A 3D Vision-Based Man-Machine interface For Telerobotics

M. Al-Mouhamed [*], O. Toker [†], and A. Al-Harthy [‡]

Abstract

This paper presents a robust telerobotic system that consists of a real-time vision-based operator hand tracking system (client) and a slave robot (server) which are interconnected by using a LAN. The tracking system (1) monitors the operator hand motion and (2) determine its position and orientation which are used to control the slave robot. Two digital cameras are used to monitor a four-ball called feature frame that is held by the operator hand. To determine the 3D position a tracking algorithm uses uncalibrated cameras together with the affine invariant property. This allows finding 3D differential position and orientation of operator hand. The features of 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. The operator can see the effects of the previous motion which enables making the necessary corrections through repetitive operator hand-eye interactions. In the evaluation we study the static and dynamic errors of the tracking system as well as combined errors due to the affine invariant transformation. We also present the telerobotic real-time control scheme and its network and processing delays.

1 Introduction

This paper presents a telerobotic system [1] that consists of a vision-based station (client) and a slave robot (server) which are interconnected through a 100 Mbps Ethernet 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. Using uncalibrated 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.

Fukumoto et al. [2] 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. 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. 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

[*]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/onur@ccse.kfupm.edu.sa.

[†]Department of Systems Engineering, CCSE, KFUPM, Dhahran 31261, Saudi Arabia.

[‡]Department of Systems Engineering, CCSE, KFUPM, Dhahran 31261, Saudi Arabia.

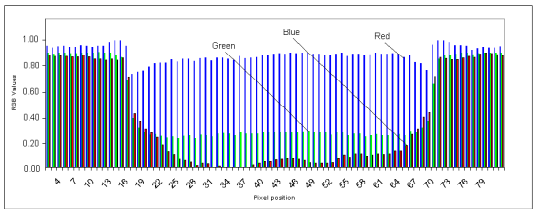

Figure 1: The RGB for a scan of a red ball with white background

of the object. Using one single camera Lloyd [3] 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 [4] 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. Uncalibrated stereo vision [5] 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 [6] 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.

Kuno et al. [7] proposed an interfacing method by using uncalibrated 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.

Our system recognizes the position and orientation of the feature frame regardless of the position of the operator. 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. 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 a robust boundary detection, fast tracking strategy, and a simple mechanism for partial occlusion. The gravity center of each ball should be computed precisely. 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 paper is organized as follows. Section 2 presents a metric to measure color matching. Section 3 presents our the tracking algorithm. Section 4 presents the 3D position matching module. Section 5 presents the evaluation. Section 6 concludes about this work.

2 A metric to measure color matching

In practice the colors (RGB) 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 1. 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.

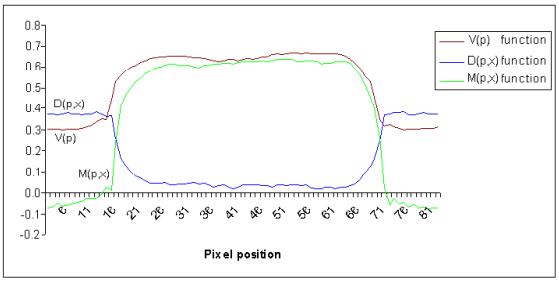

Figure 2: Metric functions for a scan of red ball with white background

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 lunimance 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 horizontal scan of a red ball with white background shown in Figure 1 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.

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 normalized color luminance function $V_i(p)$ as follows:

$$V_i(p) = \frac{1}{3} \begin{cases} c_1 + (1 - c_2) + (1 - c_3) & \text{for the primary red color } (i = 1) \\ (1 - c_1) + c_2 + (1 - c_3) & \text{for the primary green color } (i = 2) \\ (1 - c_1) + (1 - c_2) + c_3 & \text{for the primary blue color } (i = 3) \\ c_1 + c_2 + (1 - c_3) & \text{for the red-green } (i = 4) \\ c_1 + c_3 + (1 - c_2) & \text{for the red-blue } (i = 4) \\ c_2 + c_3 + (1 - c_1) & \text{for the green-blue } (i = 4) \end{cases}$$

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 ($i = 4$).

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.

