A Hierarchical Design Scheme for Application of Augmented Reality in a Telerobotic Stereo-Vision System

M. A. Al-Mouhamed,* Syed M.S. Islam,† S.M. Buhari ‡, and T. Al-Kharoubi §

Abstract

Operating a robot from a distant place using haptic and visual feedback gets enormous applications in various situations. But due to time-delay in the communication, an tele-operator sometimes has to go for a move-n-wait strategy. However, the problem can be minimized by using Augmented Reality (AR) concepts of superimposing virtual objects onto the real video image of the remote scene to create a simulation plan in the local machine. Operator can make trial and error to finalize his plan. This increases task safety and also reduces realtime network interaction by transferring only the finalized trajectory data. In this paper we have presented a hierarchical design scheme for developing such an AR system. At first a geometric model of a six Degree of Freedom(DOF) robot arm is developed. Based on that model state-of-art graphics system is used to model a 3D graphical arm. Then the graphics is superimposed onto the real image using accurate camera calibration and registration methods. Algorithms are also developed for activating motion in the visualization system. A graphical user interface is designed to facilitate the task simulation. The design scheme is implemented and tested using Microsoft .NET framework, visual C#.NET and Microsoft DirectX with a stereovision system comprising of a PUMA-560 robot and operating over a LAN.

Keywords: *Telerobotics, augmented reality, stereovision, camera calibration, 3D graphics rendering.*

1. Introduction

Telerobotics is a modern technology of robotics that extends an operator's sensing and manipulative capabilities to a remote environment. A telerobotic system consists of a master arm(workstation) and a slave arm(workstation) that are connected through a computer network and a stereovision system to provide 3D views of slave scene [see Figure 1]. Tele-operator is also provided with force feedback to have a better sense of his task manipulation. Teleroboitcs is now becoming very useful to be applied in many situations specially in scaled down and scaled up situations, hazardous and hostile situations and environment where human presence adversely affect the task operation.

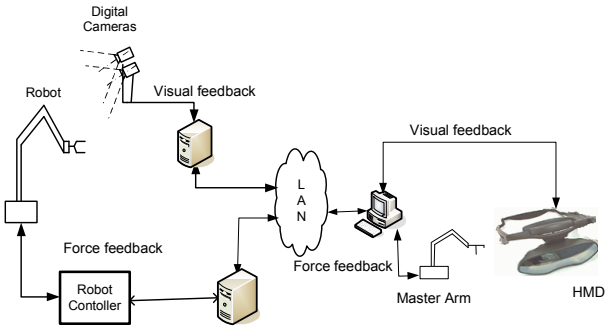

Figure 1. A typical telerobotic system

Telerobotics has enhanced the surgery through improved precision, stability and dexterity. Stereo image guided telerobots allow surgeons to work inside the patient's body precisely and without making large incision. Telerobots are now routinely used for biopsy brain lesions with minimal damage to adjacent tissue, for closed-chest heart bypass, for shaping the femur to precisely fit prosthetic hip joint in orthopedic surgery, for microsurgical procedures in ophthalmology and for surgical training and simulation. Information from various medical imaging sensors, such as CAT, PET, and MNR scanners can be used to generate graphic images of the interior of the human body. These images can be super-imposed onto a live video image of the body us-

*Department of Computer Engineering, College of Computer Science and Engineering (CCSE) King Fahd University of Petroleum and Minerals (KFUPM), Dhahran 31261, Saudi Arabia. Email: mayez/mibuhari/sislam123/talal@ccse.kfupm.edu.sa, iqbal@ipp.zess.uni-siegen.de

†Department of Computer Engineering, CCSE, KFUPM, Dhahran 31261, Saudi Arabia.

‡Department of Information and Computer Science, CCSE, KFUPM, Dhahran 31261, Saudi Arabia.

§Department of Computer Engineering, CCSE, KFUPM, Dhahran 31261, Saudi Arabia.

ing Augmented Reality(AR) tools, and seen in three dimensions, providing a clear advantage of systems that use flat two-dimensional displays. This will also provide the teleoperator e.g. surgeon with the facility of making simulation plan with probable rehearsal and corrections before going for exact operation with patient's body. Thus it will give additional safety in telesurgery. This graphical overlaying will also help to overcome the adverse effect of communication delay and saving bandwidth by sending less frequently only the finalized planned trajectory points.

AR is a system that combines real and virtual environment and which is interactive in real time and registered in 3-D Area based stereo.A comprehensive survey on AR is made in [1]which explored number of applications including medical visualization, maintenance & repair, annotation, entertainment and military aircraft navigation & targeting, interior design and many more. In robotics AR lies between telepresence (completely real) & Virtual reality (completely synthetic) and between manual teleportation & autonomous robotics [2].

