrival delay of around 55 ms can be achieved while utilizing nearly 90% of the bandwidth. By creating independent threads instead of serial threads in the process of capturing and data transfer, we are able to obtain a mean inter-arrival time of 58.57 ms.

Internet-based telerobotic System [82] can be implemented using JAVA for network interfacing and video and C++ for the design of the robot controller. In a LAN setup, a transfer rate of 9-12 fps with time delays less than 200 ms for a single image of size 200×150 pixels is reported. Video images are compressed using JPEG

technique. The Java-based [81] frame grabbing software takes one second for an image to move from camera to main-memory as compared to a mean value of 24 ms obtained by our approach using DirectShow. The reported video transfer rate is 1 frame every 3 seconds for a single image of 16 bit colour depth over the Internet.

A portable real-time telerobotic interface between master and slave arms is proposed using *.NET Remoting based Distributed Components*. The advantages of .NET are (1) ease of deployment to work across firewalls,(2) compiles the source code into platform-independent bytecode [105], and provides highly optimized data transfer [84] for symmetric configuration for the client and server. Design independency in the master and slave modules lead to mapping the operator hand motion and force feedback to the the remote tool. Overall distributed framework and design independence improves the portability and modularity of the proposed telerobotic system. In the proposed approach, a multi-threaded execution is proposed to allow pipelining of (1) processing and (2) network transfer times. In comparison to the above results, the proposed stereo video client-server system transfers uncompressed stereo frames of 288×360 pixels at a reference rate of 17 fps and a total time of 83 ms. Other achieved sampling rates are 76 Hz for force feedback and 50 Hz for operator commands. Analysis of delays in three campus routes indicated that network routers, buffers, switches, and links do incur negligible delays to the above packets. However, random surge in instantaneous network utilization even for short periods of time may cause severe degradation in telerobotic systems operating on public networks.

6.9.3 Comparing telerobotic tools and teleoperation strategies

The proposed telerobotic system provides the operator the ability of interacting with the environment in direct teleoperation tasks which enabled qualitative study of contact forces. This approach allows understanding some of the limiting factors in telerobotics and to develop engineering solutions.

In virtual reality based teleoperation [102, 101, 106, 107] the operator plans an operation using a model, the plan controls a slave arm, and slave arm transmits back parametric feedback. The primary issue is operation safety. Mainly off-line approaches are used and teleoperation is carried out on a static environment with no dynamic interaction reported. However, in [64] graphic animation of robot kinematics, dynamics, friction, and impact forces used in a closed loop control provides the operator the feeling of repulsive forces which allowed to carry out peg-in-hole insertion. The proposed telerobotic framework provides direct-oriented teleoperation with CAT tools augmented with some supervisory control schemes to improve teleoperation effectiveness in real interactions with the environment.

We concur with [108] on the importance of kinesthetic force feedback in assembly operations. We extended direct teleoperation by using compliance control that makes the slave arm continuously searching to nullify F/M sensed on the current tool while the whole arm is being driven by the operator to take advantage of the above mechanism in current task. In comparison to [109] our proposed VFF and VAC schemes have similar effects in modifying task trajectory. The active compliance controller continuously searches corrections in tool position and orientation that reduce tool external F/M. Operator sets task-oriented compliance and leads

the arm under compliance equilibrium to work location. Proposed VAC reduced peak contact forces and task time as compared to kinesthetic force feedback with vision in insertion and assembly tasks. VAC may also be useful as a task locality mechanism to ensure task continuity in delayed teleoperation.

As compared to [110] the proposed system also uses indexing and scale in addition to a tool-oriented dynamic motion mapping in position and force to carry out coordinated motion as a strategy to reduce operator cognitive load. This enables force reflection from current tool to the operator which increases the feeling of telepresence and enables teleoperation tasks to be completed more easily and with lower contact forces.

The wrench mapping of [111] is comparable to proposed tool motion and force mapping. However, our dynamic mapping scheme showed to be useful tool for many tasks where the point of interest is function of task state. Proposed mapping makes the operator logically mapped, in position and force, to remote object. The accomplishment of above experiments is fundamentally due to proposed dynamic mapping scheme which is estimated to be the most critical CAT tool in proposed telerobotics.

7 Conlusions and recommendations

The project is on the design and evaluation of a telerobotic system that consists of a master arm interconnected to a slave arm by means of a local area network. The system provides the operator with stereo views as well as displaying of reflected force feedback. This final report covers the whole duration of the project.

Basically the master and slave robot arms are interconnected by using a client-server computer programming system. The team designed and implemented a Multi-threaded Distributed Components Client-Server interface using object-oriented programming, Visual C#, and .NET technology (Section 6.4). The team imbedded in the client-server system all needed direct and inverse kinematic models for motion coordination (Section 6.1) and teleoperation mapping functions (Section 8). The team successfully implemented a library of intelligent telerobotic services, or Computer Aided Telerobotic Functions, such as the master shift, space scalability, the mapping of operator hand to a floating tool, and compliance loop at slave arm. Evaluation shows (Section 5.11) that the mapping of motion from the operator hand to the slave arm tool is excellently serving its purpose. The operator feels he is directly acting (moving) on the manipulated tool which being held by the remote slave arm. Since this is a natural capability of human arm, the operator needs not to see his arm or the master arm during teleoperation tasks but only needs to concentrate his view and haptic feeling on the remotely manipulated object.

For the stereo visualization the team designed and implemented a distributed framework (client-server) for relaying stereo vision over the network and successfully interfaced it to a head-mounted display at the client site (Section 6.5). Using HMD stereo visualization system, the operator excellently perceives the scene depth information. He can estimate the distance between manipulated objects and perceives the geometric positioning of parts with respect to each other during part-mating operations. The stereo visualization provides excellent depth perception. The remote camera zooming function and the motion scalability function present excellent operating tools for doing tasks in very small scale like a few mms. The team evaluated the performance of the above master-slave and stereo vision client-server systems, identified its performance metrics, and showed that its functional description and overall refreshing rates (transfer) of stereo vision, force information, and motion commands are adequate for telerobotics.

The team designed and manufactured at KFUPM a 6 DOF light Master Arm. The master arm structure uncouple motion translation, at hand, from change in orientation. The iso-impedance feature of master arm and its concurrent rotation axes of last three dof makes it transparent to the operator. The team successfully operated the proposed telerobotic system under real-time multi-streaming of stereo information, motion commands, and force feedback. Using the HMD, reflected force feedback, and the master arm the team successfully carried out a number of tasks like pouring water, insertion, assembly of a small water pump, and wire-wrapping an electronic circuit.

The team evaluated overall real-time delays and jitter involved in end-to-end multistreaming of video data, force information, and motion control over a LAN. To minimize delays the team engineered the telerobotic client-server, motion coordination system, and stereo vision server using concurrent programming. The optimized

system has refreshing rates of 50 Hz in real-time transfer of commands, 76 Hz for reflected force feedback, and 17 fps for stereo vision over a 100 Mbps LAN. The proposed telerobotic system is a useful tool to perform remote working tasks in hazardous and hostile environments.

The electro-mechanical structure (Section 5.11) of the proposed teleoperation system consists of (1) the slave arm, (2) the master arm represented by its motors and its position sensors, and (3) the human operator. During telerobotic tasks involving contact like the insertion of a peg into a hole the display of reflected force feedback causes a pre-matured master arm reaction, due to transmission elasticity, before the force is being sensed by the operator. Therefore it is recommended to: (1) use of less elastic wire in the transmission system to increase transmission stiffness, (2) accept a reduction of master arm work-space and adopt a smaller size master arm to reduce wire length, and (3) minimize overall master arm inertia and friction by adopting a very light structure.

The team designed and manufactured a generic wrist force sensor (Section 6.3) that is used to detect approximative force information which is exerted on the manipulated object. At the slave server, the detected forces and moments can be selectively activated to orchestrate a force regulation over a given sub-set of DOFs. At the master arm the transmitted forces and moments are converted into motor torques and displayed on the master arm. Since the local loop is much closer to the slave arm it immediately activates the force regulation until the external force is removed. The operator reaction to displayed forces is slower due to network and protocols which make the remote force control useful for coarse corrections.

The processing and network delays were addressed as follows:

1. The team extensively used thread engineering and software optimization (see Section 6.4 and 6.5) to produce an effective multi-streaming which will be summarized as follows. The team used a 100 Mbps LAN, 2.0 GHZ P-IV client and server computers, 1 GB DRAM, and accelerated graphics card with 256 MB DDR memory. Each force data packet contains 6 double values which equal $6 \times 8 = 48$ bytes. For force and video multistreaming the refreshing rate of the inter-arrival times of force packets is 250 Hz. The mean value of the inter-arrival times of stereo video frames is 87.57 ms with a 90% confidence interval falling between 72 and 107 ms. For force, command, and video multi-streaming when all of the three force, command and video threads are invoked simultaneously, for the force packets the mean inter-arrival rate is 1.1 ms while 100% of the population remains under 8 ms. The video transfer rate for an image to move from camera to DRAM is 24 ms using DirectShow. The stereo video client-server transfers two images (stereo frame) of size 288×360 pixels at a rate of 17-18 fps with a delay of around 58 ms only. In summary multi-streaming is running with a sampling rate of 17 Hz for stereo video, 76 Hz for force feedback, and 50 Hz for operator commands over a commodity 100 Mbps LAN. Thanks to multithreading for the graceful degradation of real-time force feedback data in periods of intensive video streaming.

2. The team implemented an augmented reality system (Section 6.6) to reduce network and processing delays in Telerobotics. The operator carries out task

planning by creating a virtual ball and sending refined group of motion specifications in one transfer to the slave arm which greatly reduces network delays. Augmented reality contributes in reducing delays by sending only planned motion tasks that have been refined and validated by the operator which reduces the need for intensive interaction with the remote site.

3. To reduce processing delays and overhead, the PUMA robot was run under a fine trajectory control (see [70] and Section 6.7) so that only coarse master arm positions need to be send while local slave control is continuously active. The interface allows: (1) downloading programs to the PUMA to provide local trajectory control at initialization only, and (2) uploading state information online to control the PUMA from the server station. It was found and that using the above approach, the dominant delays in the LAN environment are only due to robot motion. This approach provides the best possible optimization to reduce the mechanical delays which are due to the PUMA slave system.

4. The team also minimized the network delays by operating the client and server kinematic functions and communication functions in concurrent way, i.e. asynchronous or independent communication. The kinematic motion computation time is overlapped with the network communication delays (Section 6.4). The important issue to note here is that all computations are done, in the server or the client, in an asynchronous or independent way to overlap their times with the motion or communication times which result from the motion synchronization of the PUMA with the master arm. In summary the network delays are minimized by carrying out all computations in an asynchronous way in both client and server but keep motion synchronization by using fast messaging system (busy-wait or ready) between the PUMA motion and the client commands.

The team tested the system by using a set of telerobotic tasks (section 5.11) which are: (1) pouring of water, (2) per-in-hole insertion with low tolerance, (3) assembly of a small water-pump, (4) operating drawers, and (5) carrying out wire-wrapping operations on an electronic circuit breadboard. The above tasks were successfully carried out. Video clips on the above experiments are available at [1]. The mapping of operator hand motion to slave arm tool proved to be very effective in allowing the operator to set up the manipulated object in the desired position and orientation with minimal number of trials as evidenced in the attached video clips describing the above tasks. The local (server) and remote (operator) cooperative force control proved to be convergent in carrying out tasks that require fine force control like the insertion. The motion scalability and camera zooming allowed us to operate in a very small scale requiring high positioning accuracy like in the wire-wrapping tasks. It is clear that the proposed system can scale down the overall telerobotic operations and keep a high quality eye-hand motion coordination capability through the network regardless of the distance constraints. This capability is precious for robotic surgery. Therefore, it is highly recommended to use the proposed system as the basis to develop an advanced telerobotic system for robotic surgery by developing further the master arm, the computer aided teleoperation and assistance functions, the stereo vision and its augmented reality component, and overall force feedback

display and interaction. The proposed telerobotic system is an excellent tool for extending human's manipulative capabilities for remote work through a network. The proposed telerobotic system contributes in allowing human operators to perform working tasks in hazardous, hostile, unaccessible, and small and extremely small environments.

It is recommended to design a Universal Master Arm Station that would be an excellent man-machine interface equipped with a highly sophisticated computer-aided teleoperation tools. It also recommended to design specialized slave arms, each arm with a given structure is for a given class of tasks and environment. A Universal master arm station may interface humans to a quite different working environment allowing ease of operations and reciprocal exchange of effects from different physical laws to advance human knowledge at all scales.

The publication record out of this research project is as follows:

1. Mayez Al-Mouhamed, Onur Toker, and Abdul-Khalik Al-Harthy, "A 3D Vision-Based Man-Machine Interface For Hand-Controlled Telerobot", IFAC Inter. Conf on Intelligent Control Systems and Signal Processing, C-sponsored by IEEE, IFSA, and EVONET, Faro, Portugal, April 8-11, 2003, pp. 586-591.

2. Mayez Al-Mouhamed, Onur Toker, and Abdul-Khalik Al-Harthy, "A 3D Vision-Based Man-Machine Interface For Hand-Controlled Telerobot", Accepted in the IEEE Trans. on Industrial Electronics.

3. Mayez Al-Mouhamed, Onur Toker, and Asif Iqbal, Design of a Multi-Threaded Distributed Telerobotic Framework, The 10th IEEE Inter. Conf. On Electronics, Circuits and Systems (2003), pp. 1280-1283.

4. Mayez Al-Mouhamed, Onur Toker, and Asif Iqbal, Performance Evaluation of a Multi-Threaded Distributed Telerobotic Framework, The 10th IEEE Inter. Conf. On Electronics, Circuits and Systems, Dec. 2003, pp. 1284-1287.

5. Mayez Al-Mouhamed, Onur Toker, and Asif Iqbal, Design of a Multi-Threaded Distributed Telerobotic Framework for Telerobotics, Under review with the IEEE Trans. on Mechatronics, April, 2004.

6. Mayez Al-Mouhamed, Onur Toker, Mohammed Nazeeruddin, and Asif Iqbal, "A Distributed Framework for Relaying Stereo Vision for Telerobotics", Accepted in the IEEE Inter. Conf. on Pervasive Services, ICPS'2004, July, 2004, Beirut, Lebanon.

7. Mayez Al-Mouhamed and Nesar Merah, "Decoupled structure for the design of telerobotic master arms", Under preparation.

References

[1] Telerobotics video clips. *http://www.ccse.kfupm.edu.sa/researchgroups/robotics1/, 2003.*

[2] R.D. Howe and Y. Y. Matsuoka. Robotics for surgery. *Annual Review of Biomedical Engineering*, pages 211–240, 1999.

[3] H. Fuchs et al. Virtual space teleconferencing using a sea of cameras. *Proc. First Intl Symp. on Medical Robotics and Computer Assisted Surgery*, pages 161–167, 1994.

[4] Wendy Wifler. The future is now - revolutionary new voice-activated robots used in surgery. *WWW of MedTech*, October 1999.

[5] F. Arai, M. Tanimoto, T. Fikuda, K. Shimojima, H. Matsuura, and M. Negoro. Distributed virtual environment for intravascular tele-surgery using multimedia telecommunication. *Proc. of the IEEE Annual Inter. Symp. on Virtual Realty*, pages 79–85, 1996.

[6] S.C. Jacobsen, F.M. Smith, E.K. Iversen, and D.K. Backman. High performance, high dexterity, force reflective teleoperator. *Proc. 38th Conf. Remote Systems Technology*, pages 180–185, 1990.

[7] L. Stocco and S.E. Salcudean. A coarse-fine approach to force-reflecting hand controller design. *Proc. of the Inter. Conf. on Robotics and Automation*, 1:404–410, 1996.

[8] R.D. Howe. A force-reflecting teleoperated hand system for the study of tactile sensing in precision maniplulation. *IEEE Inter. Conferance on Robotics and Automation*, pages 1321–1326, May 1992.

[9] H. Iwata. Artificial reality with force-feedback: Development of desktop virtual space with compact master manipulator. *ACM SIGGRAPH Computer Graphics*, 24 (4), 1990.

[10] S.E. Salcudean, K. Hastrudi-Zaad, S.P. Tafazoli, S. DiMaio, and C. Reboulet. Bilateral matched impedance teleoperation with application to excavator control. *Proceedings of the IEEE International Conference on Robotics and Automation, Leuven, Belgium*, pages 133 – 139, May 1998.

[11] J.P. Desai and R.D. Howe. Towards the development of a humanoid arm by minimizing interaction forces through minimum impedance control. *Proc. IEEE International conference on Robotics and Automation,ICRA*, 4(2):4214 – 4219, 2001.

[12] S.E. Salcudean, N.M. Wong, and R.L. Hollis. Design and control of a force-reflecting teleoperation system with magnetically levitated master and wrist. *IEEE Transactions on Robotics and Automation*, 11(6):844–858, December 1995.

[13] S.E. Salcudean, N.M. Wong, and R.L. Hollis. A force-reflecting teleoperation system with magnetically levitated master and wrist. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1420–1426, 1992.

[14] F. Lai and R.D. Howe. Evaluating control modes for constrained robotic surgery. *IEEE Inter. Conf on Robotics and Automation*, 1:603–609, 2000.

