A 3D Vision-Based Man-Machine Interface For Hand-Controlled Telerobot

Mayez A. Al-Mouhamed [*], Onur 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 through a LAN. The tracking system (1) monitors the operator hand motion and (2) determines its position and orientation which are used to control the slave robot. Two digital cameras are used to monitor a four-ball based feature frame that is held by the operator hand. To determine the 3D position a tracking algorithm based on uncalibrated cameras with weak perspective projection model is used. This allows finding 3D differential position and orientation of operator hand. The features of proposed system are (1) a metric for color matching to discriminate the balls from their background, (2) a uniform and spiral search approach to speedup the detection, (3) tracking in the presence of partial occlusion, (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. Evaluation shows that the static and dynamic errors of tracking algorithm are 0.1% and 0.6% for a centered workspace of 20^3 inches^3 that is 40 to 60 inches away from cameras. Running the tracking algorithm on two PCs in parallel allowed (1) a parallel image grabbing delay of 60 ms, (2) a stereo matching delay of 50 ms, and (3) a global refresh rate of 9Hz.

Keywords: stereo vision; telerobotics; 3D tracking; 3D vision

1 Introduction

This paper presents a telerobotic system [1, 2, 3] 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 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

[*]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.

scene, the operator uses eyes shuttering glasses to see the robot scene in 3D from the client station. The generation of stereo views [4, 5] based on display of left and right images of the robot which produces the stereo illusion 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 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 for 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 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. [6] 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. The operator cannot expect how the system will work when he moves his body from the initial position.

Estimating 3D position of oject 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 uses 2D images of these features to solve for the 3D position of the object. Using one single camera Lloyd [7] 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 [8] 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 [9] 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 [10] 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. [11, 12] 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. 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 is 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 center of gravity of each ball should be computed precisely. The color information is useful to improve discrimination and to enable 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 paper 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 tracking algorithm. Section 5 presents the 3D position matching module. Section 6 presents the evaluation. Section 7 concludes this work.

2 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 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 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.

Besides the RGB color space, there are some other 3D color coding, like YUV, YCBCR, HIS, etc., but in general use of the above coding other than RGB in converting from one coding to another increases the processing delays. For a high-speed image application with real-time control, the extra computational overhead and delay may not be acceptable.

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

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 ortho normal 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 3), (2) a tracking algorithm running on two computers where each computer is interfaced to a digital camera (Section 4), (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 used to compute the 3D position of each ball which enables finding the position and orientation of the feature frame Section 5.

The tracking algorithm tracks the ball motion, carry out color edge detection 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

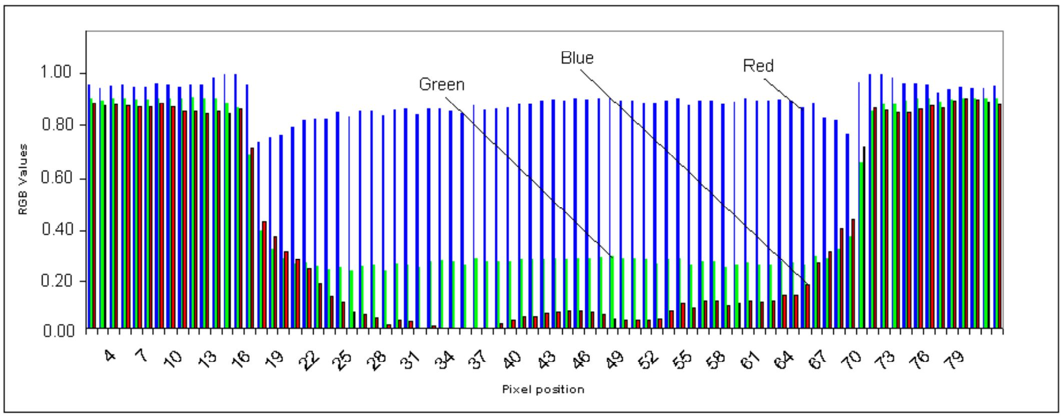


Figure 2: The RGB for a scan of a red ball with white background

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 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.

3 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 horizontal scan of a red ball with white background shown in Figure 2 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. 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 3.

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

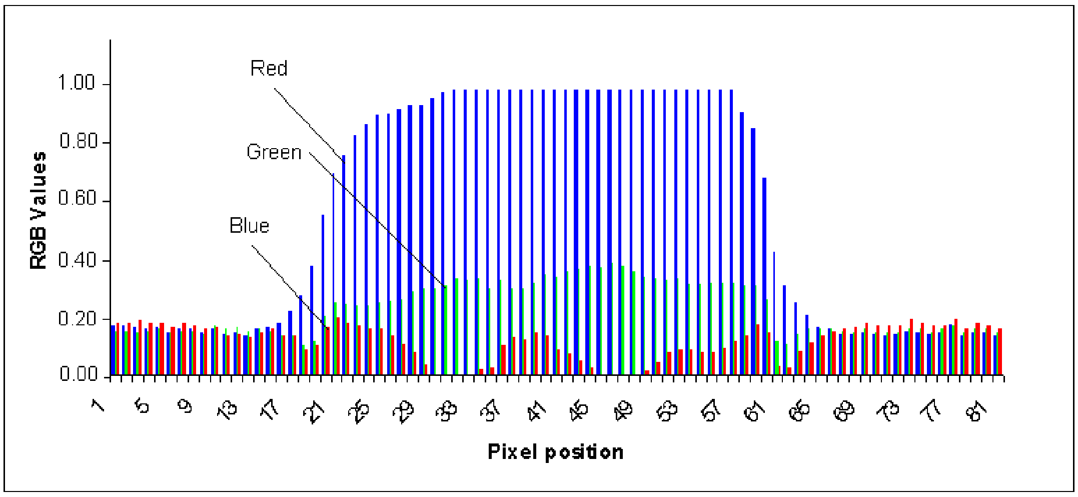


Figure 3: The RGB for a scan of a red ball with black background

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:

$$V_i(p) = \frac{1}{3}w(p) + \frac{1}{3}\begin{cases} 2c_i - 1 & \text{for } 1 \leq i \leq 3 \text{ (one color)} \\ \sum_{j=1, j\neq k}^{3}(2c_j - 1) & \text{for } 4 \leq i \leq 6 \text{ (two colors)} \end{cases} \tag{1}$$