To apply AR on a telerobotic stereovision system requires proper registration of real world i.e. robot workspace scene data (e.g. patient's anatomy in case of surgery) with the graphics image data. Proper registration on the other hand, requires accurate camera calibration technique to be used to align graphic camera co-ordinate to the real video camera co-ordinates. Camera calibration is the establishment of the projection from the 3D world co-ordinates to the 2D image co-ordinates by finding the intrinsic and extrinsic camera parameters [3]. Intrinsic parameters includes optical and electronic properties of a camera, such as focal length, lens distortion coefficients, image center, scaling factors of the pixel array in both directions. While the extrinsic parameters are the pose estimation (rotation and translation) of the camera system relative to a user-defined 3D world coordinate frame [4].

Registration referes to the proper alignment of the virtual object with the real world. The accuracy of registration is mostly dependant on the accuracy of calibration [3]. Two kinds of registrations are: Static and dynamic [4]. In static registration user and the objects in environment remain still. It is done at initialization with the help of Human Operator. Dynamic registration is done while the viewpoint starts moving to automatically update the registration data.

In [2] a telerobotic AR system is discussed that implements 3D view in the monitors. It uses position tracking through mechanical and optical sensors for camera calibration. Gomez, S.R et. al. [5] develops a virtual environment for teleoperation using OpenGL. Herve, J. Y. et. al. [4] proposed a model based camera calibration and registration for AR where camera moves. Abdullah, J. et. al. [3] described a pinhole camera model with weak perspective projection, correction for radial distortion of lens and display on monitors for telerobotic AR applications. Heikkila, J. [6] proposed a geometric camera calibration using circular control points that gives calibration accuracy up to 1/50 th of a pixel size. Marin, R. et. al. [7] proposed an Internet based high level telerobotic framework using Java3D and CORBa. Iqbal, A. [8] developed multi-threaded distributed telerobotic framework over LAN with video frame transfer rate of 17 fps. They used a fiducial frame of reference based calibration for drawing a small red ball in the most current position of the gripper to show the AR effect.

In this paper we have presented a design strategy for augmenting a telerobotic system with graphical overlays. A mathematical model of robot arm is developed which is then drawn and animated at first in graphic system and then seamlessly overlayed on the stereo-video received at the client side.

The organization of this paper is as follows. In Section 2 design strategy is described. Implementation aspect is described in Section 3. Performance of the system is discussed in Section 4 followed by conclusion in Section 6.

2 Design

Our design strategy for augmenting the stereovision system by overlaying graphics will be discussed in this section.

2.1 Design Methodology

The design methodology of our augmentation of the telerobotic stereo vision system is discussed as follows. Figure 2 shows the important steps.

- Developing the Mathematical Model of the Robot: A set of geometric equations are developed to build the mathematical model of the robot.

- Drawing Graphic Robot Arm and Other Graphical Objects: Having chosen the graphics software that will allow exchange of data into and out of the model, the next step is to build the graphic robot arm model and other graphic objects.

- Animating the Graphical Objects for Simulating Task: Having constructed the graphic model, codes are then inserted to enable manipulating the graphics with the interface by handshaking and exchanging data to and from the model and interface. Algorithms are developed for the movements.

- Interfacing to the Telerobotic Stereovision System: Graphics sub-system is interfaced with the stereovision system..

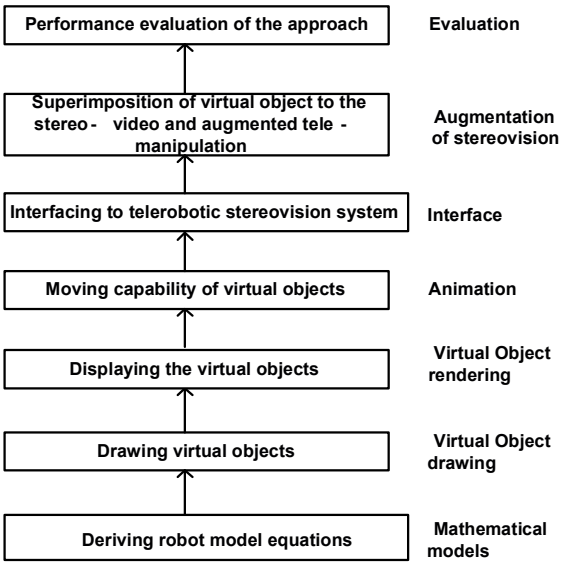

Figure 2. Hierarchy of Design Abstraction

- Identification of Camera Calibration Technique and Superimposition of Graphic Model into the Real Image: Appropriate camera calibration and image registration techniques is determined to perfectly superimpose the graphic model into real video image captured in server PC and received in client PC through network.
- Augmented Tele-manipulation: Once the methodology of data transfer is stabilized, tapping of all necessary data is implemented to perform the AR telerobotic manipulation.
- Evaluating the Design:Lastly, the full integration is tested.

2.2 Mathematical Model of the Robot Manipulator

PUMA 560 robot is an industrial robot with six degree of freedom i.e. there are six independent variables (or coordinates) to completely specify the configuration of its the mechanics. There are six links namely, Trunk or Base, Shoulder, Upper Arm (the Inner Link), Fore-arm (the Outer Link), Wrist and Gripper in this robot. Links are connected as a serial chain.