[15] C.R. Wagner, N. Stylopoulos, and R.D. Howe. The role of force feedback in surgery: analysis of blunt dissection. *Inter. Conf. on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 68–74, 2002.

[16] L. Sooyong, D.-S. Choi, M. Kim, C.-W. Lee, and J.-B. Song. A unified approach to teleoperation: human and robot interaction. *IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems*, 1:261–266, 1998.

[17] L. Sooyong, P. Sangmin, M. Kim, and C.-W. Lee. Design of a force reflecting master arm and master hand using pneumatic actuators. *Proc. of the IEEE Inter. Conf. on Robotics and Automation*, 3:2574–2579, 1998.

[18] J.S. Lee, H.M. Kwon, D.Y. Shin, and J.B. Song. Force feedback using sensibility ergonomics theory in teleoperation system. *IEEE/RSJ Inter. Conf. on Intelligant Robots and Systems*, pages 597–602, 1999.

[19] S. Won, A. Kim, and K.B Antal. Demonstration of a high-fidelity predictive/preview display technique for telerobotics servicing in space. *IEEE TRansactions on Robotics and Automation*, 9:698–702, October 1993.

[20] Hayward V. Survey of haptic interface research at McGill University. *Proc. Workshop in Interactive Multimodal Telepresence Systems*, pages 91–98, March 2001.

[21] Y. Y. Yokokohji, R. L. Hollis, and T. Kanade. What you can see is what you can feel. *IEEE Virtual Reality Annual International Symposium*, pages 46–53, 1996.

[22] T.M. Massie and J.K. Salisbury. The PHANToM haptic interface: a device for probing irtual objects. *3rd Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems*, DSC-Vol 1:295–301, 1994.

[23] H. Noma, Y. Kitamura, T. Miyasato, and F. Kishino. Haptic and visual feedback for manipulation aid in a virtual environment. *ASME-DSC*, -Vol.58:469–476, 1996.

[24] S. Grange, F. Conti, P. Helmer, P. Rouiller, and C. Baur. Overview of the delta haptic device. *Eurohaptics '01*, July 2001.

[25] M. Brown and J. Asquith. A man-machine interface with feelings. *AEA Technology, (www.aeat.co.uk)*, 1998.

[26] A. K. Bejczy and J. K. Salisbury. Kinesthetic coupling between operator and remote operator and remote manipulator. *Proc. ASME International Computer Technology Conference , San Francisco*, page 197, 1980.

[27] Y. Tsumaki, H. Naruse, D.N. Nenchev, and M. Uchiyama. Design of a compact 6-dof haptic interface. *IEEE International Conference on Robotics an Automation*, pages 2580–2585, 1998.

[28] S. Chang, J. Kim, I. Kim, J.H. Borm, and C. Lee. KIST teleoperation system for humanoid robot. *IEEE/RSJ Inter. Conf. on Intelligant Robots and Systems*, pages 1189–1203, 1999.

[29] S. Lee, D.S. Choi, M. Kim, C.W. Lee, and J.B. Song. A unified approach to teleoperation: Human and robot integration. *International Conference on Intelligent Robots and Systems*, pages 261–266, 1990.

[30] S. Lee, S. Park, M. Kim, and C. W. Lee. Design of a force reflecting master arm and master hand using pneumatic actuators. *IEEE International Conference on Robotics and Automation*, pages 2574–2579, 1998.

[31] W.K. Yoon, Y. Tsumaki, and M. Uchiyama. An experimental teleoperation system for dual-arm space robotics. *Journal Robotics and Mechatronics*, 12, No 4:378–384, 2000.

[32] M. Takahashi, Y. Tsumaki, D.N. Nenchev, and M. Uchiyama. A teleoperation system with collision avoidance capability based on virtual radar. *International Workshop on Robot and Human Communication*, pages 154–159, 1997.

[33] W.K. Yoon, Y. Tsumaki, and M. Uchiyama. An experimental system for dual-arm robot teleoperation in space with concepts of virtual grip and ball. *Proc. 9th International Conference on Advanced Robotics*, 12, No 4:225–230, 1999.

[34] H. Kazerooni. Direct drive active compliant end effector. *IEEE Journal of Robotics and Automation*, 4:324–333, Mar 1988.

[35] Sherry Draisey. "a multi-axes static/dynamic force sensor for ngst. *Science and Technology Exposition, September 13-16 1999, Hyannis, MA*, 1999.

[36] William A. Lorenz. Force sensors for human/robot interaction. *MSc. Thesis, Mecahanical Engineering Department, Northwestern University*, 1999.

[37] www.demensional.com/manuals/eye3d 3-in-1. *User's Manula (E).pdf, 2003.*

[38] S. Lee, S. Lakshmanan, S. Ro, J. Park, and C. Lee. "optimal 3D viewing with adaptive stereo displays for advanced telemanipulation. *International Conference on Intelligent Robots and Systems*, pages 1007–1014, 1996.

[39] S. Lee, S. Ro, J. Park, and C. Lee. "optimal 3D viewing with adaptive stereo displays: A case of tilted camera configuration. *ICAR '97*, pages 839–844, 1997.

[40] D. B. Diner and D. H. Fender. Human engineering in stereoscopic viewing devices. *Jet Propulsion Laboratory report (JPL D-8186)*, 1991.

[41] R. L. Pepper, R.E. Cole, and E. H. Spain. The influence of camera separation and head movement on perceptual performance under direct and tv displayed conditions. *Proceedings of the Society for Information Display*, pages 73–80, 1996.

[42] http://www.stereo3d.com/3dhome.htm, 2003.

[43] L. M. Strunk and T. Iwamoto. A linearly-mapping stereoscopic visual interface for teleoperation. *IEEE International Workshop on Intelligent Robots and Systems, IROS' 90*, pages 429–436, 1990.

[44] http://www.3dgw.com/articles/articlepage.php3, 2003.

[45] G. Lee, S. Bekey and A. Bejczy. Computer control of space teleoperators with sensory feedback. *Proc IEEE Int. Conf. on Robotics and Automation*, pages 205–214, 1985.

[46] A. Bejczy. Space shuttle remote manipulator system force-torque system. *http://ranier.oact.hq.nasa.gov/telerobotics_page*, 2003.

[47] http://telerobot.mech.uwa.edu, 2002.

[48] T. Suzuki, T. Fujii, and K. Yokota. Teleoperation of multiple robots through the Internet. *IEEE International Workshop on Robot and Human Communication*, pages 84–89, 1996.

[49] The Pittsburgh robot. http://csc.clpgh.org/telerobt, 2002.

[50] Puma Paint. http://yugo.mme.wilkes.edu/ villanov, 2002.

[51] Raiders telerobot. http://www.usc.edu/dept/raiders, 2002.

[52] Robotic Garden. http://www.usc.edu/dept/garden, 2002.

[53] A mobile robot in a maze. http://khepontheweb.epfl.ch, 2002.

[54] Robot plays Towers of Hanoi with live video. http://www.fh-konstanz.de/studium/ze/cim/projekte/webcam, 2002.

[55] H. Friz, P. Elzer, B. Dalton, and K. Taylor. Augmented reality in Internet telerobotics using multiple monoscopic views. *IEEE Computer*, pages 354–359, 1998.

[56] F. F. Dong and M. Meng. A computer 3D animated graphical interface for control and teleoperation of robot system. *IEEE Computer*, pages 778–783, 1997.

[57] M. Jägersand. Image based predictive display for tele-manipulation. pages 550–556, 1999.

[58] F. Paolucci and M. Andrenucci. Teleoperation using computer networks: Prototype realization and performance analysis. *Proc. of the 8th Mediterranean Electrotechnical Conference, MELECON'96*, 2:1156–1159, 1996.

[59] Y.E. Ho, H. Masuda, H. Oda, and L.W. Stark. Distributed control for teleoperations. *Proc. of the 1999 IEEE/ASME International Conf. on Adv. Intelligent Mechatronics*, pages 323–325, September 1999.

[60] Y.E. Ho, H. Masuda, H. Oda, and L.W. Stark. Distributed control for teleoperations. *IEEE/ASME Transactions On Mechatronics*, 5(2):100–109, June 2000.

[61] Chong; Ohba; Kotoku and Komoriya. Coordinated rate control of multiple telerobot systems with time delay. *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages V1123–V1128, October 1999.

[62] Martin Jagersand. Image based predictive display for tele-manipulation. *Proc. of 1999 IEEE International Conf. on Robotics and Automation*, pages 550–556, 1999.

[63] Kazuhiro Kosuge, Hideyuki Murayama, and Koji Takeo. Bilateral feedback control of telemanipulators via computer network. *Proc. of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3:1380–1385, Nov. 1996.

[64] N. Funabiki, K. Morishige, and H. Noborio. Sensor-based motion-planning of a manipulator to overcome large transmission delays in teleoperation. *Proc. of IEEE International Conference on Systems, Man, and Cybernetics*, 5:1117–1122, 1999.

[65] F. Goktas, J.M. Smith, and R. Bajcsy. Telerobotics over communication networks. *Proc. of IEEE Conference on Decision and Control*, 3:2399–2404, 1997.

[66] K. Kosuge and H. Murayama. Bilateral feedback control of telemanipulator via computer network in discrete time domain. *IEEE Inter. Conf. on Robotics and Automation*, 3:2219–2224, 1997.

[67] Xi Ning and T.J. Tarn. Action synchronization and control of Internet based telerobotic systems. *Proc. of IEEE Inter. Conf. on Robotics and Automation*, 1:219–224, 1999.

[68] P. Milgram, A. Rastogi, and J. Grodski. Telerobotic control using augmented reality. *IEEE Inter. Workshop on Robot and Human Communication*, pages 21–29, 1995.

[69] W.J. Book, H. Lane, L.J. Love, D.P. Magee, and K. Obergfell. A novel teleoperated long-reach manipulator testbed and its remote capabilities via the Internet. *Proc. of the IEEE Inter. Conf. on Robotics and Automation*, 3(6):1036–1041, 1997.

[70] A. Al-Harthy. Design of a telerobotic system over a local area network. *M.Sc. Thesis, King Fahd University of Petroeum and Minerals*, January 2002.

[71] A. Bejczy. Grasping force sensor for robot hand. *http://ranier.oact.hq.nasa.gov/telerobotics_page*, 2002.

[72] F. Aghili, M. Buehler, and J.M. Hollerbach. A joint torque sensor for robots. *Proc. ASME Int. Mech. Eng. Conference*, pages 66–69, 1998.

[73] www.jr3.com/man/, 2002.

[74] www.ati.com, 2002.

[75] www.omega.com/pressure, 2002.

[76] J. W. Dally and W. F. Riley. Experimental stress analysis. *McGraw Hill*, 1978.

[77] W. Al-Wadiee. Design of a force sensor. *Senior Design Project, ME-KFUPM, Dhahran, KSA*, 2002.

[78] ANSYS. Swanson analysis system, inc. *Version 5.5, 2003.*

[79] A. Rovetta, R. Sala, Xia Wen, and A. Togno. Remote control in telerobotic surgery. *IEEE Trans. on Sys., Man, and Cybernetics, Part A*, 26(4):438–444, 1996.

[80] D.W. Marhefka and D.E. Orin. A compliant contact model with nonlinear damping for simulation of robotic systems. *IEEE Trans. on Sys., Man, and Cybernetics, Part A*, 29(6):566–572, 1999.

[81] Teresa Ho. System architecture for Internet-based teleoperation systems using java. Master's thesis, Department of Computing Science, University of Alberta, Canada, 1999.

[82] Huosheng Hu, Lixiang Yu, Pui Wo Tsui, and Quan Zhou. Internet-based robotic systems for teleoperation. *International Journal of Assembly Automation*, 21(2):143–151, May 2001.

[83] M. Al-Mouhamed, O. Toker, and A. Al-harthy. A 3D vision-based man-machine interface for telerobotics. *IFAC Inter. Conference on Intelligent Control and Signal Processing*, April 8-11 Portugal, 2003.

[84] Microsoft. http://msdn.microsoft.com/default.asp, 2002.

[85] N. Hollinghurst and R. Cipolla. Human-robot interface by pointing with uncalibrated stereo vision. *Image and Vision Computing*, 14(3):171–178, 1996.

[86] F. Wolfgang and A. Geva. *Beginning Direct3D Game Programming.* Prima Tech, ed. Andre LaMother, 2001.

[87] T. Kazerooni, H. Tsay and K. Hollerback. A controller design framework for telerobotic systems. *IEEE Trans. on Contr. Syst. Technol.*, 1, Mar 1993.

[88] Ajit Shah. Developmnet of a telepresence surgical system. *WWW site of Defense Sciences Office*, October 1999.

[89] M. Fukumoto, K. Mase, and Y. Suenaga. Real-time detection of pointing actions for a glove-free interface. *IAPR Workshop on Machine Vision Applications*, pages 473–476, 1992.

[90] J. E. Lloyd, J. S. Beis, K. D. Pai, and D. J. Lowe. Model-based telerobotics with vision. *Proc. IEEE ICRA*, pages 1297–1304, 1997.

[91] N. J. Hollinghurst and R. Cipolla. Uncalibrated stereo hand-eye coordination. *Image and Vision Computing*, 12:187–192, 1994.

[92] R. Cipolla, D.P. Robertson, and N. J. Hollinghurst. Human-robot interface by pointing with uncalibrated stereo vision. *Proc. IAPR workshop on machine vision applications MVA'94*, pages 171–178, 1994.

[93] S. Sato and S. Sakane. A real-time tracking system that visually tracks the operator's pointing hand and projects a mark in the real work space. *Proc. IEEE Intr. Conf. and Robotics and Automation*, 2000.

[94] Y. Kuno, M. Sakamoto, K. Sakata, and Y. Shirai. Vision-based human interface with user-centered frame. *Proceedings of the IROS 94*, 3:2023–2029, 1994.

[95] Y. Kuno, K. Hayashi, K.H. Jo, and Y. Shirai. Human-robot interface using uncalibrated stereo vision. *International Conference on Intelligent Robots and Systems 95*, 1:525–530, 1995.

[96] H. Masuda; H. Oda; Y. F. H.; Stark. Distributed control for tele-operations. *IEEE Trans. on Mechatronics*, 5(2):100–109, June 2002.

[97] M. Al-Mouhamed. A robust gross-to-fine pattern recognition system. *IEEE Trans. on Industrial Electronics*, 48(6):1226–1237, Dec. 2001.

[98] J.E. Shigley and C.R. Mischke. Mechanical engineering design. 2000.

[99] J.E. Shigley and C.R. Mischke. Standard handbook of machine design. 1986.

[100] FAG. Kugelfischer georg schafer kgaa "fag standard programme". *Catalogue 41510 EA, 2001.*

[101] E. Even, E. Fournier, and R. Gelin. Using structural knowledge for interactive 3D modeling of piping environments. *Proc. of IEEE Inter. Conf. on Robotics and Automation*, pages 2013–2018, Apr. 2000.

[102] E. Even, P. Gravez, E. Maillard, and E. Fournier. Acquisition and exploitation of a 3D environment model for computer aided tatleloperation. *Proc. of the 1999 IEEE Inter. Workshop on Robot and Human Integration*, pages 261–266, Sep. 1999.

[103] L.F. Penin, K. Matsumoto, and S. Wakabayashi. Force reflection for time-delayed teleoperation of space robot. *Proc. of IEEE Inter. Conf. on Robotics and Automation*, pages 3120–2125, Apr. 2000.

[104] KFUPM. Robotics lab. *http://www.ccse.kfupm.edu.sa/researchgroups/robotics, 2003.*

[105] J. Singer. JVM versus CLR: A comparative study. *2nd International Conference on the Principles and Practice of Programming in Java*, 2003.

[106] Bailin Cao, G.I. Dodds, and G.W. Irwin. An event driven virtual reality system for planning and control of multiple robots. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 1999. IROS '99*, 2:1161–1166, Oct. 1999.

[107] A. Codourey, M. Rodriguez, and I. Pappas. A task-oriented teleoperation system for assembly in the microworld. *Proc. 8th International Conference on Advanced Robotics, 1997. ICAR '97*, pages 235–240, July 1997.

[108] T. Imaida, Y. Yokokohji, T. Doi, M. Oda, and T. Yoshikwa. Ground-space bilateral teleoperation experiment using ETS-VII robot arm with direct kinesthetic coupling. *Proc. of the IEEE Int. Conf. on Robotics and Automation,ICRA 2001*, 1:1031 – 1038, June 2001.

[109] Sukhan Lee, Ming-Feng Jean, Jong-Oh Park, and Chong-Won Lee. Reference adaptive impedance control: a new paradigm for event-based robotic and telerobotic control. *IEEE/RSJ International Conference on Intelligent Robots and Systems, 1998*, 2:1302 – 1307, Oct. 1998.

[110] Tan Fung Chan and R.V. Dubey. Design and experimental studies of a generalized bilateral controller for a teleoperator system with a six dof master and a seven dof slave. *Proc. IEEE/RSJ/GI International Conference on Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94*, 1:415 – 422, Sept. 1994.

[111] L.E.P. Williams, R.B. Loftin, H.A. Aldridge, E.L. Leiss, and W.J. Bluethmann. Kinesthetic and visual force display for telerobotics. *IEEE Inter. Conf. on Robotics and Automation, ICRA '02*, 2:1249–1254, 2002.