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 there is a good preservation of relative composition of the RGB components as compared to the case of the white backgournd. 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 $D(p, p^i_{ref})$ between the a reference color $p^i_{ref} = (c^i_1, c^i_2, c^i_3)$ and a given color pixel $p = (c_1, c_2, c_3)$. The normalized distance color matching is defined as:

$$D(p, p^i_{ref}) = \left(\frac{1}{3} \sum_{j=1}^{3} (c_j - c^i_j)^2 \right)^{1/2}$$

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 dicrimination of ball colors from a wide spectrum of realistic background colors. One approach to preserve the benefit of the distance matching $D(p, p^i_{ref})$ and the color luminance function $V_i(p)$ is to combine them into one single color matching function $M(p, p^i_{ref})$ 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 2 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 $D(p, x)$.

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 paper as the main technique for color detection and matching.

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 3.1) 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 3.3). For this each camera tracks the feature frame, shown in Figure 3-(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 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 3.4) among the balls and determine reasonable solution for

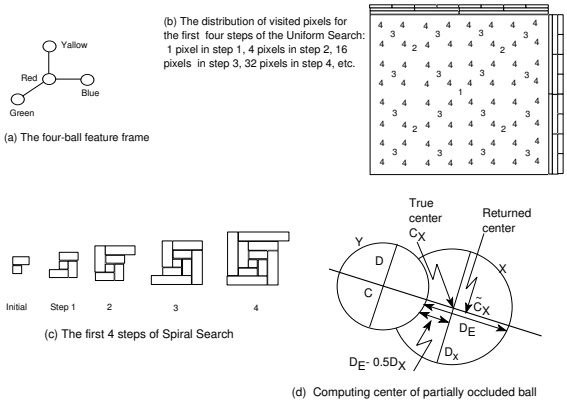

Figure 3: Feature frame, Uniform and Spiral search, and the case of occluding

each case. To consolidate the detection our algorithm validates the detected balls through shape and geometric matching 3.5 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 4 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.

3.1 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 RGB parameters for each color are determined by using the histogramming 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.

3.2 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:

$$M(p, p^i_{ref}) \geq M(p_{bkg}, p^i_{ref}) + \alpha \times (M(p_{typ-i}, p^i_{ref}) - M(p_{bkg}, p^i_{ref})) \quad (1)$$

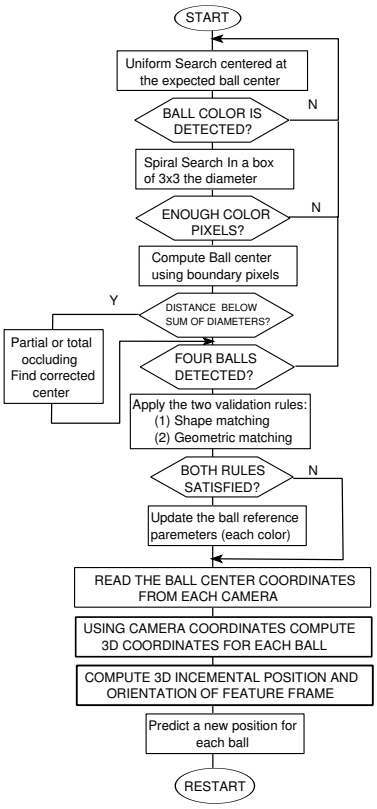

Figure 4: Flow chart of the tracking algorithm and its major functions.

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 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 1 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 3-(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.

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

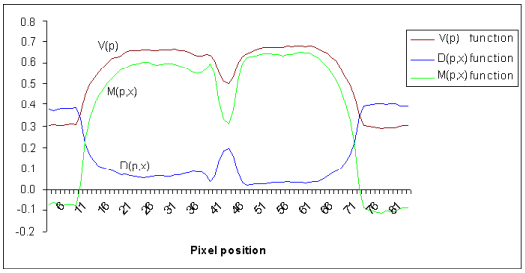

Figure 5: The luminance, distance, and matching functions for red ball a reflection area

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.

3.3 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 5. 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:

$$(i_r, j_r) = \left(i, \frac{j_{max} \times (j_{max}+1) - j_{min} \times (j_{min}+1)}{2(j_{max} - j_{min} + 1)} \right) \tag{2}$$

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.