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 Equation 1, for each ball of the feature frame we define a set of four colors for which $V_i(p)$ is given by the expressions:

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

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 maximum 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 [13] is $V_i(p) = 1 - \prod_{j=1}^{3}(c_j - c_j^i)^2$, where c_j

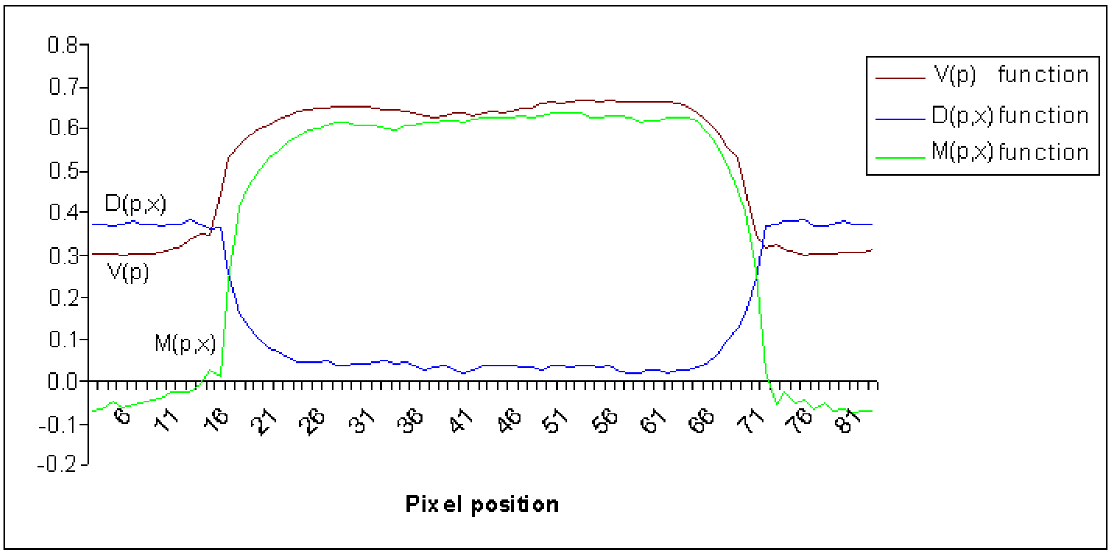


Figure 4: Metric functions for a scan of red ball with white background

and c_j^i are the jth measured and referenced components of color i, respectively. In this case $V_i(p)$ will be maximal if at least one measured component matches the corresponding reference color of the same 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 $V_1(p) = 1$. Our luminance function (Eq. 2) equally accounts for the accumulation of color matching over all the RGB components.

The function $V_i(p)$ measures 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 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 to those of the red ball. In the case of black background 3 there is a good preservation of relative composition of the RGB components as compared to the case of the white background 1. 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:

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

The distance depends on the feature parameters of a specific reference color which

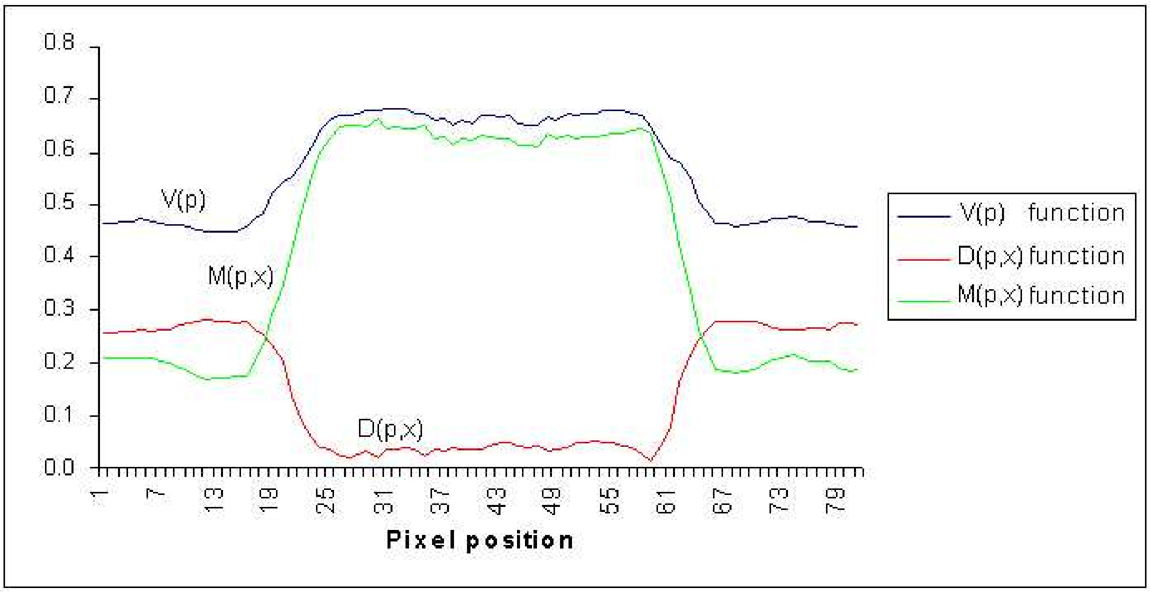


Figure 5: Metric functions for a scan of red ball with black background

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 benefits of both distance matching $D(p, p^i_{ref})$ and color luminance function $V_i(p)$, is to combine them into one single color matching function $M(p, p^i_{ref})$ defined by:

$$M(p, p^i_{ref}) = V_i(p) - D(p, p^i_{ref}) \tag{4}$$

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 pixel, respectively. Figures 4 and 5 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 using 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 paper as the main technique for color detection and matching.