8 APPENDIX A: Description of multi-threaded distributed telerobotic framework

In this appendix we provide a description that combines software specifications with mathematical functionality of each software component to better explain the software architecture. The software used significantly deviates from traditional input-output procedure-oriented programming to a newer programming style called object-oriented, distributed components paradigm. The mathematical functionality of each component will help the reader understand the service provided at each component activation.

8.1 Server side components

The server components are: (1) PUMA Component, (2) Force Sensor Component, and (3) Decision Server Component. In addition to these components, we also have three interfaces known as (1) Proxy Robot Interface (2) Force Sensor Interface, and (3) Decision Server Interface. In the following sub-sections, the software component aspects are presented after describing the functionalities of each component and associated interfaces.

8.1.1 PUMA component

The kinematics of slave arm is represented by means of three frames: (1) a fixed world frame (R_w) at arm origin, (2) an effector frame (R_e) located at arm terminus, and (3) a user defined tool frame (R_t). The controllable frame R_e is represented by its 3×1 position vector ($E_w(\theta)$) and its (3×3) orientation matrix ($M_w^e(\theta)$), where θ is the slave arm joint vector and w refers to R_w. The tool frame R_t is user or system defined by its position vector T_t and orientation matrix M_e^t of tool frame R_t with respect to frame R_e. The position of the tool point is defined by:

$$T_w = E_w + M_w^e(\theta) M_e^t T_t \tag{324}$$

The slave station receives a command to translate the tool frame R_t by ΔT_w and to rotate it by ΔM_t. The operator motion can be efficiently mapped onto the tool frame when the translation is specified in tool frame, i.e. ΔT_t. The new arm controllable position vector is:

$$\Delta E_w = M_w^t (I - \Delta M_t) T_t + \begin{cases} \Delta T_w & \text{Operator-tool} \\ M_w^t \Delta T_t & \text{Operator-world} \end{cases} \tag{325}$$

where $M_w^t = M_w^e M_e^t$. The new effector orientation matrix (controllable)is:

$$\Delta M_e = M_e^t \Delta M_t M_t^e \tag{326}$$

The PUMA component reads current robot joint θ as a 6×1 vector which allows computing current effector position $E_w(\theta)$ and orientation $M_w^e(\theta)$. The target effector position and orientation are $E_w^+ = E_w(\theta) + \Delta E_w$ and $M_w^{e+} = M_w^e(\theta) \Delta M_e$. The inverse kinematic model $\theta^+ = G^{-1}(E_w^+, M_w^{e+})$ of the slave arm allows finding the

joint vector θ^+ that moves the tool by the commanded translation ΔT and rotation ΔM. The new joint vector θ^+ is sent to slave arm motion controller.

The PUMA component is a software framework that implements the above tool motion coordination. It acts as a software proxy of the robot for which commands are issued to the component as they are issued to the robot. Whenever robot changes its states, the component updates itself automatically to reflet these changes.

Some important *public methods* exposed by PUMA component include *ConnectRobot* that connects server to slave robot, *InitializeRobot* sends a program to robot that repeatedly moves the robot in an incremental fashion which reduces the communication data payload. Two overloaded methods allow us to incrementally move robot using either joint (angular position) or cartesian space parameters.

The PUMA component reads current robot joint and compute the cartesian coordinates of the hand using the direct kinematic model of slave arm. A command for an incremental motion is specified by a translation vector and an orientation matrix. PUMA computes the new robot hand position and orientation and finds the new robot joint using the inverse kinematic model which is assigned to the slave arm.

The PUMA component accepts a user defined tool frame of reference as (1) robot base frame (world), (2) robot wrist frame, or (3) robot tool frame. Robot statuses are (1) connection to Robot is not detected or Robot not initialized, (2) Robot is connected but not initialized, (3) initialization is pending, (4) robot is ready to receive a motion parameter, (5) robot is moving, etc. The PUMA component also makes available some public properties to set or retrieve the attributes of the robot in realtime. The major properties are: (1) reading robot Angles, (2) computing the position vector and orientation Matrix of robot hand frame, (3) setting up specification of robot tool frame, (4) setting up the communication between server and robot, etc. The events invoked by PUMA component include: (1) Data received from PUMA, (2) some error occurred with PUMA, (3) Robot moved to a new location, and (4) PUMA status changed.

8.1.2 Force sensor component

A 6 dof force sensor is implemented at the wrist of the slave arm to provide (1) measurement of external forces, and (2) passive compliance of the tool. The sensor consists of two parallel plates p_1 (frame R_e) and p_2 (frame R_s) interconnected by three elastic links. The elementary motion of p_2 with respect to p_1 is measured by a differential (1) translation vector ΔS_e of the origin of R_s, and (2) orientation matrix ΔM_e of R_s measured in R_e. The sensor structure allows finding ΔS_e and ΔM_e as functions of the six sensing signals. The sensor frame R_s is located between the effector frame R_e and the tool frame R_t. An external force applied to the tool causes a deflection vector $\Delta T_e = \Delta S_e + (\Delta M_e - I)M_s^t T_t$ to the tool frame origin as well as a change ΔM_t in R_t orientation as $\Delta M_t = M_t^s \Delta M M_s^t$. Since $M_e^t = \Delta M M_s^t$ we can compute the tool deflection vector in its frame R_t:

$$\Delta T_t = M_t^s \Delta M^{-1} \Delta T_e \tag{327}$$

The force (F_t) and moment (C_t) vectors applied to the tool are computed using the tool linear and rotational compliance vectors ΔT_t and $M_t^s \Delta M M_s^t$. Using the

passive compliance matrices for linear (K_l) and rotational (K_r) motion of the tool we compute the force $F_t = (f_x, f_y, f_z)^t = K_l \Delta T_t$ and moment $C_t = (c_x, c_y, c_z)^t = K_r \Delta M_t$ vectors. The tool force and moments vectors F_t and C_t are used to: (1) display the reflected force feedback at the client station, and (2) implement active compliance mechanism at as supervisory functions at the slave arm level.

The force sensing component (FSC) reads the robot wrist force sensor and create a stream of reflected force feedback directed to the master station. FSC is implemented as a separate thread, the priority of which can be adjusted during runtime to allow for the management of CPU usage.

A new instance of FSC creates a new thread with a default *normal* priority and waits until the sensing is triggered. After the reading has started, it continues sensing the force information at a pre-specified, alterable, default frequency. It invokes an event to inform the parent application of the availability of a new force packet. The parent application can respond to this event using some event handler at the higher level of application hierarchy. The event directly transfer the force information bypassing the global memory. Similarly the component also provides *StopReading()* function to abort the force sensing thread anytime we want. This will free the CPU of the load of the force thread and all events coming from the Force Component will stop. After the force thread is stopped, the force sensing can be triggered again using *StartReading()*.

There are three public properties exposed by FSC. The *SensorThreadPriority* is used to set the thread priority that is one out of five OS levels. However, the *Highest* priority does not guarantee non-preemptive thread behavior because of the operating system constraints. The *TimerValue* is used to set a time interval between two successive readings. This is useful to set up the frequency of the force sampling using this property. The *ThresholdValue* is helpful in situations when we need the force event to be invoked only when there is noticeable change in at least one of the force sensor values. We can set the *ThresholdValue* property in accordance with the minimum change that we want to notice.

8.1.3 Decision server component

DecisionServer is a component that provides an autonomous local loop on the server to support supervisory telerobotic control. This is needed to avoid teleoperator using stop-and-wait strategy in the presence of significant network delays. This is a higher abstraction layer which is used as an agent to implement a number of assistance functions like the local robot impedance control and workspace scalability functions. This layer can also accommodate the repeatability of a set of movement commands. For this we need an interface that allows a remote client to control a distributed telerobotic system.

The assistance functions are:

1. Relative or world motion
   The incremental operator hand motion is applied at the slave arm as increments with respect to: (1) world frame in pre-positioning tasks, or (2) tool frame in tool manipulation tasks.

2. Floating Tool Mapping

The mapping of operator hand motion to slave tool frame can be set by defining the tool frame position and orientation with respect to the slave effector frame located at arm wrist. In pre-apprehension the tool frame can set at end of first 3 dof to properly orient the slave tool to object. While object manipulation is easier if tool frame matches the tool setting.

3. Planar or Linear Geometrical Constraints
   Geometrically constraining some tool motion axes to linear or planar motion and leaving the other axes under direct user manual control. This is useful in drilling like operations where it is difficult to the user to manually maintain some geometrical constraints such as tool direction.

4. Force Control
   To simplify the task complexity and reduce contact forces the user may use supervisory control to activate remote compliance mechanisms at the server station. Here the user may selectively set up force control over a sub-set of tool axes. This is useful in insertion like operations. In pre-insertion, the user uses manual control of the search plan and force control to maintain contact. In insertion, manual control is needed for the insertion direction while force control is used in the jamming plan to minimize contact forces. In this case the selected components of computed force F_t and moment C_t vectors are feedback as elementary tool translation $\Delta T = AF_t$ and rotation $\Delta M = BC_t$, where A and B are two 3×3 matrices that determine the selected axes.

The master arm station support activation and dis-activation of the above functions. For this a set of keys is integrated at the operator handle tool. The keys arrangement is such that it lead to least attention and minimum finger motion of the operator.

For the software aspects, the DecisionServer is a higher abstraction layer present on the server side. The presence of this layer allows high-level control of the robot based on force sensor data. Another advantage of this layer is the implementation of different modifiers to the commands coming from the client, for example workspace scalability function can be implemented on this level. If a learning mode is implemented in the future, DecisionServer can serve as an agent that will record the trajectories and will reproduce them by implementing an impedance control using PUMA and Force Sensor components.

The server side logic is implemented in four layers. The last layer down the hierarchy is the physical layer consisting of robot and force sensors. On the highest level of the hierarchy is the human operator that might interact with the system using a UI(user interface). A possible autonomous local loop on the server side can be constructed in the lower three layers. This can help automate the execution of simpler tasks in the presence of large time delays.

8.1.4 Server side interfaces and .NET remoting

An interface is a set carrying definitions of public methods and properties. It servers as a contract for any component that implements this interface. In other words, any component that inherits or implements the definitions contained in an interface,

must provide the implementation of all the methods or properties enumerated in the interface. This scheme is needed in .NET based distributed applications because any client that accesses or executes the methods of a component on the server needs an access to the server assembly or component. By giving a reference to an interface that the server component implements, we can hide the actual component or assembly from the client. This provides security from potential unsafe clients as well as gives the developers freedom to the easily amend the logic of the server methods while the interface remains unchanged for all the clients because an interface is only a definition, the implementation being only inside the component.

In order to attain references to both the PUMA and Force Sensor components, two interfaces are defined: *IProxyRobot* and *IForceSensor*. These interfaces carry the definitions of public methods, properties and events of PUMA and Force Sensor components as explained in sections 8.1.1 and 8.1.2. Further we define another interface *IDecisionServer* which inherits both the *IProxyRobot* and *IForceSensor* interfaces. Using this approach we are able to define a unified set of public members (methods, properties and events) that are required to be implemented in the form of DecisionServer component on the server side.

Once *IDecisionServer* is fully implemented, .NET Remoting can be used to publish an instance of DecisionServer component on the network. This instance is identified to potential clients by a unique object identifier issued by .NET Remoting. Any client can get a reference to this instance through an *IDecisionServer* interface. .NET Remoting enables accessing objects using SOAP(Simple Object Access Protocol). This scheme isolates the network protocol issues from the software development of a distributed application. Any object/component that might be located on the other end of the world can be referenced using this distributed scheme as if it was available on the same machine.

8.2 Client side components

The client contains the *IDecisionServer* interface to reference the server side component through .NET Remoting. In addition to *IDecisionServer*, there are instances of .NET Remoting and client GUI(Graphic User Interface).

8.2.1 Decision server interface

The DecisionServer is inherited from IDecisionServer and in turn from IProxyRobot and IForceSensor interfaces. .NET Remoting is responsible for making socket calls to the client and we may choose either network protocol for these requests. The client side is fairly simple and contains all the familiar components which have been explained previously.

An important feature that we need on client side is to receive the events fired by the DecisionServer instance on server side. In order to use an event handler for any event invoked by DecisionServer, we must provide the client assembly to the DecisionServer. This violates object oriented design philosophy and introduces potential security threats. To overcome this issue, we have used *shim* classes as intermediatory agents to forward DecisionServer events over to the client or *IDecisionServer* interface. Shim classes are thin assemblies visible to both the server and the client.

DecisionServer invokes the event which is received by an event handler hooked by shim classes. This event handler then calls the event handler of the client (*IDecisionServer*). By following this approach we hide the server and client assemblies from each other.

Care must be taken while receiving events from the server and writing event handlers for them because these are synchronous events which means that the thread that is invoking the event on the server side will be blocked until all the event handlers for this event are executed. So manipulating different threads in a multi-threaded application, especially the GUI thread during the invocation of the events may cause deadlocks in the distributed client-server environment.

8.2.2 MasterArm component

The operator holding of the master arm may easily exit a geometric area in which a human can efficiently move his arm while maintaining excellent motion control, i.e. arm dexterity area. In this case the teleoperation becomes very difficult. One needs to index the master arm to operator dexterity area without affecting the current position of slave arm. This defines the indexing function which must be activated by the operator hand handling the master arm in an optimized man-machine interface.

Denotes by (E, M) the previous operator position vector and orientation matrix at the master arm and p a boolean that is 0 during the indexing period, i.e. when the motion mapping between master and slave arms is disabled:

$$(\Delta E, \Delta M) = \begin{cases} (E^+ - E, M^{-1}M^+) & \text{for } p = 1 \\ (0, I) & \text{else} \end{cases} \tag{328}$$

The operator may need to scale-down his motion in the neighborhood of a critical task location to increase the motion accuracy. In this case the increment in master position vector (ΔE) and orientation matrix (ΔM) need to be scaled-down before being mapped to those of the slave arm. We evaluate three orientation angles for the operator hand frame. The operator orientation matrix M can be seen as made of three euler angles, i.e. $M = R_x(\alpha_x)\ R_y(\alpha_y)R_z(\alpha_z) = R_{xyz}(M)$, where R_u is a rotation matrix about axis u and R_{xyz} is the product of three rotation matrices sets for M. Since ΔM is known we inverse the above equation and find the three angles as $(\alpha_x, \alpha_y, \alpha_z) = R^{-1}_{xyz}(\Delta M)$. Using a user defined scale factor s, the scale function becomes:

$$(\Delta E, \Delta M) = ((E^+ - E) * s, R_{xyz}((R^{-1}_{xyz}(\Delta M)) * s))) \tag{329}$$

The *MasterArm* component implements the functionality required to carry out real-time rendering of the operator motion as well as to display haptic feedback (force) on operator hand. The *MasterArm* component, after initialization, has active instances of two force components for reading and writing in different threads.

It also implements a generic impedance control to minimize arm inertia that aims at reducing the mechanical impedance of master arm handled by the operator. The local force feedback uses a second order model for minimizing the mechanical impedance of the master arm. In order to estimate the force feedback, the component maintains a record of all the force data read for a certain number of samples (history) along with the record of the system time. Then it evaluates the velocity

and acceleration of the master arm at each sampling instant. This information is used to calculate the force proportional to what the operator is applying which is then feed back to the master arm.

In the following we shortly describe the generic impedance control implemented as part of the *MasterArm* component. Using a 6 dof master arm, the *MasterArm* component has control of a 6×1 motor torque vector τ which is in turn controls the dynamics [11] of the master arm articulated system:

$$\tau = D(q)\ddot{q} + C(q, \dot{q}) + G(q) \tag{330}$$

where q is master arm joint angular vector, $\dot{q}$ velocity vector, $\ddot{q}$ acceleration vector, $D(q)$ is the inertia matrix, $C(q, \dot{q})$ is the coriolis and centrifugal coefficients, and $G(q)$ is the gravity vector. The operator motion is characterized by vectors q, $\dot{q}$, and $\ddot{q}$. This allows computing terms $C(q, \dot{q})$ and $G(q)$ based on the geometric and dynamic of master arm. The inertia matrix $D(q)$ is nearly constant for a light master arm operating in a restricted work volume. The *MasterArm* component computes the motor torque as follows:

$$\tau = \alpha\ddot{q} + \beta\dot{q} + C(q, \dot{q}) + G(q) + \tau_{ff} \tag{331}$$

where term $\alpha\ddot{q} + \beta\dot{q}$ is generated based on the operator motion, terms $C(q, \dot{q})$ and $G(q)$ are used to compensate for dynamic effects and gravity, and τ_{ff} is the reflected force feedback torque. The overall dynamic motion equation becomes:

$$\tau_{ff} = (D(q) - \alpha)\ddot{q} + \beta\dot{q} \tag{332}$$

where term $D(q) - \alpha$ represents the reduced master arm inertia (impedance) and $\beta\dot{q}$ is a motion damping factor. The motivation for injecting term $\alpha\ddot{q} + \beta\dot{q}$ in the master arm torque is reduce overall mechanical impedance felt by the operator. The values of the parameters α and β are experimentally determined.

There are two major inputs to the *MasterArm* component, 1) angular position being read from master arm and 2) the network stream of force data coming from the remote side. *MasterArm* uses the *ReadForce* module of *Force* component to read the position data from master arm joints.