All the six joints of PUMA-560 robot arm are revolute joints. Each of the links connected with revolute joint is attached with a frame of reference, $R_i(x_i, y_i, z_i)$ to describe its position and orientation. Link L_{i+1} in Figure 3-(c) rotates with respect to link L_i when frame R_{i+1} rotates with respect to either axes of $x_i, y_i or z_i$. The end point O_{i+1}

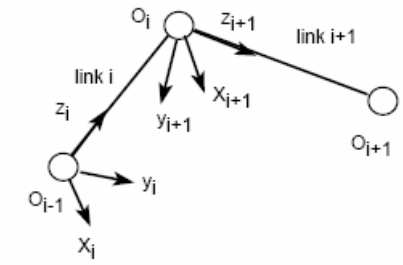

Figure 3. Two rotational links attached with frame of references

of L_{i+1} can be associated a vector O_iO_{i+1} which is 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:

$$O_iO_{i+1,i} = M_i^{i+1}.O_iO_{i+1,i+1} \quad (1)$$

where $O_iO_{i+1,i+1}$ denote the vector O_iO_{i+1} observed in frame R_{i+1}.

Now the position and orientation of O_{i+1} with respect to O_{i-1} can be expressed as:

$$O_{i-1}O_{i+1,i-1} = O_{i-1}O_{i,i-1} + M_{i-1}^{i+1}.O_iO_{i+1,i+1} \quad (2)$$

Similarly we can express the position and orientation of gripper or the end effector with respect to the base frame of reference which is kept fixed. Therefore, the geometric model of robot manipulator can be expressed as the following expression which provides the position and orientation of each of the n (here six) links given the n (here six) joint angles θ.

$$G(\theta) = O_0O_{n,0}(\theta), M_0^n(\theta) \quad (3)$$

where

$$M_0^n = M_0^1.M_1^2.M_2^3...M_{n-1}^n \quad (4)$$

$$O_0O_{n,0} = O_0O_{n-1,0} + M_0^n.O_{n-1}O_{n,n} \quad (5)$$

On the other hand inverse geometric model of the robot manipulator can be expressed as following which provides us new value of the joint angles given the new position and orientation of the end effector.

$$\theta = G^{-1}(X_{new}, M_{new}) \quad (6)$$

The details of our geometric and inverse geometric model of PUMA 560 robot manipulator is described in [9].

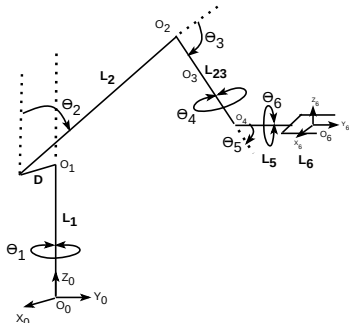

Figure 4. The kinematic model of the PUMA 560 robot arm

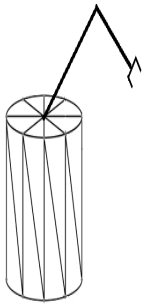

Figure 5. Finite element representation of a cylinder used as body shape of graphical arm

2.3 Building Body Shapes Around the Skeleton of the Graphical Arm

The mathematical model developed so far gives us the skeleton of the Robot arm as shown in Figure 4. To develop a solid robot arm body we need to draw polygonal shapes around the skeleton to look like real PUMA 560 robot.

The base, shoulder and the wrist part of the robot are almost cylindrical. The upper arm and the forearm are somewhat trapezoidal. Since for telerobotic manipulation our main focus is on the gripper or end-effector's middle point(between two openings) we have simplified the model by drawing the upper arm and forearm as cylindrical. We have modeled the gripper with rectangular shapes that can be opened or closed by changing the distance between the openings. We have also made simplification in making connection of link2 and link3. We have ignored the horizontal shift in both cases and hence those two links have become aligned vertically. But we were cautious about the end points of the polygons specially of the base cylinder so that we can match our graphic robot with the original robot's image in the stereo space.

Cylinders can be drawn as finite element representation [see Figure 5]. Increasing the number of segments or elements will improve the quality of view but it will increase the complexity of the computation. To draw a cylindrical shape, we need to specify the number of segments. The more segments there are, the smoother and rounder the cylinder will appear.

We have alternatives of primitives to be used for drawing the segments of the graphical arm such as line list, line stripe, triangle list, triangle stripe or triangle fan. We have chosen triangle stripe for the side wall and triangle fan for the to and bottom of the cylinder. Because among the alternatives triangle stripe and triangle fan are most memory efficient since they require less number of vertices to be specified.

2.4 Data Structure Design

To design an efficient data structure we must fist consider types of data and nature of manipulation. The robot structure developed in Section 2.2 reveals that PUMA 560 robot is an object of variable geometry where one portion of the object is connected with other portion. Since each link of the robot are represented by position and orientation matrices, movement to a new position or orientation involves multiplication of two 3×3 matrices and one 3×3 matrix with one 3×1 vector. And we have to perform these computations for each point/vertex of the graphical robot. And in Augmented Telerobotic systems robot arm has to be dynamically updated quickly and accurately.