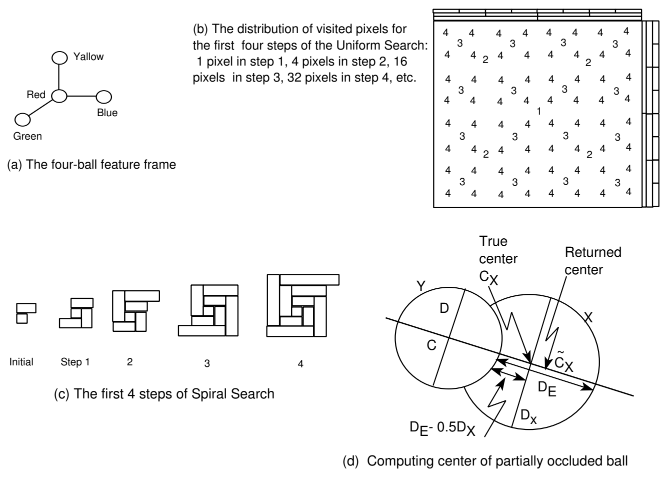

Yallow
Red
Blue
Green
(a) The four-ball feature frame
(b) The distribution of visited pixels for the first four steps of the Uniform Search: 1 pixel in step 1, 4 pixels in step 2, 16 pixels in step 3, 32 pixels in step 4, etc.
Initial
Step 1
2
3
4
(c) The first 4 steps of Spiral Search
True center C_X
Returned center
Y
D
C
X
$\tilde{C}_X$
D_E
D_X
D_E- 0.5D_X
(d) Computing center of partially occluded ball

Figure 6: Feature frame, Uniform and Spiral search, and the case of occlusion

4 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 4.1) 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 4.3). For this each camera tracks the feature frame, shown in Figure 6-(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 3) 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 4.4) among the balls and determines reasonable solution for each case. To consolidate the detection our algorithm validates the detected balls through shape and geometric matching 4.5 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 7 which describes the algorithm structure and its major functions. In this mode, 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.

4.1 Supervised learning of colors

The RGB components of a given color depend on the environment lighting and surrounding light reflection. Since the proposed approach is based on tracking a set of colored balls we need to initialize the reference RGB for a given background. Supervised learning allows finding the typical RGB parameters for each color ball. For this the operator

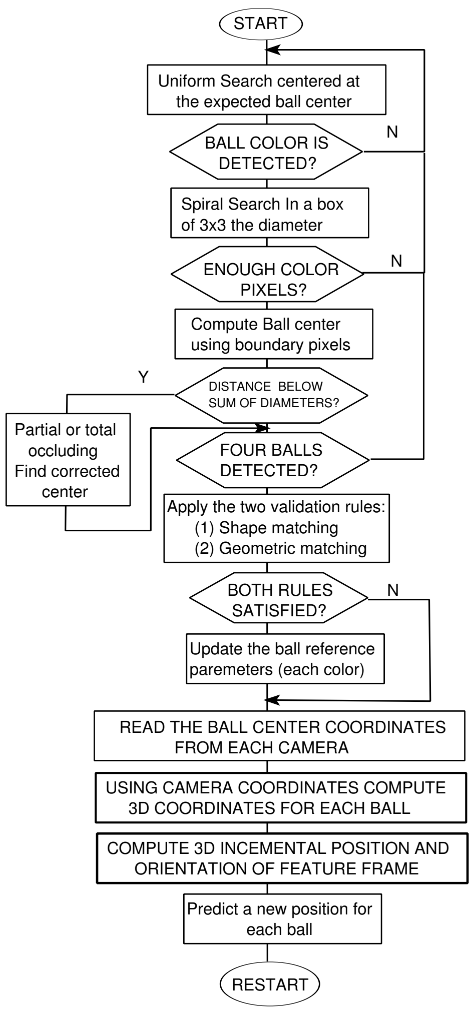

START
Uniform Search centered at the expected ball center
BALL COLOR IS DETECTED?
N
Spiral Search In a box of 3x3 the diameter
ENOUGH COLOR PIXELS?
N
Compute Ball center using boundary pixels
Y
DISTANCE BELOW SUM OF DIAMETERS?
Partial or total occluding Find corrected center
FOUR BALLS DETECTED?
Apply the two validation rules: (1) Shape matching (2) Geometric matching
BOTH RULES SATISFIED?
N
Update the ball reference paremeters (each color)
READ THE BALL CENTER COORDINATES FROM EACH CAMERA
USING CAMERA COORDINATES COMPUTE 3D COORDINATES FOR EACH BALL
COMPUTE 3D INCEMENTAL POSITION AND ORIENTATION OF FEATURE FRAME
Predict a new position for each ball
RESTART

Figure 7: Flow chart of the tracking algorithm and its major functions.

uses the mouse to point to each ball in the continuously refreshed computer image of the feature frame. The computer scans a small horizontal line centered at pointed location and having as many pixels as 3 times the ball diameter. The reference RGB parameters for each color are determined using its histogram which allows clustering 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.