Here we describe the real-time monitoring and forwarding of operator hand position and orientation. Using the joint vector q computing the kinematic model $G_M(q)$ of the master arm allows computing the operator hand position vector X_k and orientation matrix M_k at iteration k. The component also computes the incremental hand vector $\Delta X = X_k - X_{k-1}$ and incremental orientation matrix $\Delta M_k = (M_{k-1})^t M_k$ because $Mk = M_{k-1}\Delta M_k$. The *MasterArm* sends the computed data ΔX_k and ΔM_k to the slave arm as an incremental motion command for the current slave tool position and orientation. *MasterArm* also injects the force feedback term in the output of the master arm torque vector which is sufficient to "display" to operator the received force feedback.

MasterArm is a multi-threaded component that can read and write data simultaneously as well as process lengthy operation in worker threads. The *MasterArm* component also invokes events when 1) a fresh copy of position data (incremental cartesian position data) is available from ReadForce and 2) when some force data is written to the master arm.

Some of the public methods revealed by *MasterArm* consists of Boolean used to: (1) start and stop reading the master arm position (Inherited from *Force* component), and (2) writes the given force data to the master arm in a separate thread. Now we describe some of the the public properties. One property computes the change in position vector and orientation matrix after the position data ready event is fired, A boolean property is used to find/set whether a master arm is engaged or not. If this property is false, the direct geometric model will not be evaluated to save thread time. A get/set boolean property is used to indicate whether to provide force feedback to the master arm or not. This feedback is the force stream coming from remote side. Other properties are also used to compute the local impedance control function for the master arm.