Although modern computers' CPU performance is very high, CPU execution time is very crucial in the field of telemanipulation as the CPU has to deal with some network aspect, data acquisition and display as well. To reduce the execution time, we must reduce the computations which eventually necessitate choosing efficient data structure to store vertex data. If we store all the vertex data of the whole robot arm in one array then we need to perform all the computations even if we change a small portion of the robot arm. An efficient approach would be to keep geometry data of each link of the robot apart from the other link. In that case, for making change in any link, it will be sufficient to re-compute data of that link and only of those follow it.

We should also be careful about the flexibility and generality of the application such that the data structure can be used for drawing different type of robots.

To meet the above objectives we have modeled each link as a different object with properties required to represent the link [see Figure 6. Thus geometric data of each link is kept apart from the other link but each one is linked with its upper and lower links. The configuration data such as number of links, number of segments in cylinder etc are kept in separate data file. and vertex data of each link is associated with separate vertex buffer.

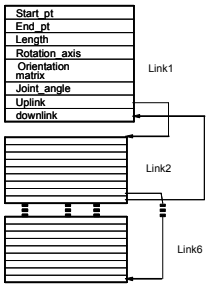

Figure 6. Data structure to represent the links

2.5 Displaying the Graphical Arm

Given the graphical arm structure in its model space to display it on the screen we need to specify the following.

2.5.1 Scene layout setup

Appropriate transformation matrices are defined to setup the scene. World transformation matrix is used to define the relative position of the objects. Camera position, target position and viewing angle are defined to project the graphic object accurately. A material with the diffuse and ambient color and a directional light its direction and finally an ambient light that scatters and lights all objects evenly are also defined to setup lighting.

The drawing can be represented using wire-frame model or solid model. The psychological studies and the performance studies in telerobotic system of show that there is no significant differences in using different viewing models [10]. But the aircraft simulation studies of [11] suggest that there should be an effect of different viewing models and [2] reports some advantages of using wire-frame model in real-time animation system specially by reducing the occlusion of graphics with real stereo image.We like to provide the user with a choice to select the most appropriate alternative.

2.5.2 Rendering

Since we are not interested to photo-realistic image quality, we have chosen polygon-based rendering for projecting the 3D models onto a 2D image plane. We have used Direct3D's scanline rendering technique as it is faster than ray tracing rendering. It works on a point-by-point basis rather than polygon-by-polygon. To deal with hidden surface determination, we have used Z-buffering algorithm. This technique employs an extra buffer to store the depth of the rendered value at each pixel. If the depth of the polygon that is currently being rendered at that pixel is less than the z-buffer value at that pixel, the output pixel is overwritten, otherwise nothing is drawn.

A sample view of the displayed graphical arm in solid and wire-frame model with 50 segments in each cylinder is shown in Figure 7.

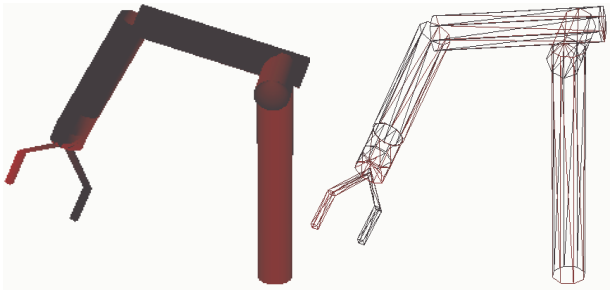

Figure 7. 3D PUMA 560 robot structure using cylindrical body shape. (a) Solid model(left) (b) Wire-frame model(right)

2.6 Algorithms for Moving the Graphical Arm

A robot is controlled in the joint space, whereas most of the tasks are done in the Cartesian space. The robot end effector is represented by position vector and orientation matrix in 3D space. Therefore, a point in the joint space can be converted to a position and an orientation in the Cartesian space. Transformation functions relate the position and orientation of the end effector coordinate system to the base coordinate system. Therefore, we can either use the joint angles as our desired position or we can issue a movement command in the cartesian space coordinates. More specifically, we can either give an increment in angular position of the robot or we can issue an incremental command in Cartesian space to move the robot from current position to the desired one. Algorithms for moving the real robot that we have used is described in [8]. Here we will describe the algorithms that govern the movement of graphical robot arm in the joint and Cartesian space.

2.6.1 Movement in the Joint Space

Whenever a command is received to move incrementally ($\Delta\theta$) in the joint space new value of the angles are calculated by adding the increment to the current values($\theta_i = \theta_i + \Delta\theta$). Then as shown in Figure 8, after checking whether the new value is within specified bound, new position and orientation of the joint point of the selected link and the links above that are calculated using the updated angle values sequentially. The computation are performed according the Direct Geometric equations based on BASE_FRAME of reference. where the value of the orientation matrix of one

link depends on the value of orientation matrices of some previous links.