4.2 Boundary detection and searching area limits

We noticed that a fast refresh 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 refresh 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 refresh 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:

$$M(p, p^i_{ref}) \geq M(p_{bkg}, p^i_{ref}) + \alpha \times (M(p_{typ-i}, p^i_{ref}) - M(p_{bkg}, p^i_{ref})) \tag{5}$$

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 $0 < \alpha \leq 1$. At initialization we may set $\alpha = 0.5$ but its values may be raised to filter similar color pixels when the number of detected color pixels is not enough (Figure 7) in the neighborhood of currently detected "ball". 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 5 is recomputed following each successful ball detection.

To minimize the search 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 6-(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 evidences 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 5 will be 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 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

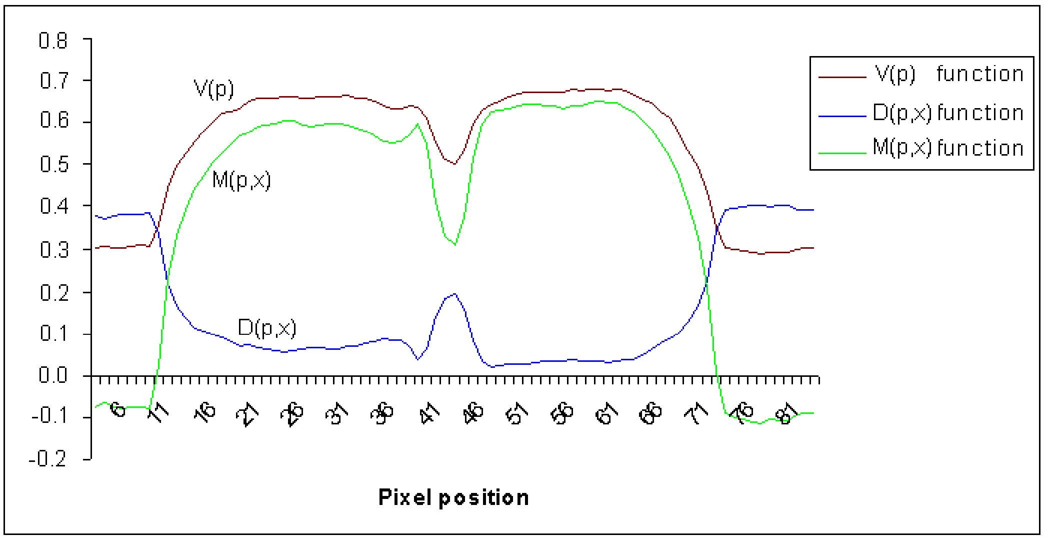


Figure 8: Luminance, distance, and matching functions for a red ball with a reflection area

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 partitioned 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 has visited a total of $\sum_{i=0}^{i=k} 4^i$ uniformly distributed pixels within the predicted search 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.

4.3 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 8. 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}) positions 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:

$$(\mathbf{i_r}, \mathbf{j_r}) = \left(\mathbf{i}, \frac{\mathbf{j_{max} \times (j_{max} + 1) - j_{min} \times (j_{min} + 1)}}{\mathbf{2(j_{max} - j_{min} + 1)}}\right) \tag{6}$$

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.

4.4 Partial and total occlusion

Partial and total occlusion may occur among the balls for each camera. The strategy is to monitor the distance between the ball centers and detect a partial occlusion situation which implies that the computation of ball center must be modified to account for the hidden part.

As shown in Figure 6-(d), given two Balls X and Y, the technique of computing the coordinate of the ball center by using Equation 6 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 occlusion situation. Otherwise, a ball is considered under partial occlusion 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 occlusion 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 occlusion 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:

$$C_X = C + \frac{\tilde{C_X} - C}{|\tilde{C_X} - C|}(D_E + 0.5(D - D_X)) \tag{7}$$

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.

4.5 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 two validation rules [14] for each camera frame which are: (1) the *shape matching*, and (2) the *geometric matching*. The confidence function provides a quantitative measure of the degree of matching of currently detected "balls" with those of the feature frame through matching of the circular shape and the relative positioning of the balls with respect to each other. In this way we avoid a false update of RGB references based on false detection.

The shape matching measures, for each camera, how circular are the balls that are detected without partial occlusion. 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:

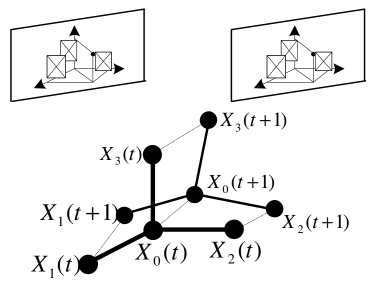
$$d(Measured, Predicted) = \sum_{k}^{3} \sum_{l}^{3} \frac{|d(x_k, y_l) - d(x_k^*, y_l^*)|}{Min\{d(x_k, y_l), d(x_k^*, y_l^*)\}} \tag{8}$$

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.

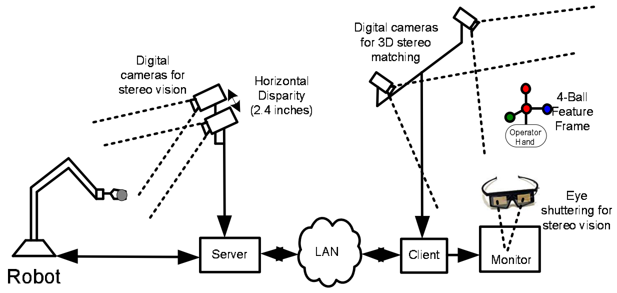
5 The 3D position matching module

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. This approach needs no camera calibration nor knowledge of camera position because the multiple view invariants are independent from the camera parameters. Therefore the cameras can be set in an arbitrary position or even moved to keep centered 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



a- Feature frame at times t and t+1 seen by camera 1 and camera 2 frame of references.