9 APPENDIX B: Assembly drawing of the master arm

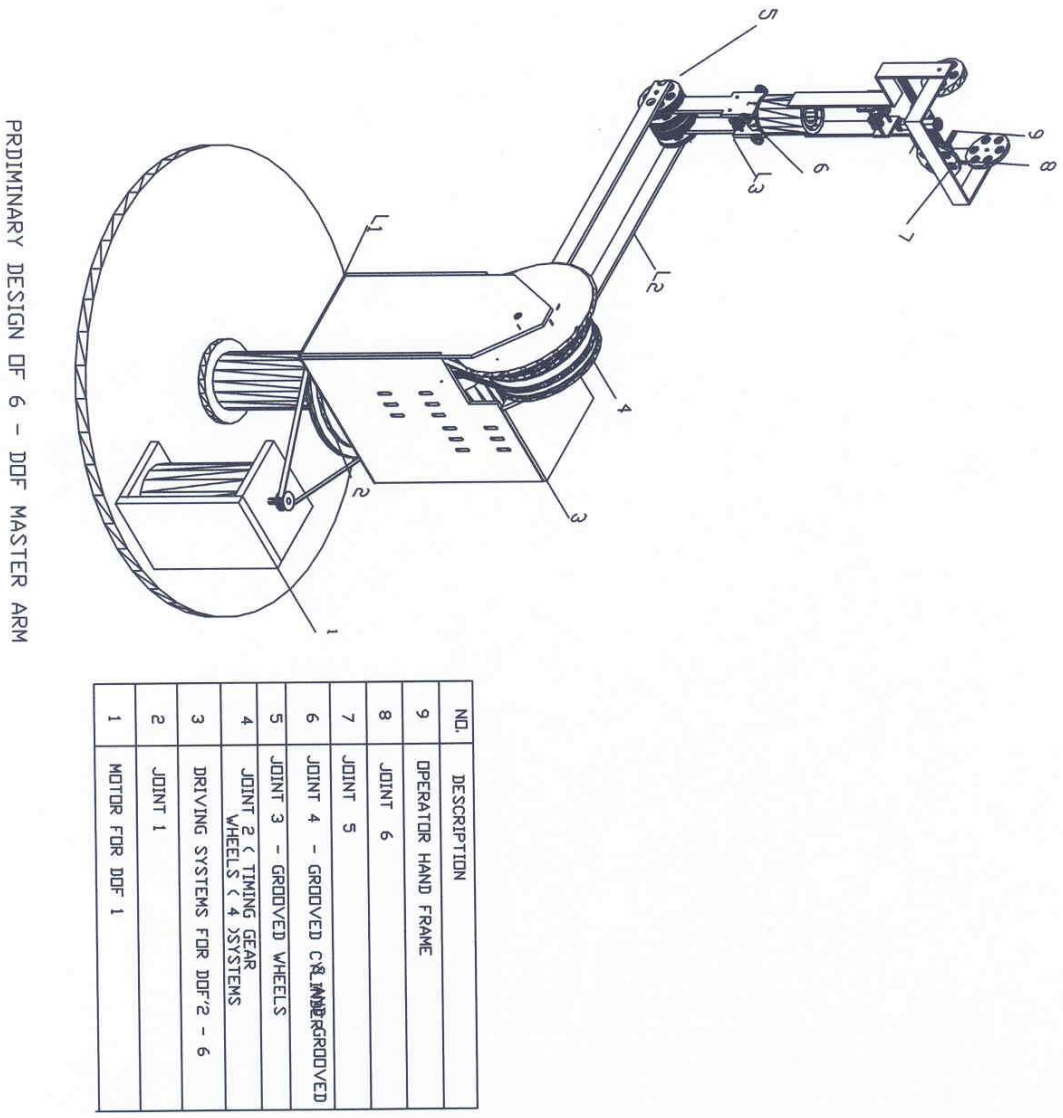


Figure 75: Preliminary design of the 6 DOF master arm

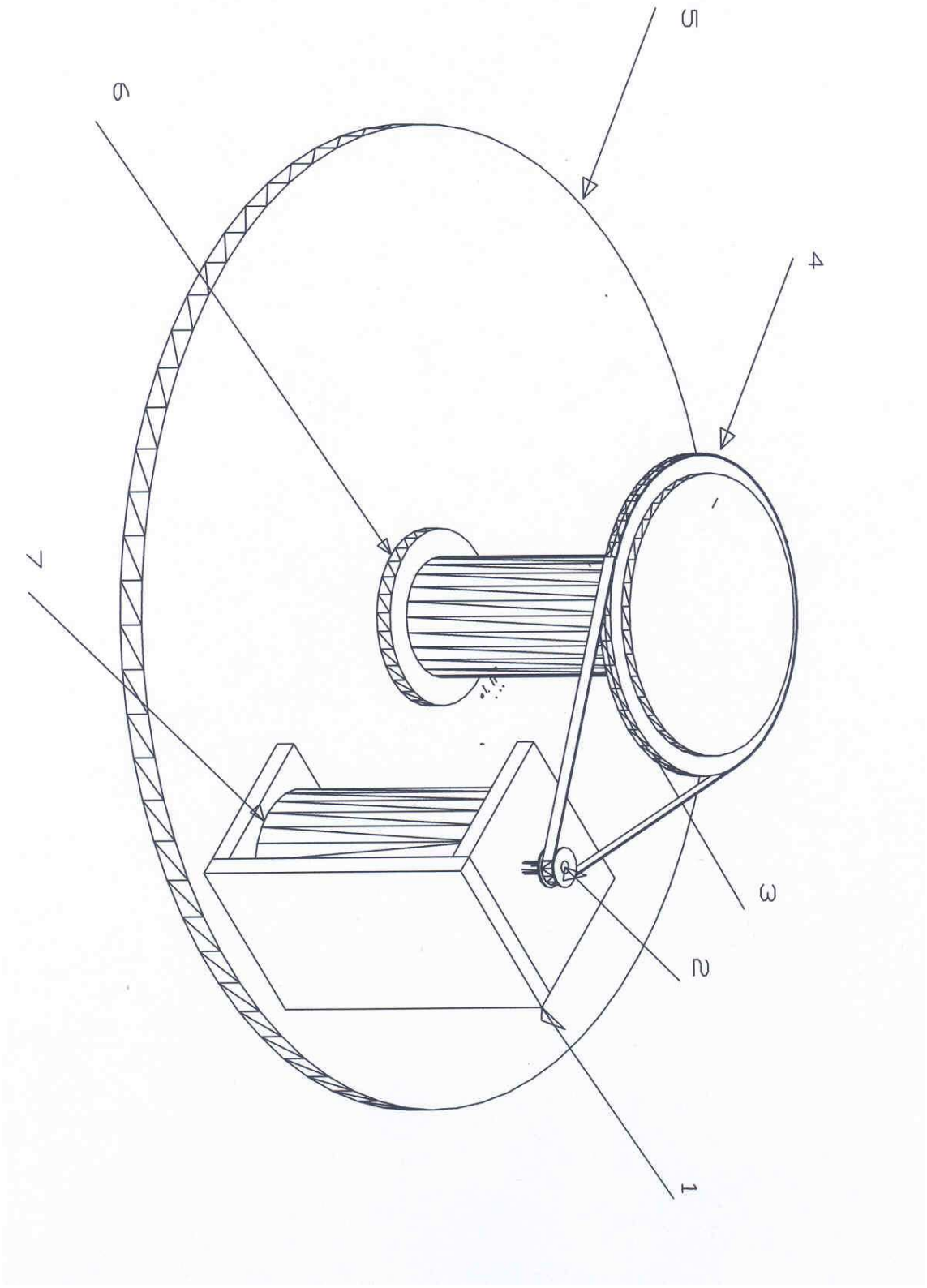


Figure 76: DOF 1 of master arm and its transmission system

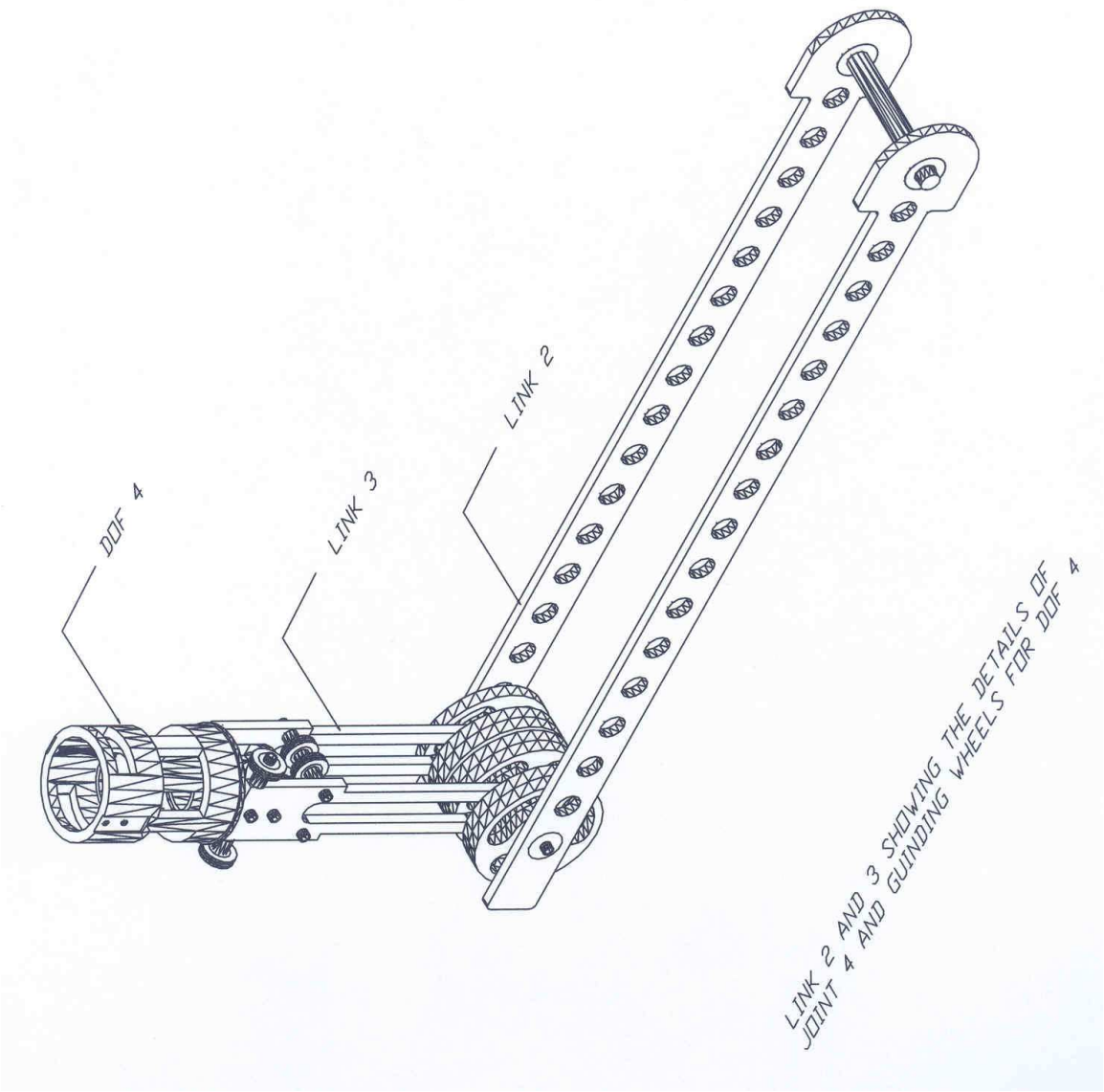


Figure 77: Links L2 and L3 and some details of rotation mechanism of DOF 4

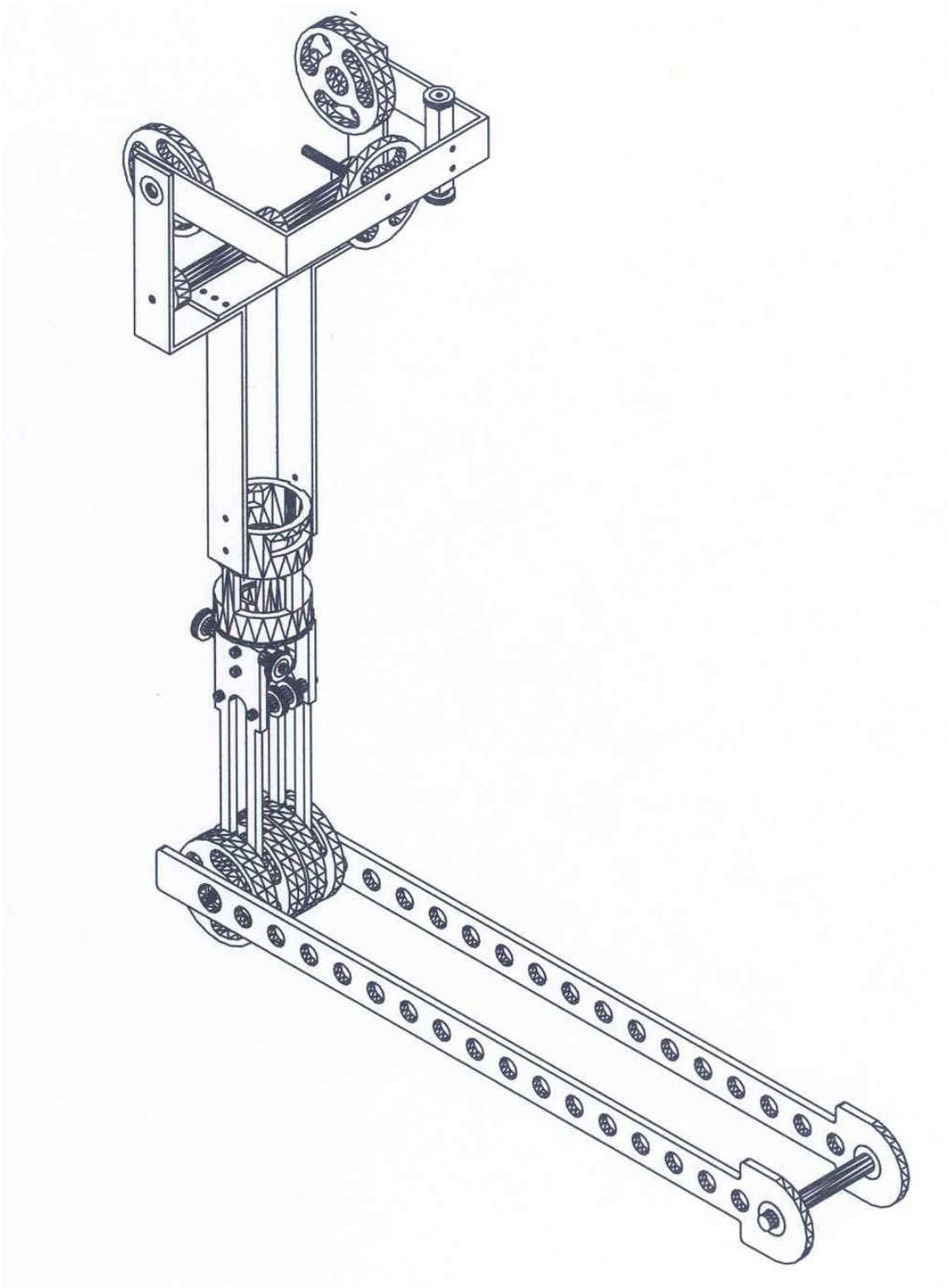


Figure 78: Links L3, L4, L5, L6 with the details of master arm handle

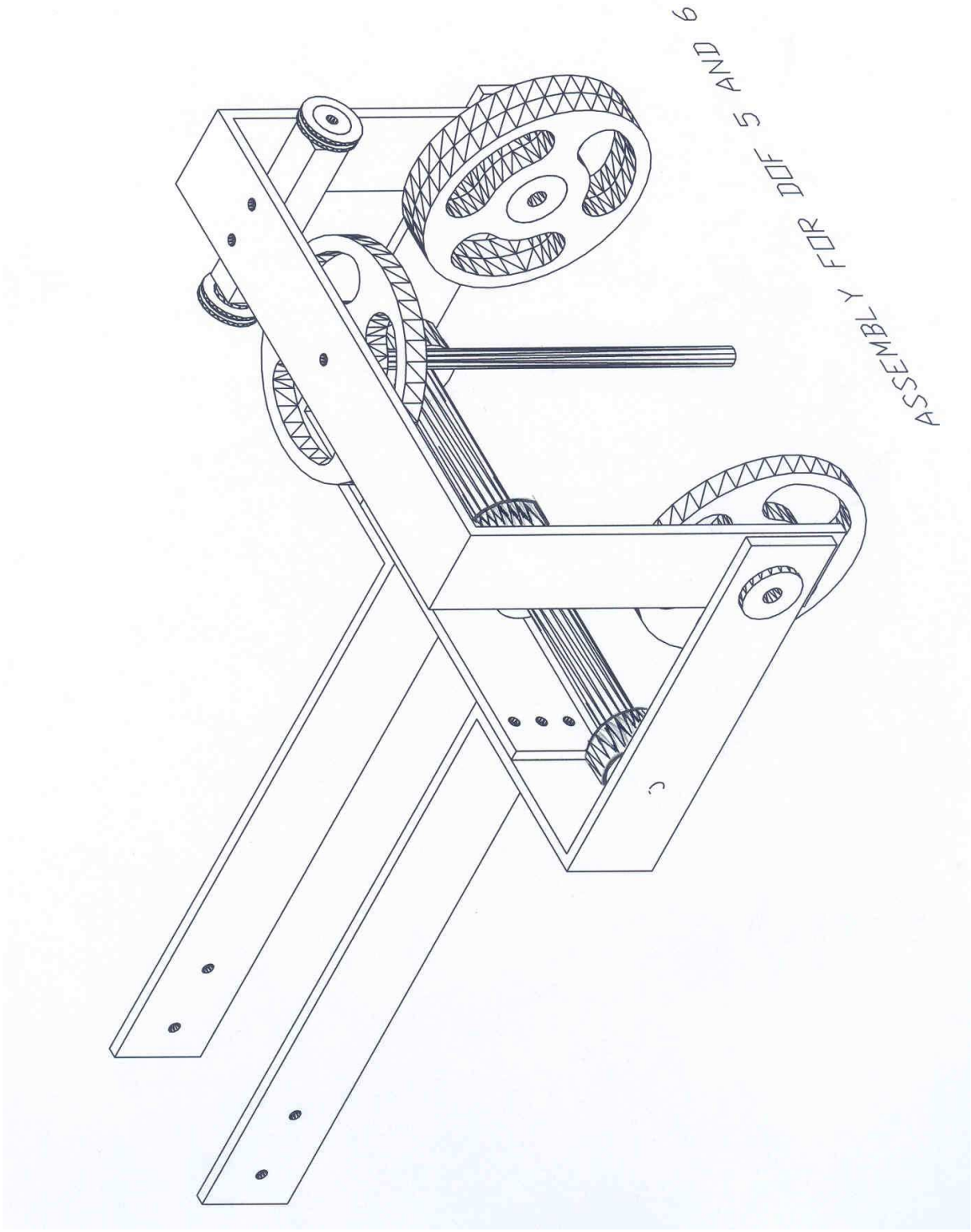


Figure 79: Assembly of DOF 5 and 6 with its transmission wheels