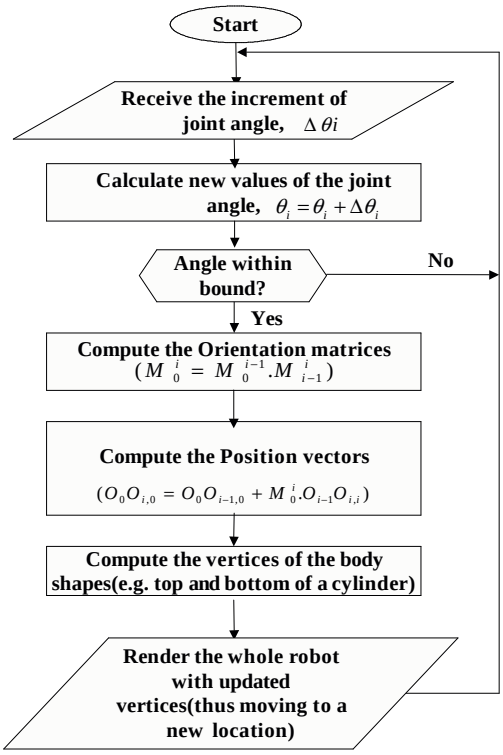

Figure 8. Movement of Graphical Robot in the joint space

Once the position and orientation of the end points or skeleton points are calculated then we need to re-calculate the top and bottom vertices of the body shape(i.e. cylinders) as discussed in Section 2.3. Then the whole robot is rendered using the updated vertices. Thus robot is moved to a new location with expected position and orientation.

If the absolute values of joint space variables i.e., $\theta_{required}$ (instead of incremental values) are supplied to move to a specified position then those angle values are directly used for subsequent computation and re-draw the robot as previous.

2.6.2 Movement in the Cartesian Space

To move the graphical robot in the Cartesian space we need to take two parameters as input: ΔX and ΔM, where ΔX holds the increments in position vector and ΔM is the change in the orientation matrix of the slave arm. At any time t, we need to hold a copy of current position vector $X(t)$, a (3×1) vector, and current orientation matrix $M(t)$, a (3×3) matrix. The new position vector $X_{new}(t)$ and orientation matrix $M_{new}(t)$ are to be calculated from $\{X(t), M(t)\}$ and $\{\Delta X, \Delta M\}$ taking into consideration the current frame of reference. Current frame of reference can be BASE_FRAME,WRIST_FRAME or TOOL_FRAME.

Once $X_{new}(t)$ and $M_{new}(t)$ are calculated, we can use the Inverse Kinematic Model $G^{-1}(X_{new}, M_{new})$ of the PUMA robot to find the joint space variables θ_{new}. This new angle is used to compute position and orientation matrices using Direct Kinematic Model. We also need to recompute the vertices of the body shapes(i.e. cylinders) of each link. Then the whole graphical robot is rendered with the updated vertices. The algorithm for moving the graphical robot arm in the Cartesian space.

To move the graphical robot to a specified position by supplying the absolute values of a certain position vector and orientation matrix i.e., $\{X_{required}, M_{required}\}$, new angular position is calculated from the supplied values using the Inverse Kinematic Model and the same procedure is used to re-draw the robot in the specified position.

2.7 Acquisition of Real Video Image and 3D Stereo-visualization

We have adopted the same Client-Server framework for image acquisition and network streaming as used in [8, 12]. Real video image of the slave robot at the server side is captured simultaneously by two video cameras. Then a reliable client-server connection is established and upon a request from the client a stereo frame comprising of two pictures is sent over LAN through window sockets. A double buffer, concurrent transfer approach is used to maximize overlapped transfer activities between cameras, processor and the network. On the client side after detecting and making connection with the server pictures are received. Page flipping technique is used for 3D visualization and HMD is used as display device that gives the effect of 3D depth perception. 3D view is also created with sync-doubling technique to view on the monitor specially when interacting with keyboard or mouse.

2.8 Virtual Image Superimposition

As mentioned in Section 1, for proper superimposition of graphics on the real video we need to we need to combine all local coordinate systems centered on the devices and the objects in the scene in a global coordinate system . For this, camera calibration and registration are performed. Although manufacturers provide a partial set of intrinsic camera parameters, we need to find them as they are not accurate enough. Besides, some of these parameters may vary from time to time, while some of them may be calibrated once for all, depending on the stability of the mechanical

and optical construction of the camera. Also in many situations the source of the images is not known, which means that the camera's internal parameters are also not known. In some cases, it is desirable to change a camera midway through an image application which also require that the internal parameters of the camera can only be extracted from the images themselves [13]. We also need to find the position and orientation of the camera(s) i.e. the extrinsic camera parameters.

We have used Heikkilas calibration [6] method implemented in MATLAB Camera Calibration tool box to find out the intrinsic and extrinsic camera parameters. It shows accuracy up to 1/50 of the pixel size. Pinhole camera model with perspective projection and least-square error optimization is used. The intrinsic parameters that can be found by this tool are: focal length (fc), principal point(cc), skew co-efficient (alpha_c) and distortions co-efficient(kc).