Digital cameras for 3D stereo matching
Digital cameras for stereo vision
Horizontal Disparity (2.4 inches)
4-Ball Feature Frame
Operator Hand
Eye shuttering for stereo vision
Robot
Server
LAN
Client
Monitor

b- Client including camera for 3D matching and eye shuttering as well as server including cameras for stereo vision.

Figure 9: Feature frame update and overall client-server system

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 6-(a). The algorithm gives the 3D position of the object with respect to the initial coordinate system composed of the basis vectors.

5.1 The affine invariants from multiple views

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:

$$E_i = \frac{X_i - X_0}{\|X_i - X_0\|^{\frac{1}{2}}} \tag{9}$$

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:

$$X(t+1) = X_0(t) + \sum_{i=1}^{3} \alpha_i \times E_i(t) \tag{10}$$

where α_i is being the ith components of a ball of frame $R(t + 1)$ that is observed with respect to $R(t)$ as shown on Figure 9 -(a). 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 coordinates of the same ball of $R(t + 1)$, with respect to $R(t)$, in the 2D frame of first camera are given by:

$$\begin{pmatrix} x^1 \\ y^1 \end{pmatrix} = \begin{pmatrix} x_0^1 \\ y_0^1 \end{pmatrix} + \begin{pmatrix} e_{1,1}^1(t) & e_{2,1}^1(t) & e_{3,1}^1(t) \\ e_{1,2}^1(t) & e_{2,2}^1(t) & e_{3,2}^1(t) \end{pmatrix} \cdot \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \tag{11}$$

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. Cameras frames are shown in Figure 9 -(a). Thus we have four equations with three unknowns. In matrix notation we have:

$$X = M.\alpha \tag{12}$$

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:

$$\begin{pmatrix} x_0^1 \\ y_0^1 \\ x_0^2 \\ y_0^2 \end{pmatrix} = \begin{pmatrix} e_{1,1}^1(t) & e_{2,1}^1(t) & e_{3,1}^1(t) \\ e_{1,2}^1(t) & e_{2,2}^1(t) & e_{3,2}^1(t) \\ e_{1,1}^2(t) & e_{2,1}^2(t) & e_{3,1}^2(t) \\ e_{1,2}^2(t) & e_{2,2}^2(t) & e_{3,2}^2(t) \end{pmatrix} \cdot \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \tag{13}$$

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

$$\hat{\alpha} = (M^t M)^{-1} M^t X \tag{14}$$

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 14 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 is derived from Equation 9. 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 position information, we can obtain 3D orientation matrix $R_{3\times3}$ information based on the 3D position information as following:

$$R_{33} = \begin{pmatrix} \frac{1}{l_1} & 0 & 0 \\ 0 & \frac{1}{l_2} & 0 \\ 0 & 0 & \frac{1}{l_3} \end{pmatrix} \cdot \begin{pmatrix} x_g - x_r & x_b - x_r & x_w - x_r \\ y_g - y_r & y_b - y_r & y_w - y_r \\ z_g - z_r & z_b - z_r & z_w - z_r \end{pmatrix} \tag{15}$$

In teleoperation a master arm is used to continuously measure increments in operator hand position $\Delta X_h = X_h^+ - X_h$ and hand frame orientation $\phi = M_h^{-1} M_h^+$, where (X_h, M_h) and (X_h^+, M_h^+) represent the previous and current operator hand frame position and orientation, respectively. In order for $\phi = M_h^{-1} M_h^+$ to produce correct results we must ensure that the column vectors of orientation matrix $M_h = [E_1, E_1, E_1]$ are normalized and orthogonal before computation of ϕ. For this we need to guarantee that the identified feature frame has normalized orthogonal vectors.

We used 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:

$$Y_i = X_i - (\frac{X_i.Y_1}{Y_1.Y_1}).Y_1 - (\frac{X_i.Y_2}{Y_2.Y_2}).Y_2 - ... - (\frac{X_i.Y_{i-1}}{Y_{i-1}.Y_{i-1}}).Y_{i-1} \tag{16}$$

Note that the set of vectors $T^* = \{Y_1, Y_2, ...Y_m\}$ is an orthogonal set. In Step 3, we Let

$$Z_i = \frac{Y_i}{\|Y_i\|}, 1 \leq i \leq m \tag{17}$$

Then $T = \{Z_1, Z_2, ..., Z_m\}$ is an orthonormal basis for subspace W.

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. Both client and server programs use TCP/IP sockets to provide the required real-time connectivity through the LAN. Figure 9-(b) presents the overall telerobotic system.

6.1 Real-time stereo vision using eye shuttering technology

At the server station, two digital cameras (left and right) are set on a common base to maintain parallel orientation with 6 cm as horizontal disparity. The IEEE 1394 Firewire adapter is used for camera interfacing. A video stream is created using TCP/IP socket programming for the real-time transport of the images from the server site to the client site. To preserve image resolution we used the LAN file sharing service.