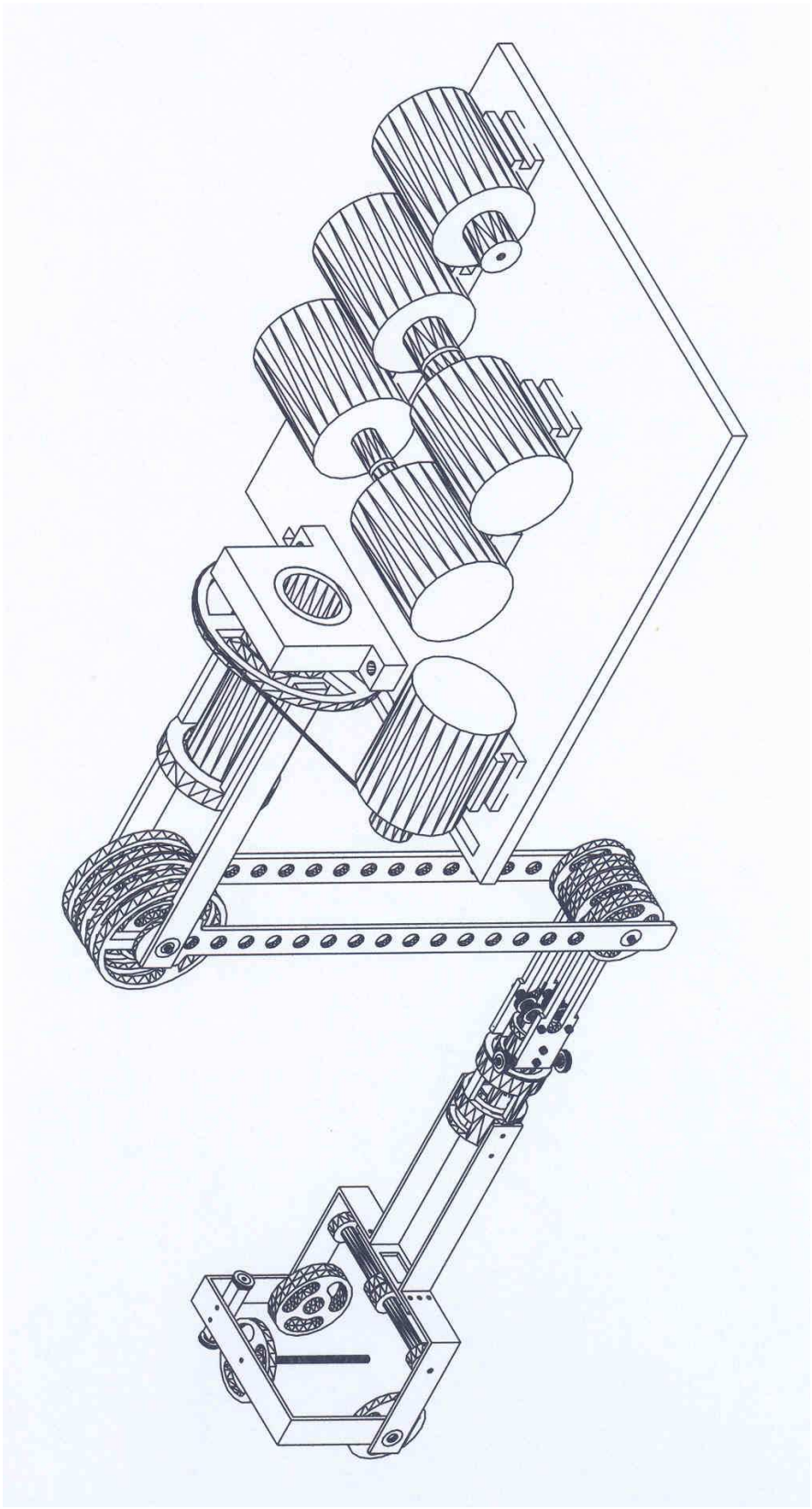


Figure 80: Overall assembly of the master arm with its motors

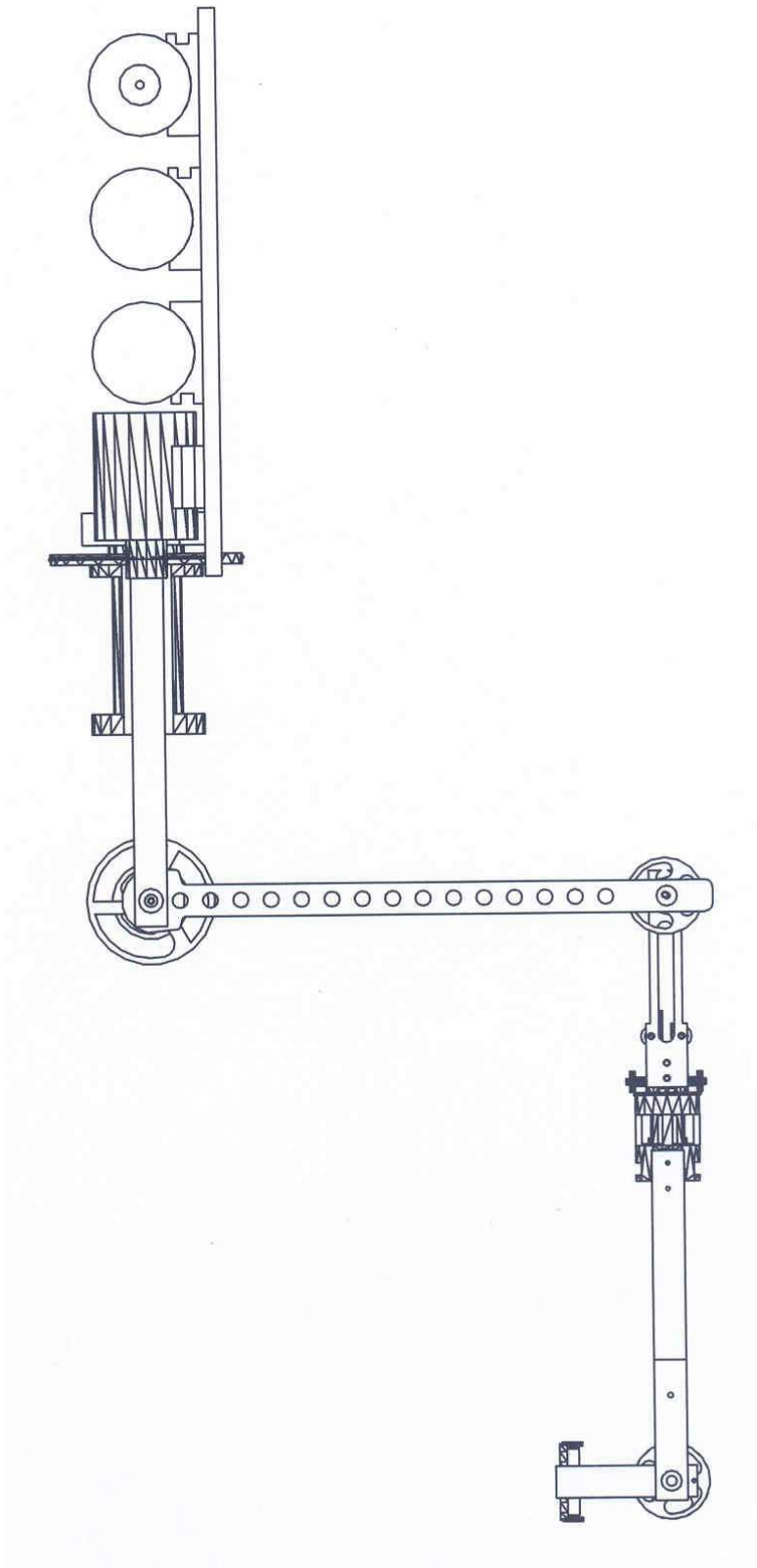


Figure 81: Side view of overall assembly of the master arm

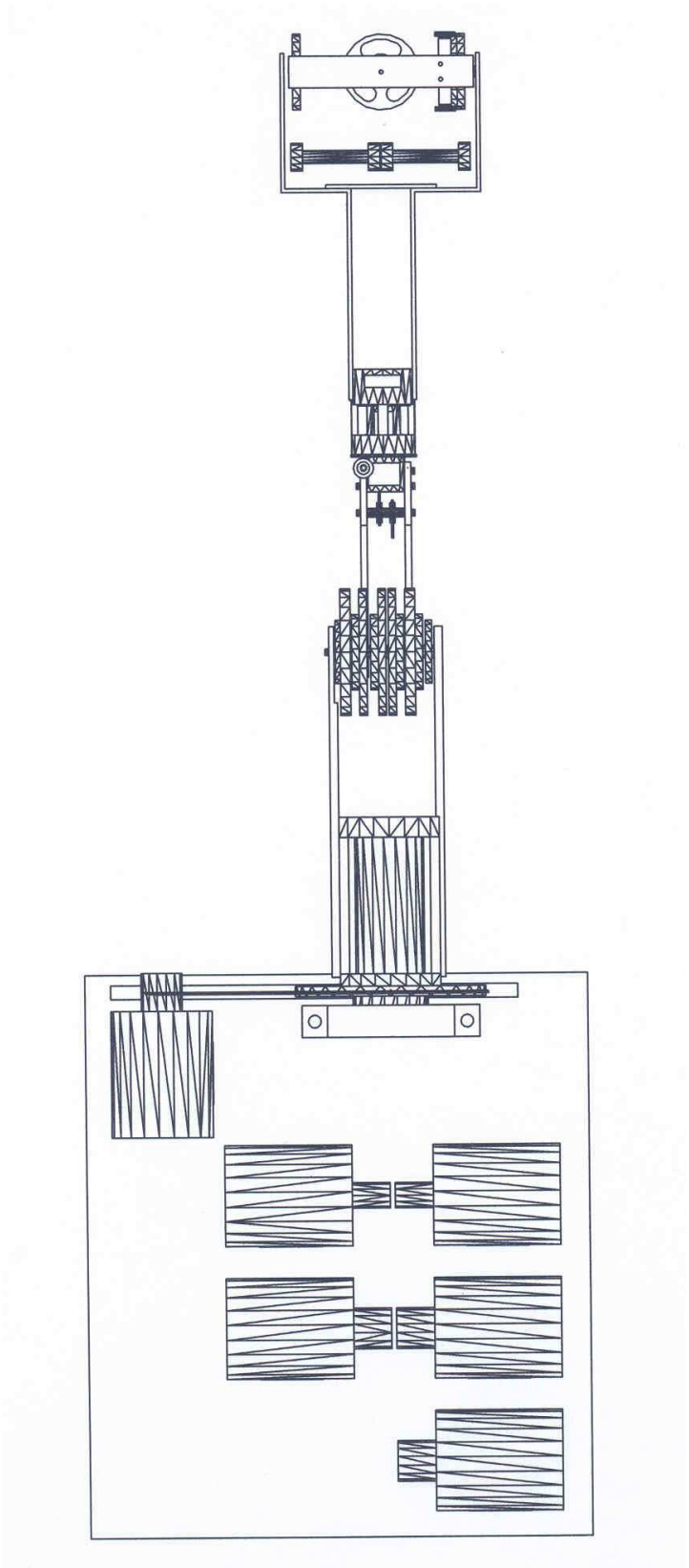


Figure 82: Top view of overall assembly of the master arm

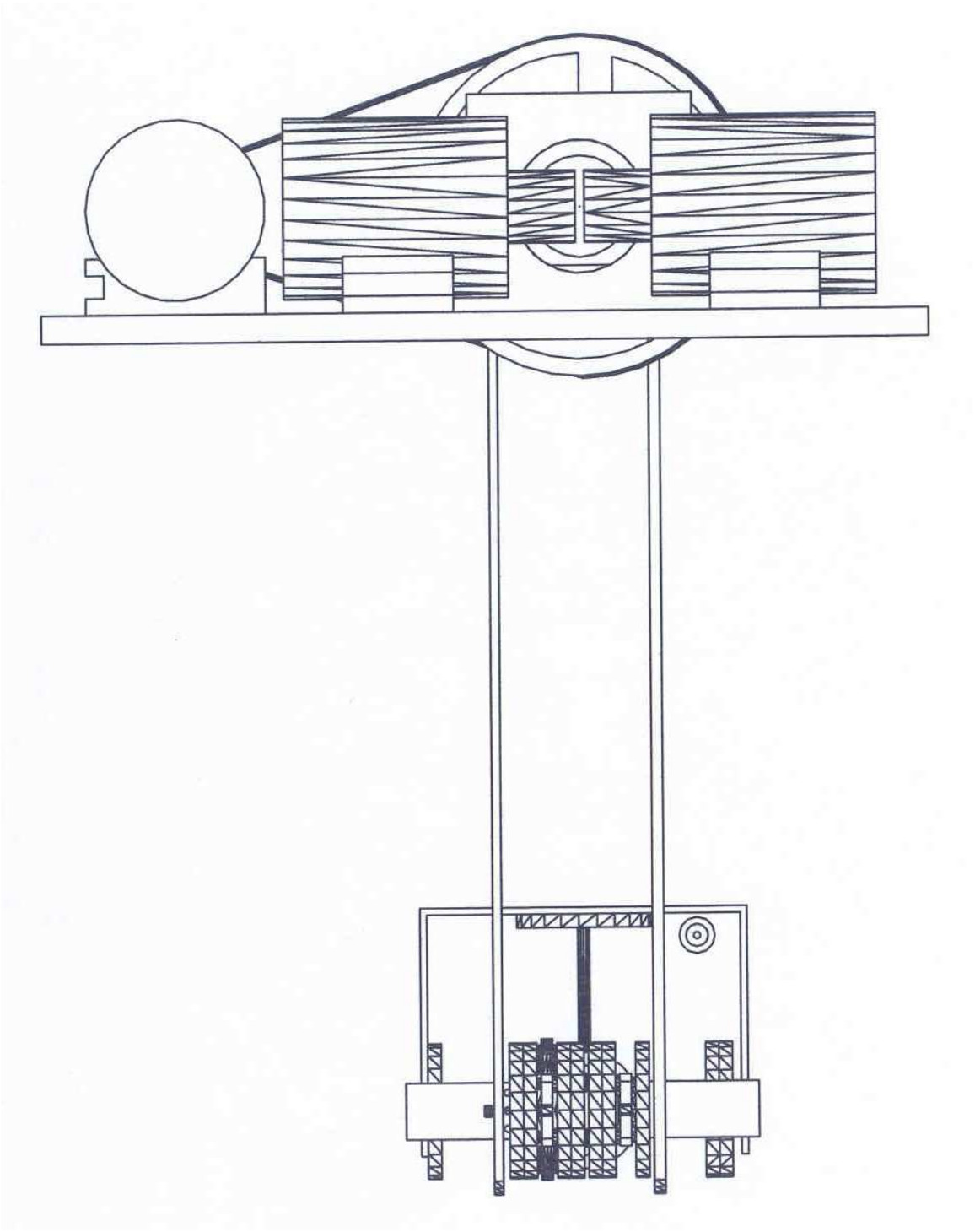


Figure 83: Bottom view of overall assembly of the master arm

10 APPENDIX C: Detailed drawing of the master arm

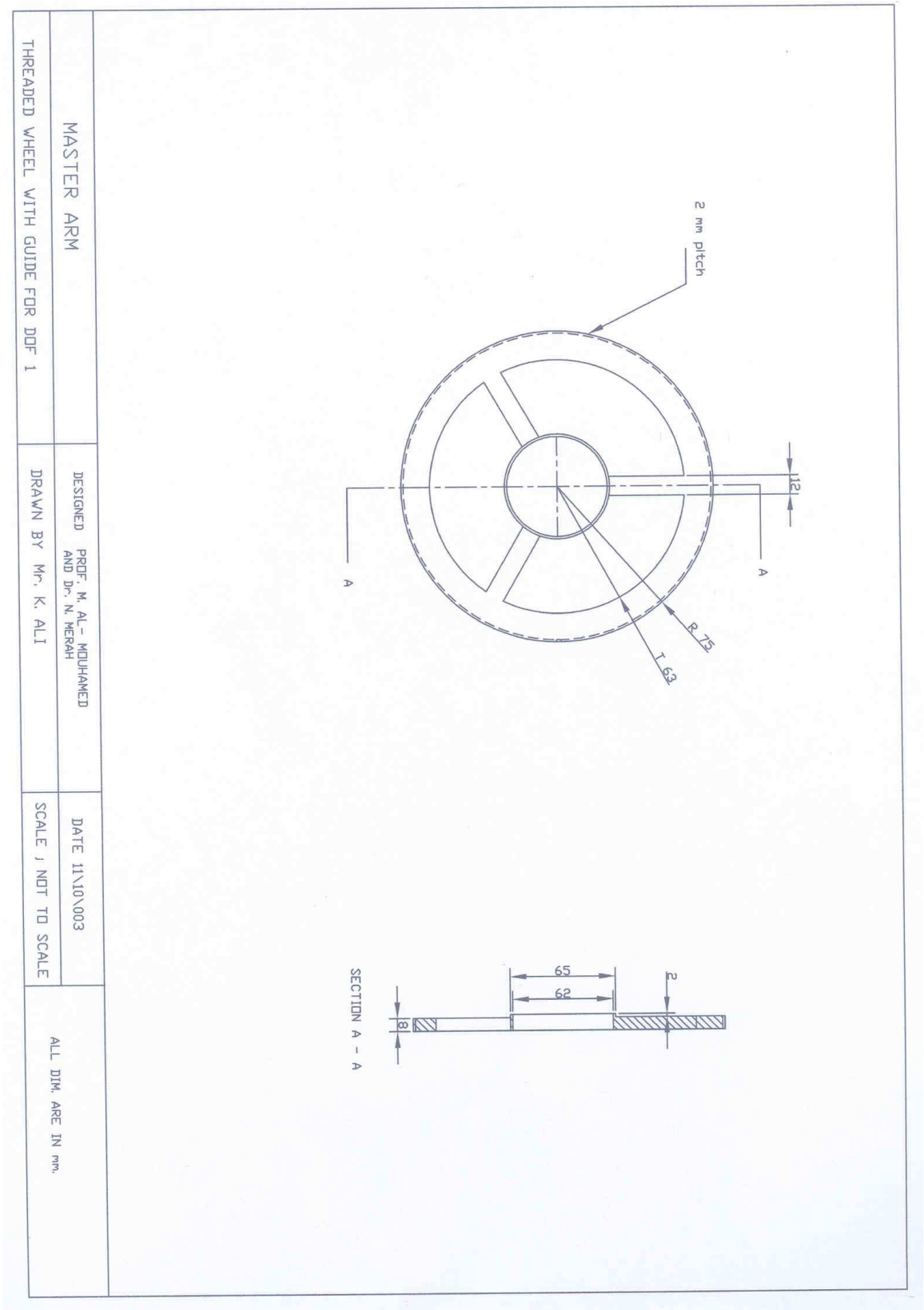