3.4 Partial and total occluding

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.

As shown in Figure 3-(d), given two Balls X and Y, the technique of computing the coordinate of the ball center by using Equation 2 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:

$$C_X = C + \frac{\tilde{C}_X - C}{|\tilde{C}_X - C|}(D_E + 0.5(D - D_X)) \tag{3}$$

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.

3.5 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 [8] 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 orthonormal 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.

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 uncalibrated 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. The cameras can be set in an arbitrary position or even move to keep centering on the observed feature frame.

We use multiple views generated by two uncalibrated 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 orthonomal 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:

$$E_i = \frac{X_i - X_0}{\|X_i - X_0\|^{\frac{1}{2}}} \tag{4}$$

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:

$$X(t + 1) = X_0(t) + \sum_{i=1}^{3} \alpha_i \times E_i(t) \tag{5}$$

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:

$$\begin{pmatrix} x^1 \\ y^1 \end{pmatrix} = \begin{pmatrix} x_0^1 \\ y_0^1 \end{pmatrix} + \begin{pmatrix} e_{1,1}^1(t) & e_{2,1}^1(t) & e_{3,1}^1(t) \\ e_{1,2}^1(t) & e_{2,2}^1(t) & e_{3,2}^1(t) \end{pmatrix} . \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \tag{6}$$

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 (superscipt) 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:

$$\begin{pmatrix} x_0^1 \\ y_0^1 \\ x_0^2 \\ y_0^2 \end{pmatrix} = \begin{pmatrix} e_{1,1}^1(t) & e_{2,1}^1(t) & e_{3,1}^1(t) \\ e_{1,2}^1(t) & e_{2,2}^1(t) & e_{3,2}^1(t) \\ e_{1,1}^2(t) & e_{2,1}^2(t) & e_{3,1}^2(t) \\ e_{1,2}^2(t) & e_{2,2}^2(t) & e_{3,2}^2(t) \end{pmatrix} . \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \tag{7}$$

The least squares solution $\hat{\alpha}$ is given by:

$$\hat{\alpha} = (M^t M)^{-1} M^t X \tag{8}$$

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 8 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 4. 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.

5 Performance evaluation

The vision system configuration used for the experiments consists of two connected PCs, each has a Firewire card interfaced 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

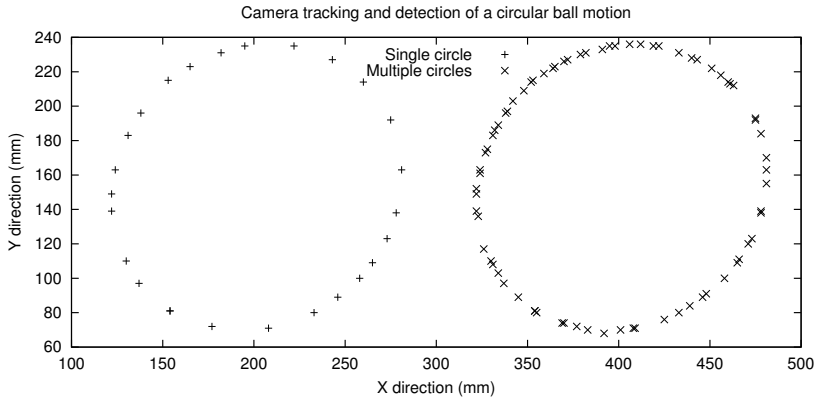

Figure 6: Tracing of the detected motion of a red ball held by a moving robot

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 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 as shown on Figure 6. 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

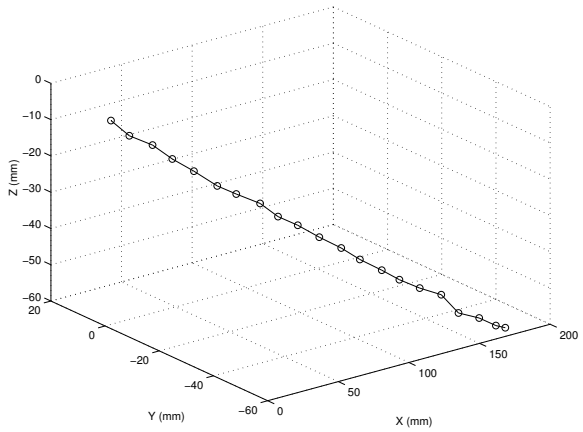

Figure 7: Two cameras tracking of a linear 3D trajectory with single-pass and lines