The "above below" format maps the left and right images to the upper and lower halves of the display. The eye shuttering electronics capture the sync signal from the graphic card, multiplies it by two, and sends the modified signal to the display. This forces the display to draw first the upper image, then the lower image, and repeat this process in a loop. When the display is looked by naked eye, one can see two superimposed images. However, when looked through shuttering glasses, one can see a 3D image. In fact the shuttering glasses synchronize the opening of left (right) glass and the closure of right (left) glass at a rate of 150Hz. The synchronization uses embedded wireless infrared control. Each eye sees an image that is refreshed at a rate of 75Hz which is sufficient to eliminate possible flickering. This process gives the illusion of 3D vision to the human brain.

6.2 The weak perspective projection assumption

To study the errors due to weak perspective projection, we performed the following tests. First, we generated on the computer a test image for a grid consisting of 25 by 25 small squares for which the side of each square is 0.2 inches. We captured pictures of the above grid using our video camera from varying distances. In all cases, the video camera's optical zoom was always enabled. We carried out analysis of errors and distortion effects on the the resulting images for the following two cases.

In the first case, the camera orientation vector is set perpendicular to the grid plan, and passes through the grid center. We observed that, the square sides appear as curved instead of straight lines. For a distance $d = 10$ inches from the video camera to the grid the maximum distortion due to curved square side is about 2% of the side length. However, when $d = 20$ inches or more, the above magnitude of the above nonlinear effects becomes below the camera resolution.

In the second case, the grid plan is placed in one of the four image quadrants and the camera orientation vector was set to 45 degrees incidence angle with the grid plan. Similar to the first case, nonlinear effects were observed. Their error magnitude increases with a decrease in d. Indeed, it has been observed that the square sides appear slightly curved and lacking parallelism. For a distance $d = 10$ inches ($d = 40$ inches), the angular error in the orientation of square parallel sides does not exceed 10 degrees (5 degrees). However, in our operating range of $d = 60$ inches to 80 inches there were hardly noticeable or measurable angular deformation using the available camera resolution, i.e. an error less than 0.1% of the side length.

6.3 Static and dynamic testing

The proposed 3D vision-based man-machine interface is part of the client station that consists of two PCs connected by using a simple cross-over cable to minimize communication delays. We use 2 PCs to speedup execution of our tracking algorithm. Each PC is interfaced to a digital camera using Firewire card. The distance between cameras is 20 inches with near perpendicular camera orientations. Thus the operator hand workspace is a sphere of 40 inches diameter that is at least 40 inches distant from the cameras to meet the requirements of the weak perspective camera model. The feature frame held by the operator hand occupies of 2.4^3 inches^3 and occupies about 1/25th of the camera image.

The PCs synchronize acquisition of two images and computation of ball coordinates. However, one PC computes the 3D stereo matching after it receives the 2D coordinates of ball centers from the other PC. At the kth iteration, this gives the 3D incremental change in the coordinates of the four balls which enables computation of increments in operator hand frame origin (α_k), a 3×1 vector, and orientation matrix (Φ_k), a 3×3 matrix.

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 (0.5 inches) measured in pixels. The ball was set at a distance of 60 inches 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 40 inches leads to a static error of 0.01 inches with an upper bound of 0.025 inches.

The measurement of dynamic errors with respect to one camera consists of evaluating

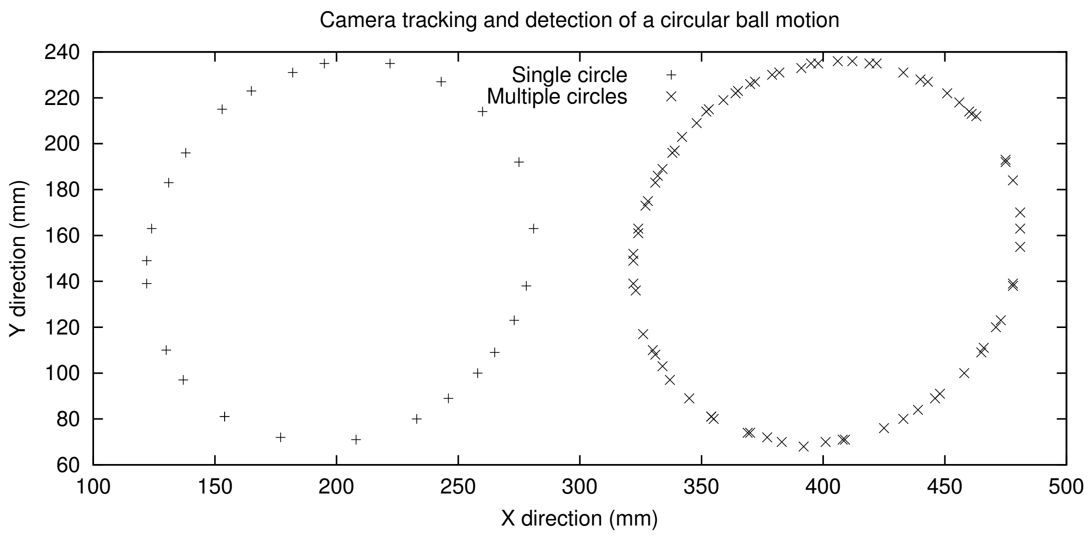


Figure 10: Tracing of the detected motion of a red ball held by a moving robot

the ball position errors in the camera frame while moving the feature frame by the robot along linear and circular trajectories as shown on Figure 10. The ball diameter was approximately 0.5 inches or 15 pixels. Our algorithm was implemented on 2 PCs, one for each camera, and dynamically tracking in parallel the balls. The camera fraplot.psme 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 a 10 Hz refreshing rate was achieved. 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 40 inches leads to a dynamic error of 0.12 inches with an upper bound of 0.2 inches. The above results are only valid when the ball speed is below 20 inches/s. The average dynamic errors increase significantly with increase in the ball speed. for example at a speed of 28 inches/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.