Figure 84: Threaded transmission wheel with guide

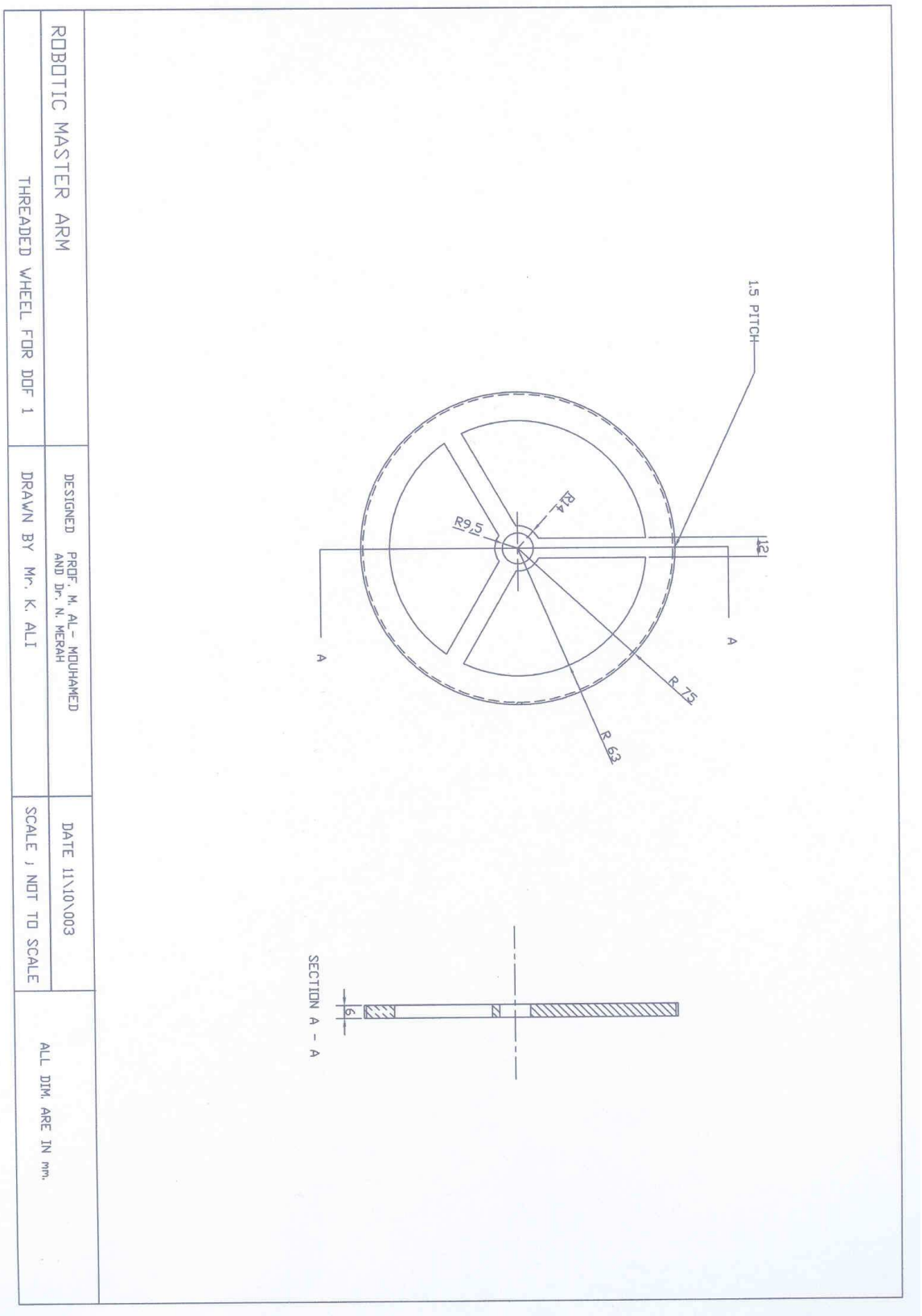


Figure 85: Threaded transmission wheel for DOF 1

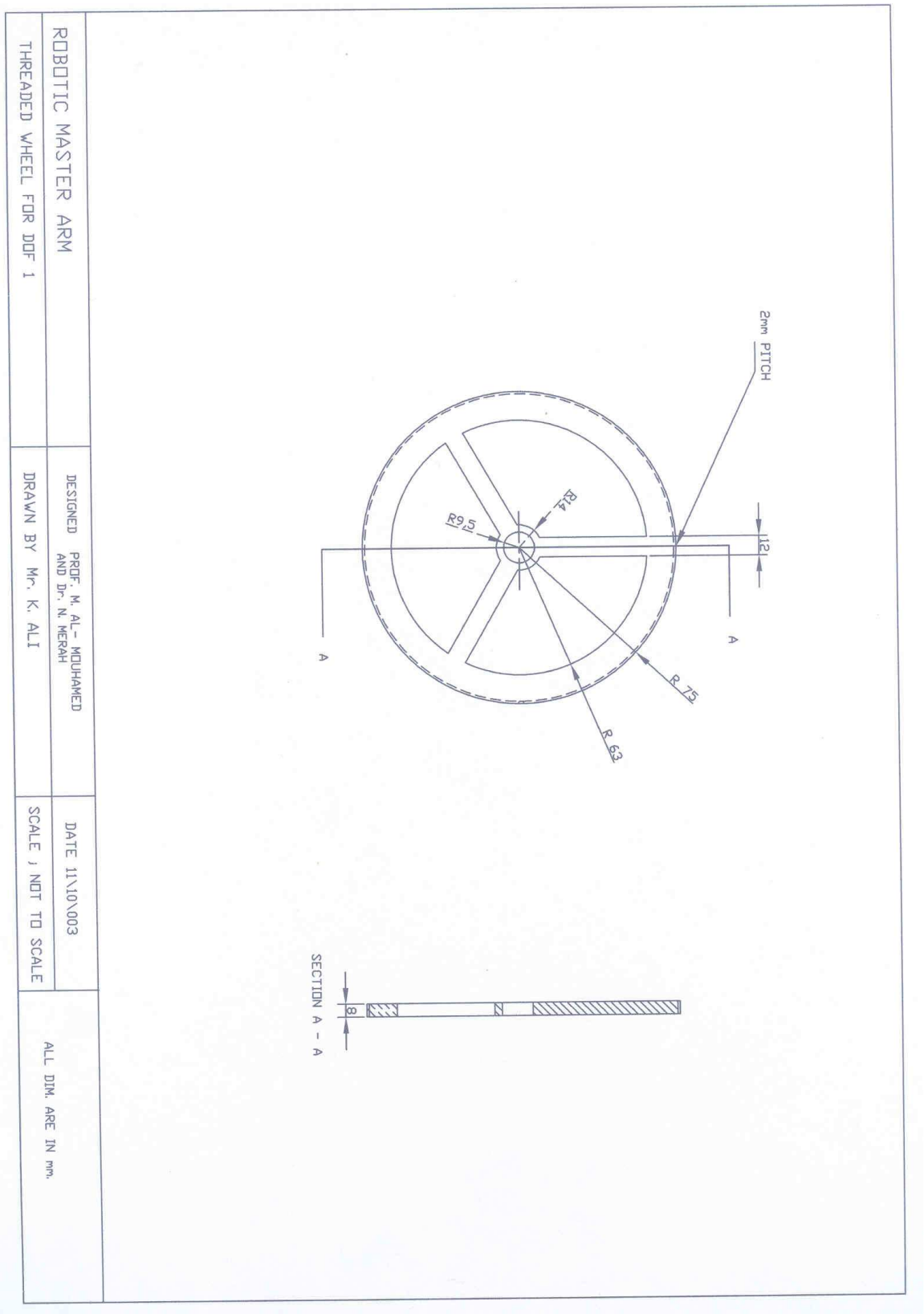


Figure 86: Inner and outer housing for DOF1

Figure 87: Inner and outer housing for DOF1

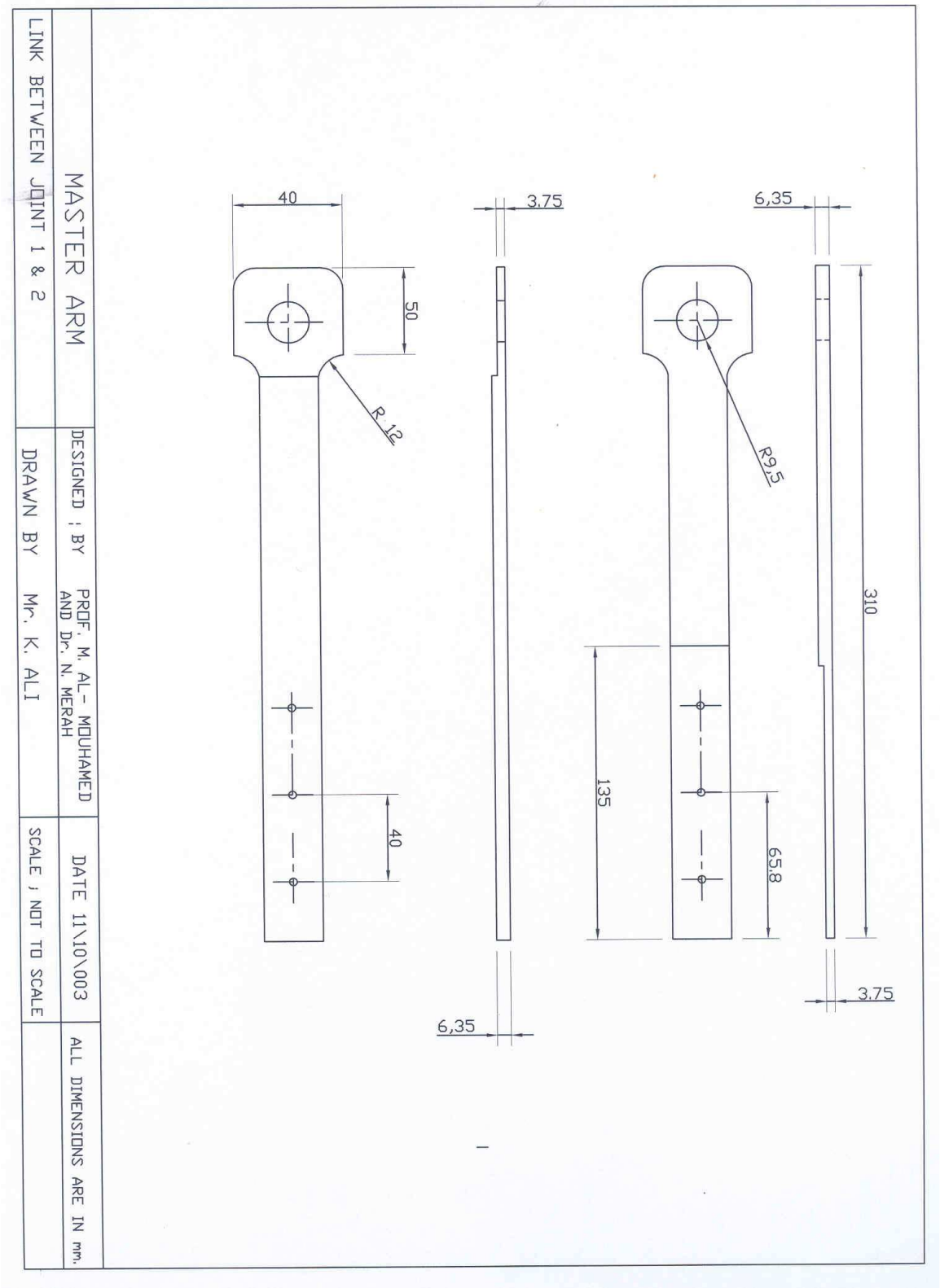


Figure 88: Link between joint 1 and 2

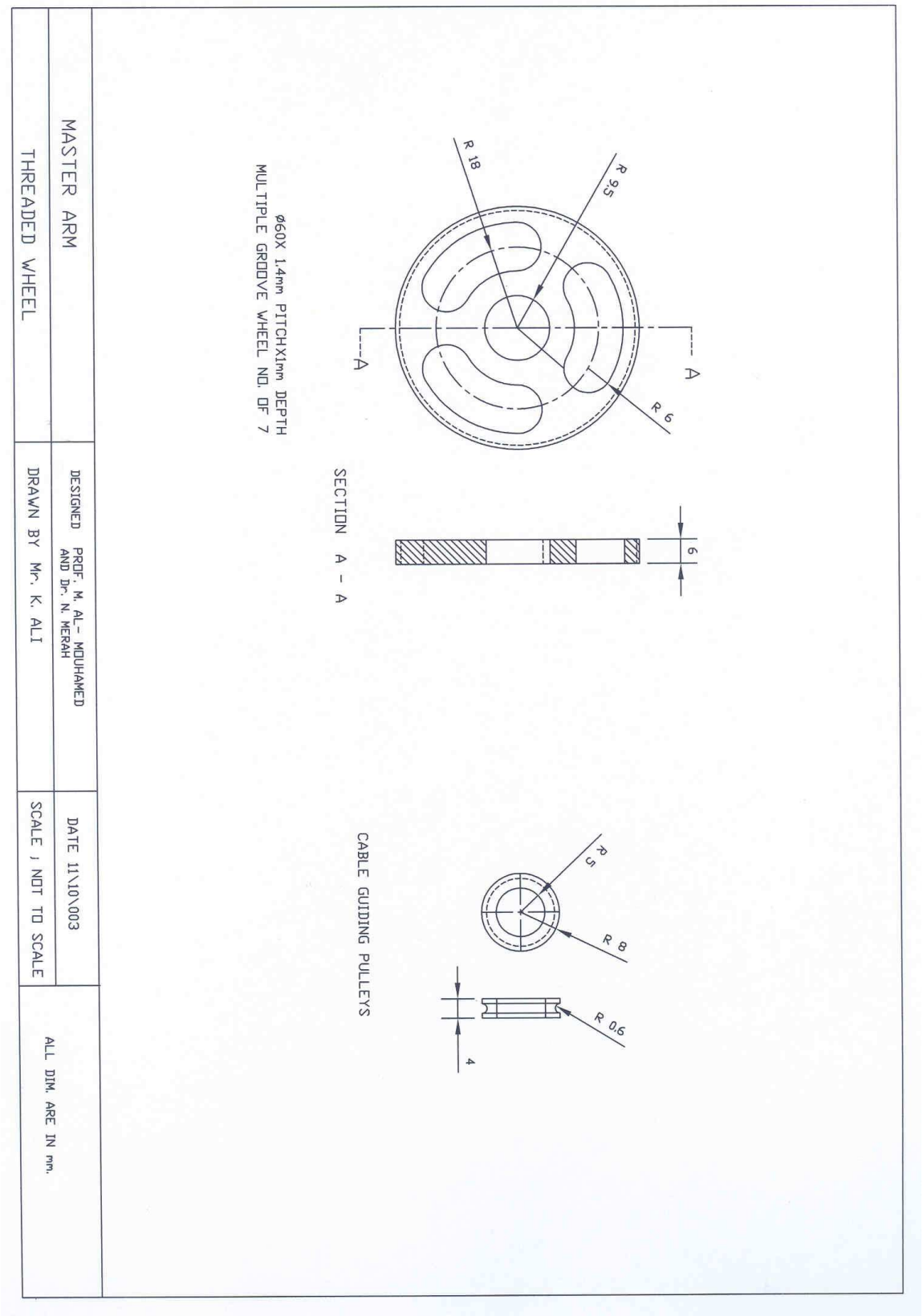


Figure 89: Multiple groove wheels and guiding pulleys

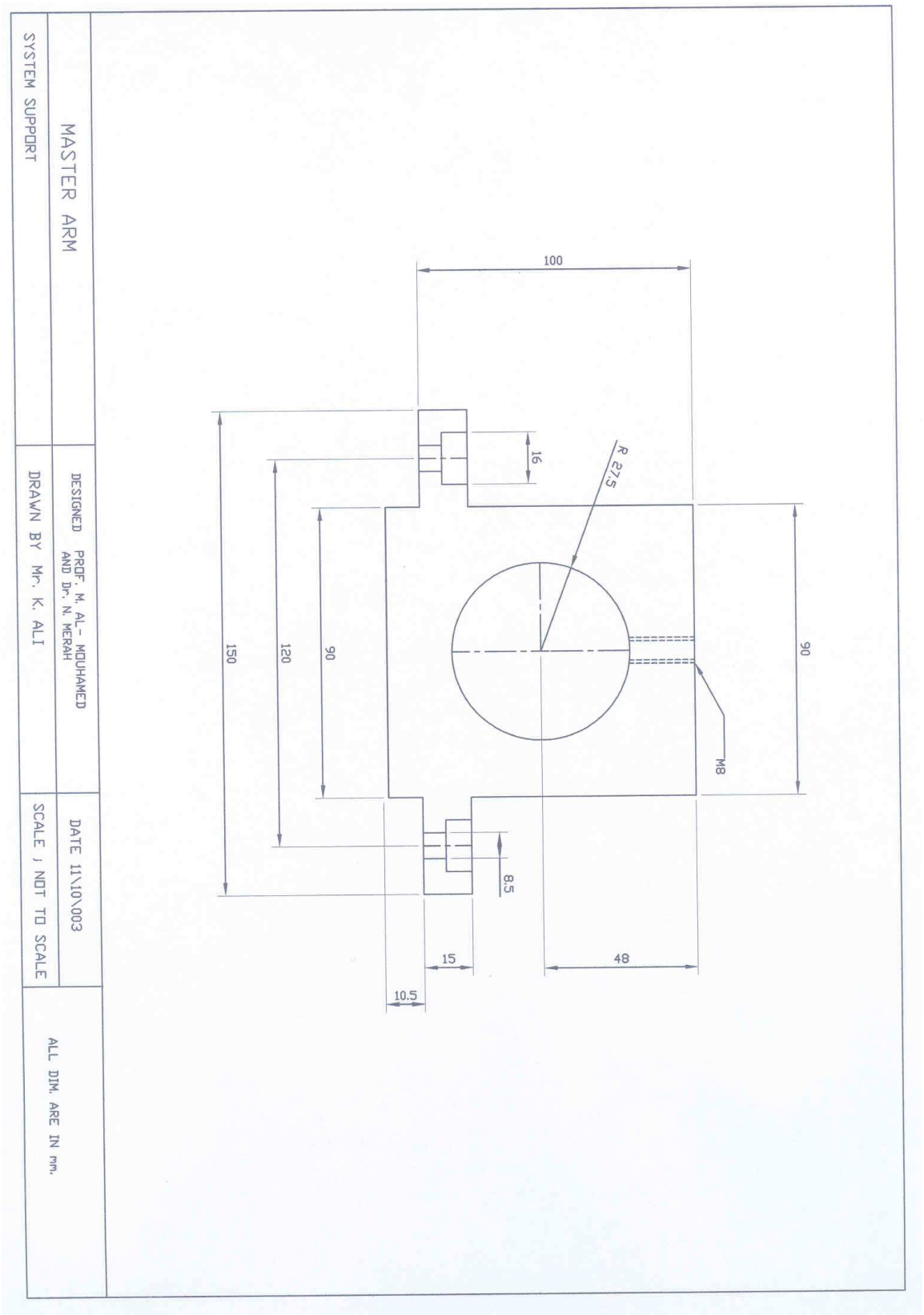


Figure 90: System support between DOF 1 and ground

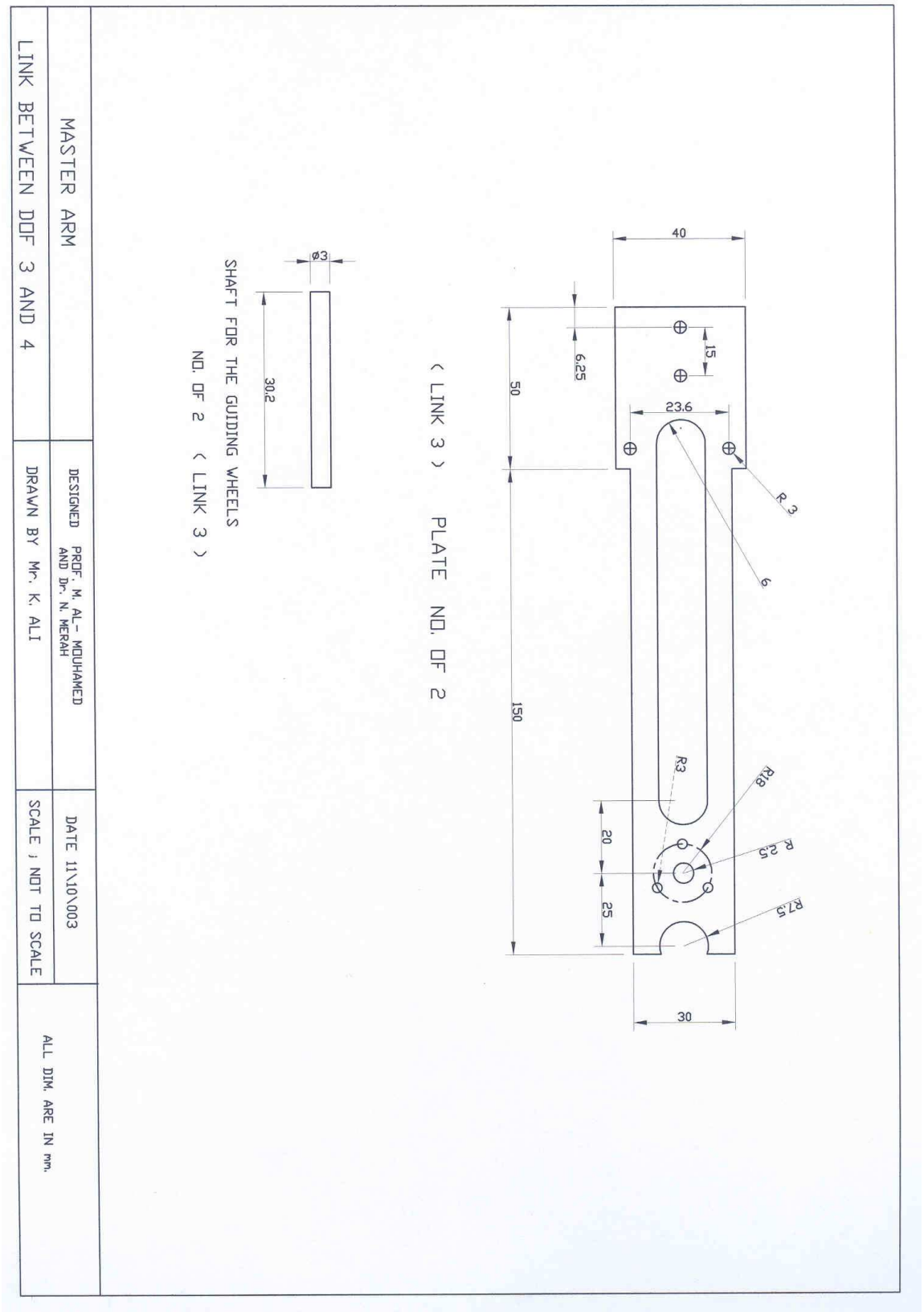