MATLAB Calibration Toolbox stores the focal length in pixels in a 2×1 vector **fc** whose elements **fc(1)** and **fc(2)** are the focal distance (a unique value in mm) expressed in units of horizontal and vertical pixels. The aspect ratio (τ) is the ratio of the pixel height and width and generally calculated according to Eq. 7.It can also be defined as the view space width divided by height. It is sometimes different from 1 if the pixel in the CCD array are not square.

$$\tau = \textbf{fc(2)}/\textbf{fc(1)} \tag{7}$$

If XX=[X,Y,Z] is the co-ordinate of P in grid, then the co-ordinate of P in camera reference frame can be expressed as

$$XX_c = R_c \times XX + T_c \tag{8}$$

where, translation vector T_c is the co-ordinate vector of O in camera reference frame and the rotation matrix R_c is the surface normal vector of the grid plane in the camera reference frame. External camera parameters are expressed in terms of these two vectors.

We like to use perspective projection. So we need to specify the amount of perspective, or "zoom", that the resultant image will have. This is done by specifying the angle of the viewing cone, also known as the viewing frustum. The viewing frustum is a rectangular cone in three-space that has the from-point as its tip, and that encloses the projection rectangle, which is perpendicular to the cone axis. The angle between opposite sides of the viewing frustum is called the viewing angle.

2.9 3D Visualization with Virtual Object Superimposed

During each flipping operation a complete stereo image is sent down to the HMD. This image is acquired from the network video stream and copied in a surface, say frontSurf while the drawing of the current image on graphics screen is in progress in the another surface, say augSurf. AugSurf is copied to the primary surface used to display(say, backSurf). Left and right camera image is displayed on two different view-ports on the monitors. Thus the stereo video is updated on local display in a page-by-page format and not pixel-by-pixel. This provides a great benefit in terms of reducing time delays.

2.10 GUI Design

Graphical User Interface (GUI)s are designed for both server and client end. The server side GUI forms described in [8] provide user the facility of connection and initialization of the real robot.

The client side GUI is designed to give the user options for connecting to the server PC, master arm and HMD and receiving and displaying the stereo video. It also takes user's input for movement of the real and graphical arm for task simulation.

2.11 Graphical Tele-manipulation

In teleoperation,if the base link of the real robot remains fixed relative to the video cameras, the base link of the graphical arm will also remain fixed relative to the graphical cameras. The end-effector of the graphical arm then can be manipulated in the graphical coordinate space, relative to objects in the task space (keeping base link in same location of real robot base).

The flexible algorithms used for movement of real and graphical robot arm in joint and cartesian space by a fixed or incremental value, lead to allow us using any of the input devices such as master arm, joystick, keypad, mouse etc.

For graphical tele-manipulation, at first we need to connect to the Real Robot Server (running on the server PC connected with PUMA-560 slave arm). Then the Real Robot arm is initialized with specified or pre-defined convenient position and orientation. The camera calibration parameters (intrinsic and extrinsic) computed earlier using MATLAB Camera Calibration Toolbox are then loaded to the program. Here we also need to specify the graphics parameters such as viewing model (solid or wire-frame), display device (HMD or monitors) etc. If Master Arm is chosen as a tool for user interaction then we need to start the process of engaging it i.e. communicating with the server system. Then we need to be connected with the Vision Server running on the Server PC to capture video images through camera. If HMD is selected as displaying device for 3D stereo view we need to connect it with the client system, otherwise monitor can be used to display the stereo

using Sync-Doubling techniques (displaying left and right images up and down on the monitor screen).

Once we have real and virtual image ready to be displayed, we need to perfectly superimpose the graphics onto the real video image by adjusting the camera calibration parameters. Then for task simulation e.g. pick and place operation we need to move the Graphics Robot first using the algorithms described in Section 2.6. If the movement is satisfactory it will be saved to be used while issuing command to the Real Robot. At the end of simulation before exiting the program we need to disconnect the client system from the Robot Server and the Vision Server running on the Server PC.

3 Implementation

The proposed design scheme is implemented on a Microsoft .NET framework using MS Visual C.NET, MS Visual C++ and MS DirectX. Among the available alternatives of 3D graphics API such as windows GDI, Direct3D, Java3D and OpenGL, we have chosen Direct3D considering its capability of hardware acceleration through it Hardware Abstraction Layer [see Figure 9]. It also allows running application in full-screen and it perfectly macth with other MS products used.

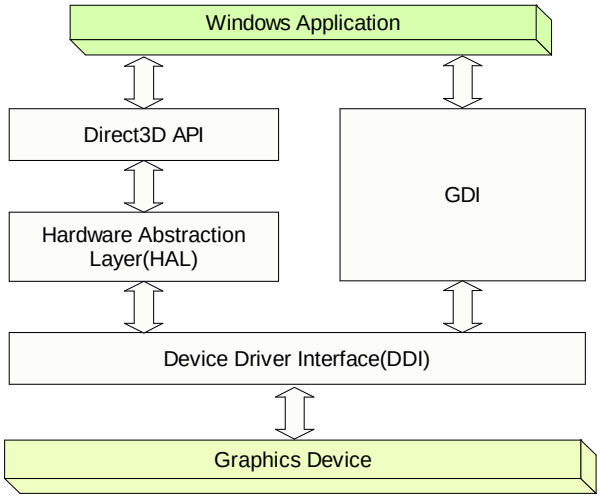

Figure 9. Dirct3D interaction to Hardware

The client system consists of six modules: robot client, vision client, camera calibration, virtual object modeling and DirectX interface. The first two are implemented to take visual and force feedback. Camera calibration module takes input from vision client to extract calibration parameters. The virtual object modeling module defines the structure of the virtual objects(graphical arm and other objects etc) and handles the functions like movement of virtual objects.