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 11 and 12. The setting is similar to the previous experiment. The upper bound on the measured error was a box of 0.24^3 inches^3 in a slave arm workspace of 40 inches^3 when the motion speed was at most 20 inches/s and the feature frame was about 40 inches away from the cameras.

6.4 Kinesthetic mapping between master and slave systems

The server station continuously reads the joint position (vector $\theta_{Puma}(t)$) of the PUMA 560 slave arm, and computes the slave arm tool position and orientation $\{X_k, M_k\} = G(\theta_{Puma}(k))$, where X_k and M_k are the slave frame position (3×1) and orientation matrix (3×3) and $G(.)$ is the direct Kinematic model of the slave arm. We use the Kinematic model to provide local control of singularities and multiple solutions. Whenever the slave station receives from the client station a position control packet with (α_k, Φ_k) it (1) com-

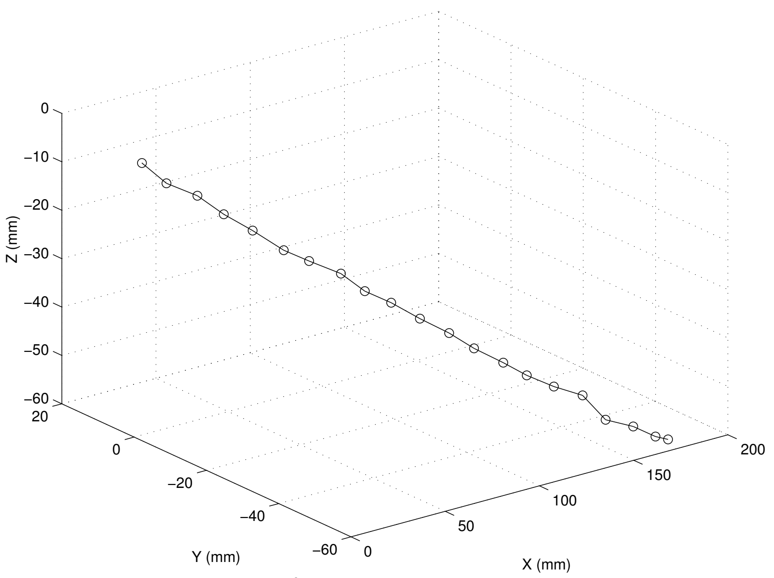


Figure 11: Two cameras tracking of a linear 3D trajectory with single-pass and lines

putes the new slave frame position $X_{k+1} = X_k + \alpha_k$ and orientation $M_{k+1} = M_k \Phi_k$, and (2) evaluates the slave joint position $\theta_{Puma}(k+1) = G^{-1}(X_{k+1}, M_{k+1})$ which corresponds to the desired tool frame, (3) sends to the slave robot the new joint position $\theta_{Puma}(k+1)$. The local servo-controller in the slave robot moves the arm from its previous position $\theta_{Puma}(k)$ to desired position $\theta_{Puma}(k+1)$. Thus the above scheme maps the operator incremental motion into increments on current slave tool position and orientation. 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.

Figure 13 shows the timing of the various components of the telerobotic system during a motion controlled by the operator. In this case the robot speed is fixed and the motion time varies depending on its magnitude.

The delay in the stereo matching algorithm has the following two components. The delays in grabbing of the data image from a camera and transfer to the PC DRAM is about 60 ms. The average stereo matching processing time is about 50 ms. Packet communication delay is about 1-5 ms between client and server. However, the dominant parts (about 0.2 seconds) are due to the motion of the electro-mechanical robot system. The peaks shown are due to larger incremental motion done at constant speed.

6.5 Comparison to others

In [8], a gesture interface is proposed for reliable tracking of a pointing hand using an uncalibrated stereo vision algorithm running on a Sun Sparc workstation. The stereo hand-eye coordination (closed-loop) converges by a straight-line path to its destination in 3 or 4 iterations. There are two sources of errors. First, large disturbances to camera geometry lead curved trajectories. Second, strong perspective is tolerated but requires

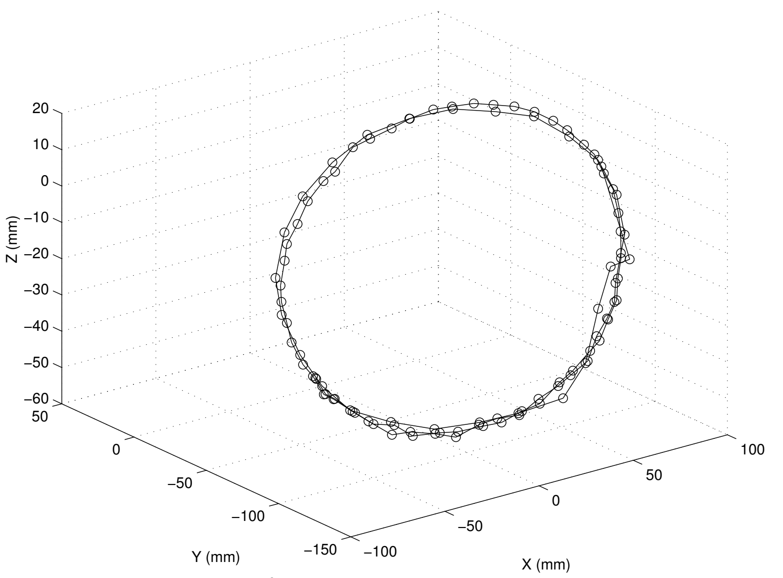


Figure 12: Two cameras tracking of a circular 3D trajectory with multi-passes and lines

changing the loop gain or tuning to finer iterations to avoid overshoot. An error of 0.6 inches (3.7%) is reported in a workspace of 16 inches.