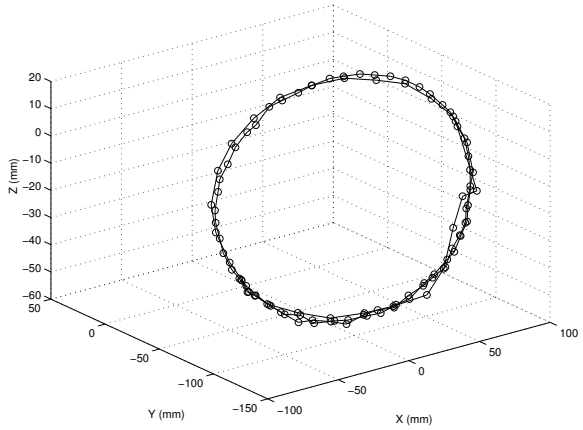

Figure 8: Two cameras tracking of a circular 3D trajectory with multi-passes and lines

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 7 and 8. 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 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

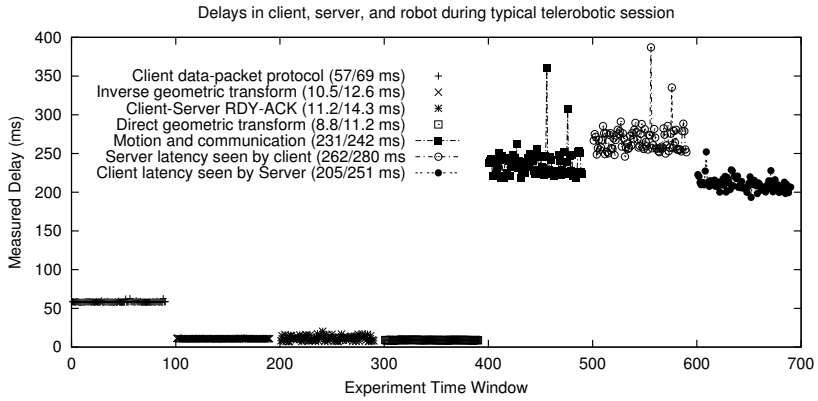

Figure 9: Delays in Client, Server, and PUMA with slow motion (0.5 mps)

in the slave robot moves the arm from its previous position $\theta_{Puma}(t)$ to the new equilibrium position $\theta_{Puma}(t+1)$.

Figure 9 shows the timing of the various components of the telerobotic system during a motion controlled by the operator. The dominant parts (about 0.2 seconds) are due to motion of the robot system and the processing at the client computer. The peaks shown are due to larger incremental motion.

6 Conclusion

In this paper we presented a telerobotic system that consists of a real-time vision-based tracking algorithm (client) and a slave robot (server) which are interconnected 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. 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.

References

[1] 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.

[2] 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.

[3] J. E. Lloyd, J. S. Beis, K. D. Pai, and D. J. Lowe. Model-based telerobotics with vision. *Proc. IEEE Inter. Conf. on Robotics and Auatomation*, pages 1297–1304, 1997.

[4] N. J. Hollinghurst and R. Cipolla. Uncalibrated stereo hand-eye coordination. *Image and Vision Computing*, 12:187–192, 1994.

[5] 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.

[6] S. Sato and S. Sakane. A human-robot interface using an interactive hand pointer that projects a mark in the real work space. *Proc. IEEE Inter. Conf. on Robotics and Automation*, 1:589–595, 2000.

[7] Y. Kuno, K. Hayashi, K.H. Jo, and Y. Shirai. Human-robot interface using uncalibrated stereo vision. *IEEE Inter. Conf. on Intelligent Robots and Systems*, 1:525–530, 1995.

[8] Mayez Al-Mouhamed. A robust gross-to-fine pattern recognition system. *IEEE Trans. on Industrial Electronics*, 48, No. 6:1226–1237, Dec. 2001.