Figure 91: Link between DOF 3 and 4

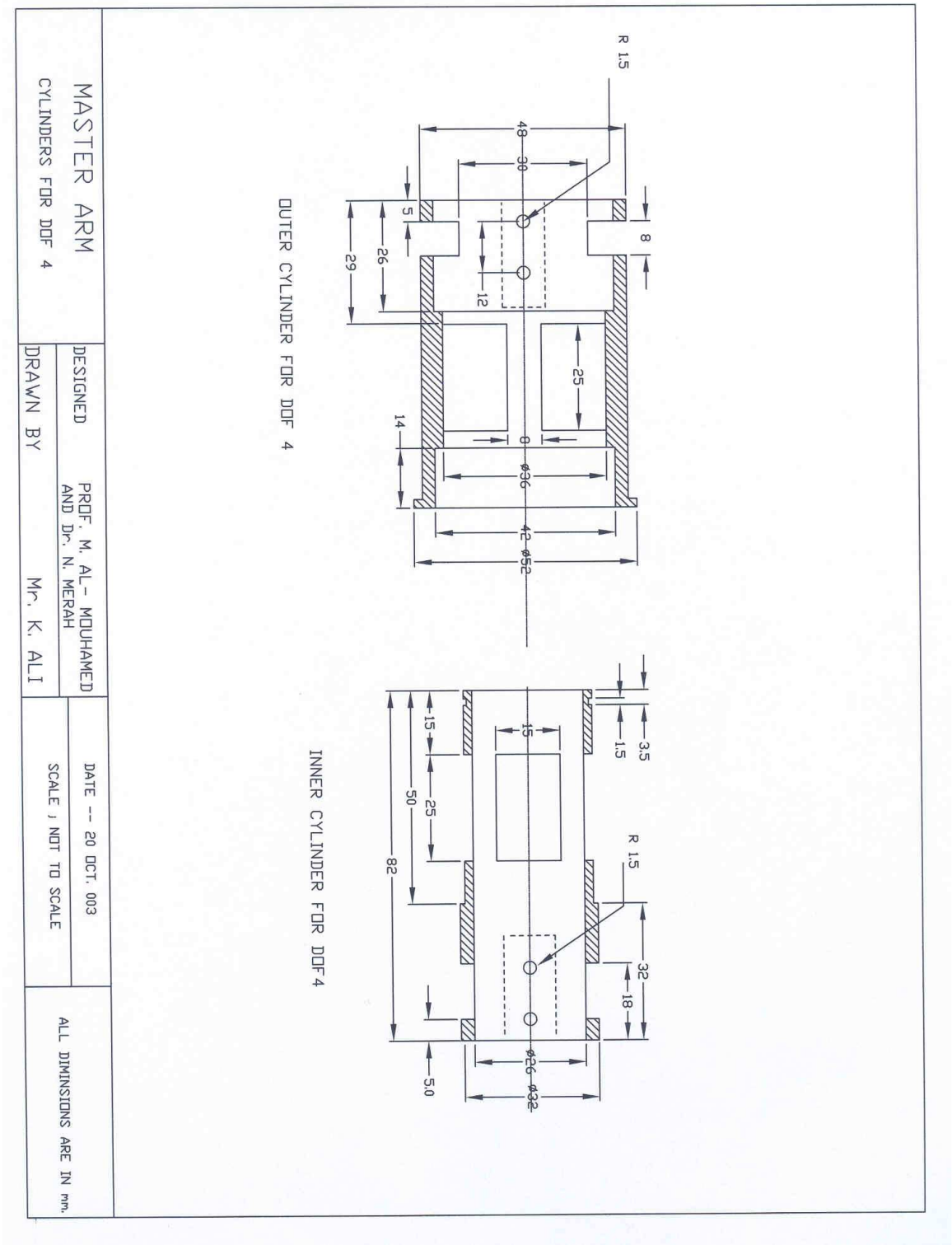


Figure 92: Cylinders for DOF 4

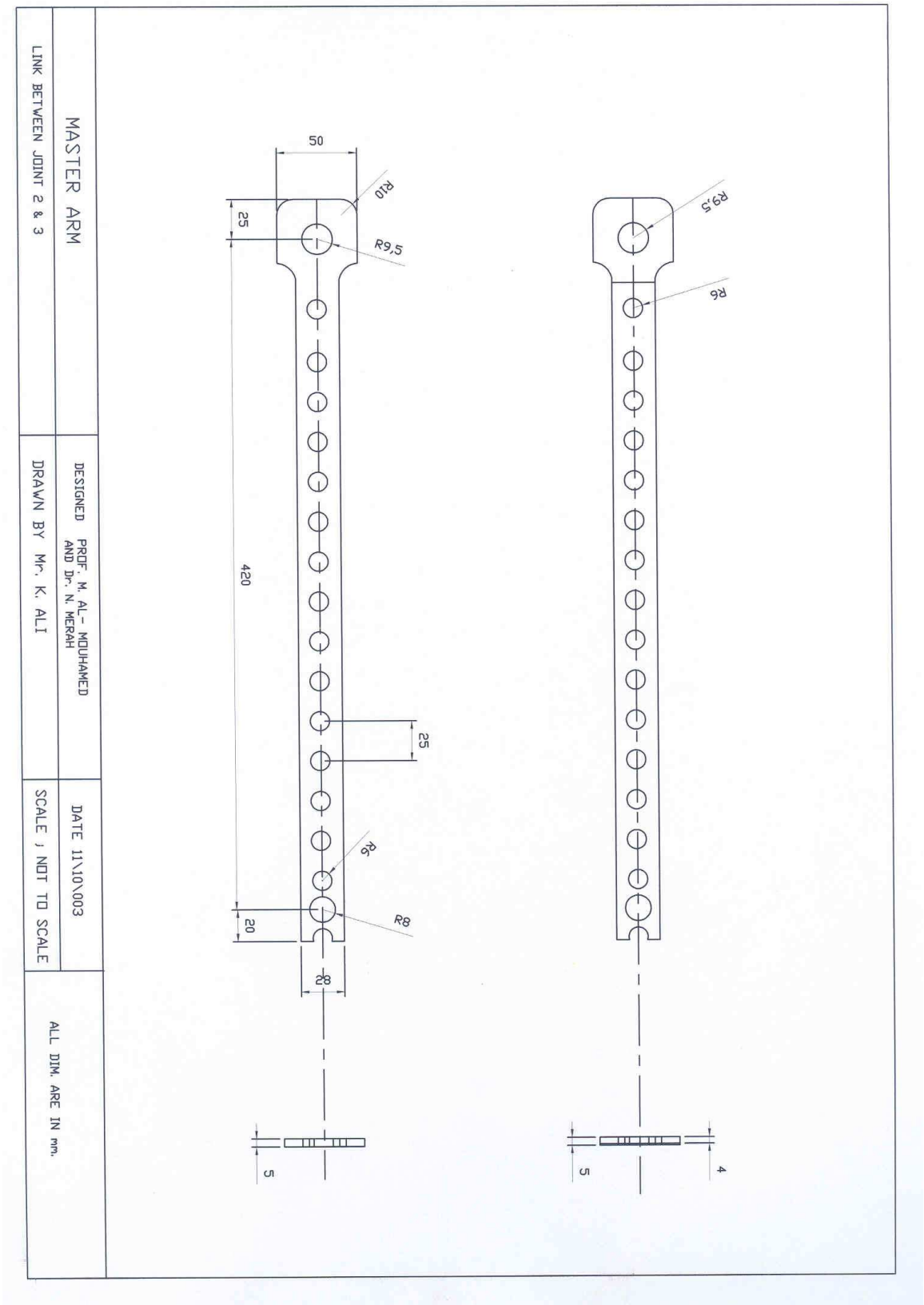


Figure 93: Link between DOF 2 and 3

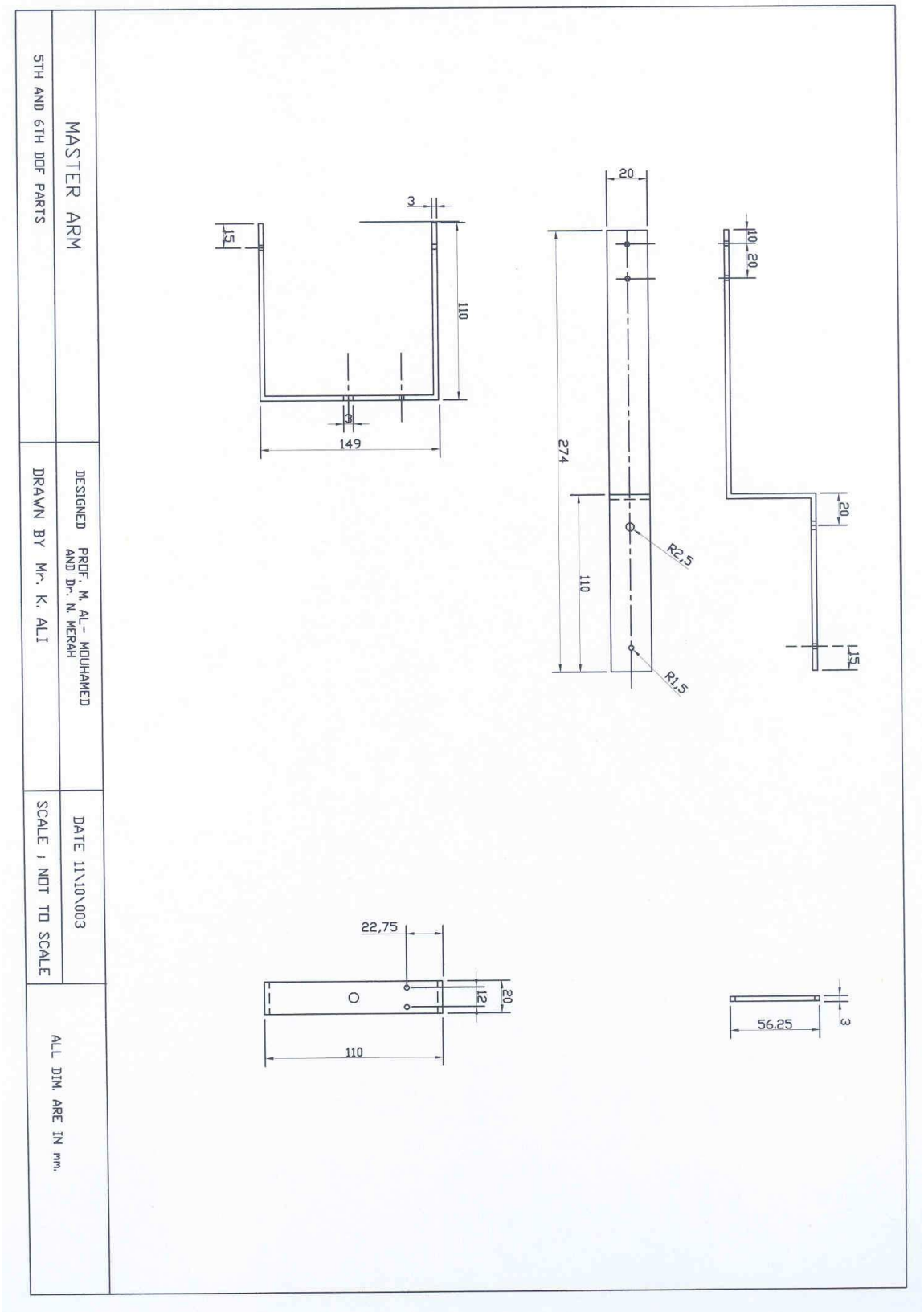


Figure 94: Detailed parts of DOF 5 and 6

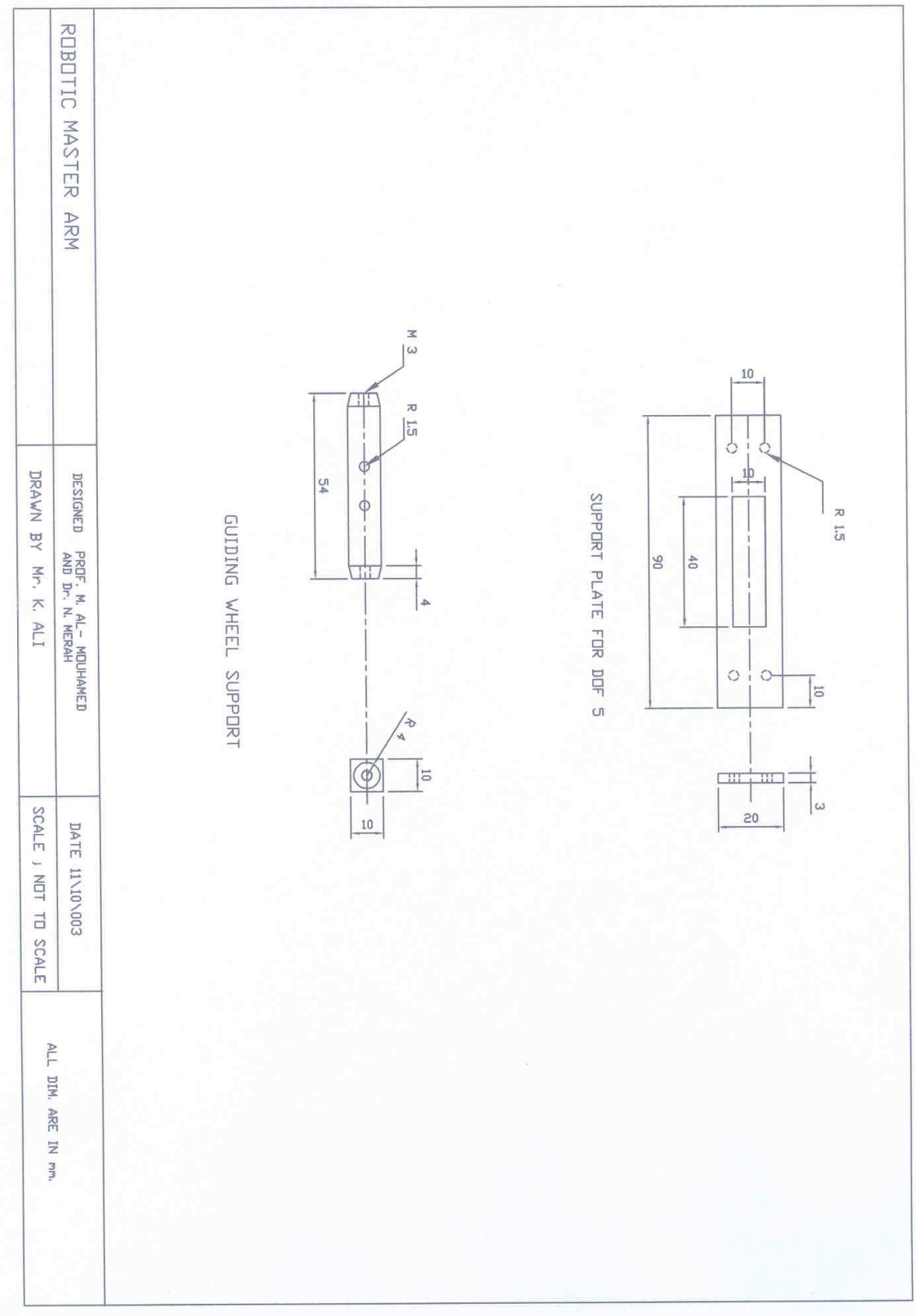


Figure 95: Support plate for DOF 5 and guiding wheels

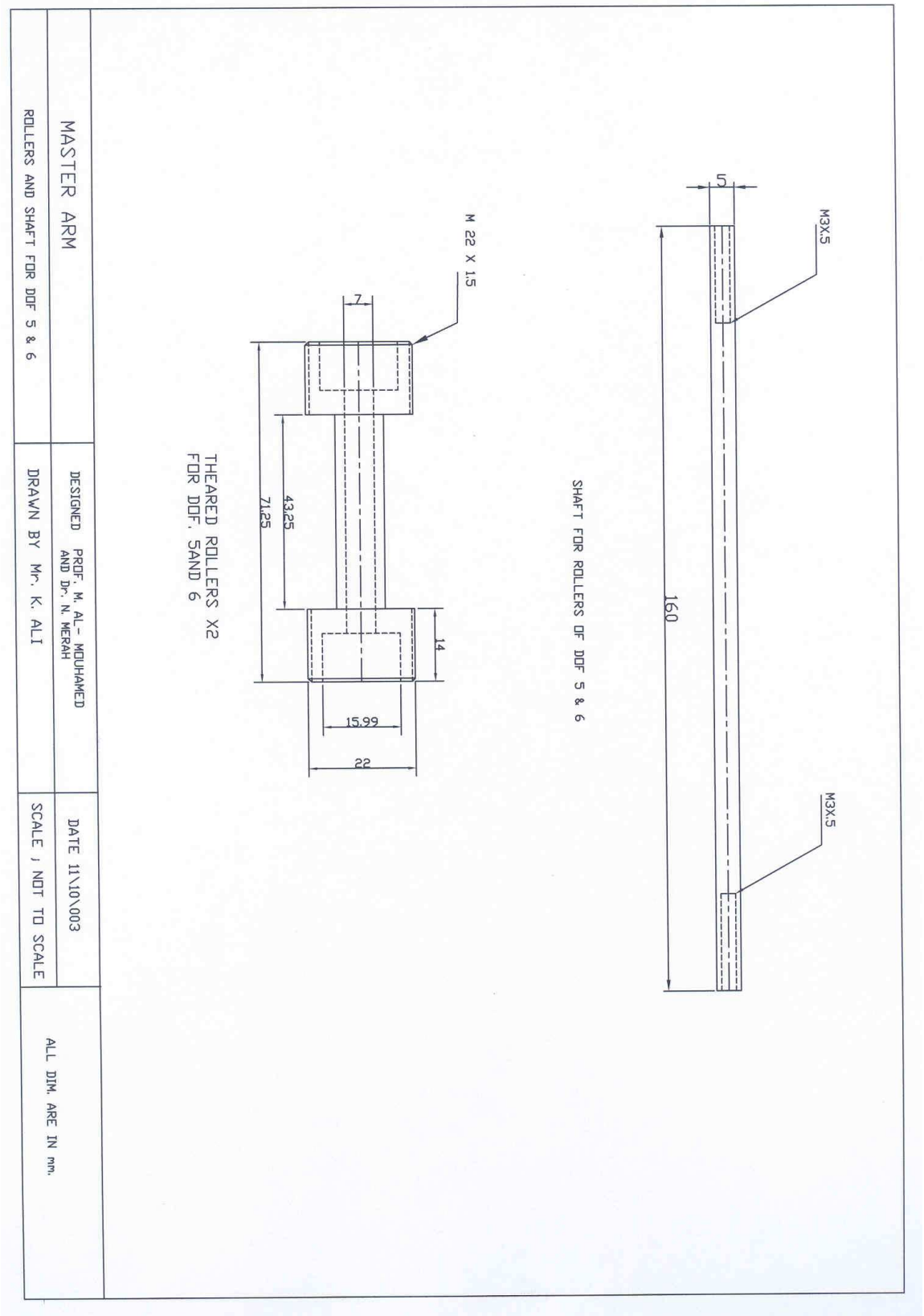


Figure 96: Rollers and shaft for DOF 5 and 6

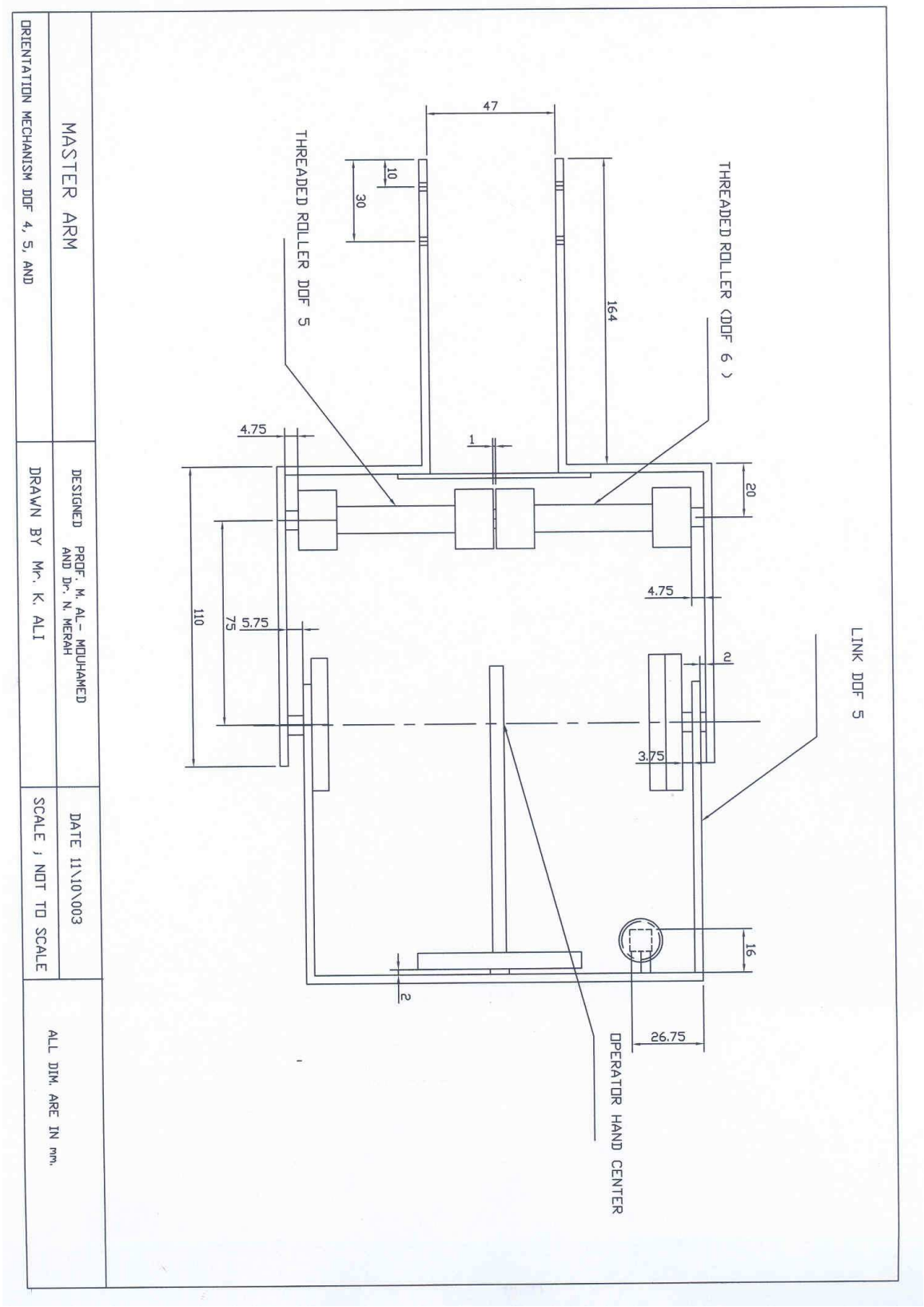


Figure 97: Details of mechanism for DOF 5 and 6