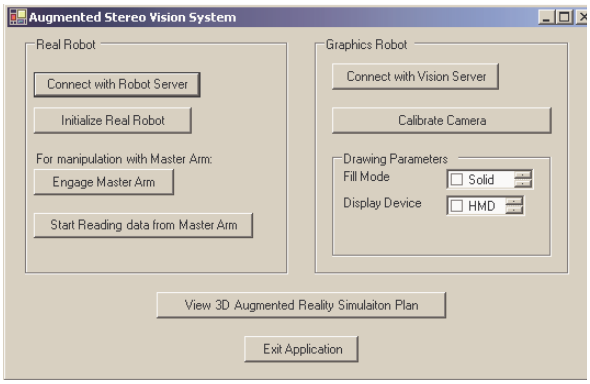

Figure 10. Main User Interface Form at Client Side

DirectX Interface module is the core module of the client application for displaying the augmented view. It takes input from virtual object modeling module, camera calibration module video client and output the augmented video. Thus it synchronies real and virtual data, make projection on video surface and performs the page flipping for the use with HMD.

Two GUI forms are designed in the client side to provide users interacting with the system. On the main GUI form [see Figure 10] buttons are attached for connecting with vision and robot server running on the server PC connected with the real robot. It also takes user's option for various graphics parameters. The stereo form GUI [see Figure 11 can be accessed by pressing the "View 3D Augmented Reality Simulation" button. Stereo form shows the left and right image to two separate monitors (or on the two eye sides of the HMD). It also displays the control features to change the joint angles of real and graphic robot arm, changing Cartesian position and opening or closing of gripper which may be used for task simulation.

4 Performance Evaluation

The performance of the proposed system against the rendering time and accuracy.Data was taken by averaging over 1000 samples and running both server and client systems on PCs having 2-GHz Intel P4 processors with 1GB DRAM and 512 KB cache memory and connected to a campus network by using a 100 Mbps NIC card. The server PC is interfaced to two Sony Handycam digital cameras using a 400 mbps FireWire PCI (IEEE-1394) card.

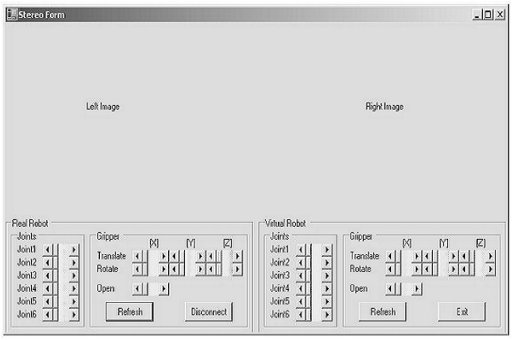

Figure 11. User interface form in stereo view at client side

4.1 Speed of Rendering Graphics

We have computed the refresh rate of the output screen displaying the graphics in term of frame per second. We have tested the system for different environments- with or without overlaying graphics over the real video. For each environment we have recorded the refresh rate without any drawing, with a cuboidal object and the robot arm, only robot arm with 8 segments per cylinder and only robot arm with 50 segments per cylinder. As shown in Table 1, refresh rate is proportional to the complexity of the drawings and image acquisition time.

Environment	**Graphics Complexities**	**Avg. Refresh Rate (fps)**
Only graphics	No drawing	273.36
	With an object in the scene	243.74
	Only robot with 8 segments in each cylinder	253.59
	Only robot with 50 segments in each cylinder(solid view)	239.78
Graphics on video	Without any drawing	11.498
	With an object in the scene	11.384
	Only robot with 8 segments in each cylinder	11.347
	Only robot with 50 segments in each cylinder (solid view)	11.325

Table 1. Refresh rate of the output screen.

Time required to render the whole robot with different numbers of triangles in a cylindrical shape and different views is shwon in Table 2. It is observed that there is not significant difference in performance for using solid or wire-frame model.

Environment	Graphics Complexity	Rendering Time (ms)	
		Wire-frame	Solid
Only graphics	Only robot with 8 segments in each cylinder	64.088	64.283
	Only robot with 50 segments in each cylinder	65.034	65.151
Graphics on video	Only robot with 8 segments in each cylinder	105.02	108.362
	Only robot with 50 segments in each cylinder	105.37	109.037

Table 2. Time required to render the graphical robot.

4.2 Accuracy of the System

The system is evaluated in terms of the accuracy of the calibration method and the accuracy in movement. To evaluate the accuracy of our calibration method we have re-computed the projection of grid points using our calculated calibration parameters and re-projected onto the original grid image. The errors are analyzed using MATLAB calibration Toolbox which gives result of re-projection error as the standard deviation of the re-projection error (in pixel) in both x and y directions respectively. The errors can be even minimized (p to 1/50 of a pixel size) by identifying the location of the pixels that create larger errors and re-defining the window size.