In [10], a human-robot interface based on an interactive hand pointer is studied. By using a Sun Ultra 5 computer, they estimated various heights in the robotic scene and compared it to the actual height. They reported about 4% or less errors in height estimation for heights of 2 inches or more in the robotic scene.

In [15], a visual hand posture tracking problem is studied by using a 26-dof model of the hand. By using a Power Macintosh running at 180MHz, they were able to achieve a 15 frames per second rate with angular errors of at most 10% between the actual and reconstructed fingers.

In [11], a vision based human interface system is investigated. Authors used uncalibrated stereo vision techniques, and reported that the error between the actual nonlinear camera projection operator and the weak perspective projection operator is small, when the distance (d) between the camera and the object is large. More precisely, they reported that the error due to weak perspective projection assumption is about 1.5% of "d" when "d" is no less than 80 inches.

We proposed a real-time vision-based operator hand tracking system using a hand held four-ball feature frame. Using digital cameras, the static and dynamic errors of our tracking algorithm are 0.1% and 0.6% for a centered workspace of 20^3 inches^3 that is 40 to 60 inches distant from cameras. Using commodity PCs our refreshing rate is 9Hz. Due to the interactive nature of telerobotics absolute accuracy is useful but not critical especially in pre-positioning situations. However, one alternative to gain accuracy in pre-contact situations is to scale-down (1-to-1 to 1-to-0.2) the mapping of operator workspace to slave arm space which improves accuracy in contact phases at the cost of slowing down speed of teleoperation. To minimize delays Future direction is to overlap in time the tasks (threads) of grabbing data images (60 ms) and stereo matching (50 ms) which might significantly increases the refreshing rate. However, overall communication delays are influenced by network and communication protocols over the LAN or the Internet. These delays force the operator to use the stop-and-wait strategy in simple manual teleoperation.

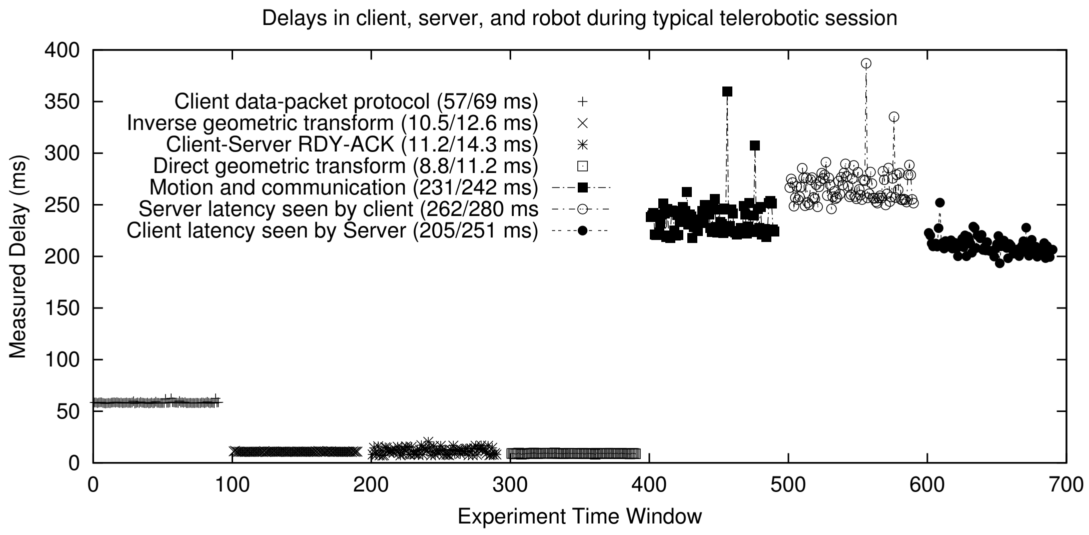


Figure 13: Delays in Client, Server, and PUMA with slow motion (0.5 mps)

The general consensus to overcome this problem is to raise the level of operator commands through the use of slave arm automated tasks that are activated and supervised by the remote operator.

7 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. The features of the proposed systems are (1) a metric for color matching to discriminate the balls from their background, (2) uniform and spiral search approaches to speedup the detection, (3) tracking in the presence of partial occlusion, (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 static and dynamic errors of tracking algorithm are 0.1% and 0.6% for a centered workspace of 20^3 inches^3 that is 40 to 60 inches distant from cameras. The running of the tracking algorithm on two PCs in parallel allowed (1) a parallel image grabbing delay of 60 ms, (2) a stereo matching delay of 50 ms, and (3) a global refresh rate of 9Hz. 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] T. Kazerooni, H. Tsay and K. Hollerback. A controller design framework for telerobotic systems. *IEEE Trans. on Contr. Syst. Technol.*, 1, Mar 1993.

[2] Ajit Shah. Developmnet of a telepresence surgical system. *WWW site of Defense Sciences Office*, October 1999.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] Y. Kuno, M. Sakamoto, K. Sakata, and Y. Shirai. Vision-based human interface with user-centered frame. *Proc. Inter. Conf. IRS'94*, 3:2023 –2029, 1994.

[12] 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.

[13] Y. F. Ho, H. Masuda, H. Oda, and L.W. Stark. Distributed control for teleoperations. *IEEE/ASME Trans. on Mechatronics*, 5, No 2:100–109, June 2000.

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

[15] F. Lathuiliere and J.-Y. Herve. Visual hand posture tracking in a gripper guiding application. *Proc. IEEE Inter. Conf. on Robotics and Auatomation*, 2:1688–1694, 2000.