The accuracy of the movements of the graphical object on the output screen mainly depends on the accuracy of the calibration. But it also depends on the accuracy of the computation of the graphics parameters. We have compared the position of the graphic robot arm before and after the movement and compared the difference with the real data provided by the user through user interface.

5 Comparing Our Approach to Others

Iqbal, A. [8] augmented with only a small red ball at the position of gripper in comparison to our whole graphical arm. [8] used Faugeras [14] calibration with Kuno et. al. [15]s affine frame of reference which led him to noticeable mismatch with real error in the matching shown in his figure. Whereas our computer vision-based calibration reduces error up to 1/50 of pixel size. J. Vallino [16] reports in his PhD thesis refresh rate of 10 fps to be required for AR, when we get above 11-17 fps after overlaying graphical arm with live stereo video. Graphics rendering of our system is faster than [8, 2, 7] for our use of Direct3D. Our system is comparatively cheaper due to the use of commodity hardware (PC) and software.

6 Conclusion

A hierarchical design strategy for augmenting a telerobotic stereo vision system is described in this paper. It is shown that by using hardware accelerated graphics rendition through Direct3D provides excellent refresh rate of the output screen. It also implements a computer vision based calibrations method giving accuracy up to 1/50 of pixel size. A flexible and generalized data structure is proposed which is suitable for telerobotic visualization. A user-friendly graphical user interface is developed for simple manipulation in the telerobotic AR system. It can be used as a base framework for further research on virtual and augmented telerobotic manipulations. As our future research we like to provide an intelligent system to switch between VR and AR modes of operation based on network delays to ensure QoS. We also plan to use commercially available software to extract exact 3D model of the workspace objects which will facilitate more accurate task manipulation.

7 Acknowledgement

Authors would like to acknowledge and give thanks to King Fahd University of Petroleum and Minerals for her continuous support in the research activities.

References

[1] R. Azuma, Y. Baillot, S. Behringer, R.and Feiner, S. Julier, and B. MacIntyre. A survey of augmented reality. *Computer Graphics and Applications, IEEE*, 21(6):34–47, Nov.-Dec. 2001.

[2] A Rastogi. Design of an interface for teleoperation in unstructured environments using augmented reality. *M.A.Sc. Thesis, Department of Industrial Engineering, University of Toronto, Toronto*, 1996.

[3] J. Abdullah and K. Martinez. Camera self-calibration for the artoolkit. *The First IEEE International Workshop on Augmented Reality Toolkit*, page 5, Sept. 2002.

[4] J.-Y. Herve, C. Duchesne, and V. Pradines. Dynamic registration for augmented reality in telerobotics applications. *IEEE International Conference on Systems, Man, and Cyberneticsn*, Vol.2:1348 – 1353, October 2000.

[5] S.R. Gomez, J. Carro, E. Moreda, J.J. Escribano, and C. Cerrada. A design srategy for low cost experimental telerobotics platforms. *Proc. IEEE International Workshop on Robot and Human Interaction, Pisa, Italy*, pages 273 – 278, Sept. 1999.

[6] J. Heikkila. Geometric camera calibration using circular control points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.22, No. 10:1066 – 1077, Oct. 2000.

[7] R. Marin, P.J. Sanz, and J.S. Sanchez. A very high level interface to teleoperate a robot via web including augmented reality. *Proc. IEEE International Conference on Robotics and Automation, 2002 ICRA '02*, Vol.3:2725 – 2730, May 2002.

[8] A. Iqbal. Multistream realtime control of a distributed telerobotic system. *M.Sc. Thesis, King Fahd University of Petroeum and Minerals*, June 2003.

[9] Mayez Al-Mouhamed, Onur Toker, and Nesar Merah. Design of an intelligent telerobotic system. *Technical Report AT-20-80, King Abdulaziz City For Science and Technology, Kingdom of Saudi Arabia*, 2004.

[10] R.A. Browse and S. Little. The effectiveness of real-time graphic simulation in telerobotics. *Proc. of IEEE International Conference of Systems, Man, and Cybernetics, Charlottesville, Virginia*, pages 895–898, Oct 1991.

[11] G. Litern, K. E. Thomley-Yates, B.E. Nelson, and S.N. Roscoe. Content, variety, and augmentation of simulated visual scenes for teaching air-to-ground attach. *Human Factors*, Vol. 29(1):45–59, 1987.

[12] M. Al-Mouhamed, Onur Toker, and Asif Iqbal. A multi-threaded distributed telerobotic framework. *the IEEE/ASME Transactions on Mechatronics(accepted)*.

[13] A. Henrichsen. 3d reconstruction and camera calibration from 2d images. *M.Sc. Thesis, University of Cape Town*, December 2000.

[14] O.D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? *In Proc. 2nd European Conference on Computer Vision, LNCS 588, Springer-Verlag*, pages 563–578, 1992.

[15] 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.

[16] J. Vallino. Interactive augmented reality. *PhD thesis, Department of Computer Science, University of Rochester, Rochester, NY*, April 1998.