Tracking a Sphere Dipole

Michael Greenspan[1,2] Ian Fraser[2]

michael.greenspan@ece.queensu.ca fraser@cs.queensu.ca

[1]Dept. of Electrical and Computer Engineering
[2]School of Computing
Queen's University, Kingston, Ontario, Canada, K7L 3N6

Abstract

A novel visual user interface is presented that is based upon monocular tracking of a sphere dipole, i.e., two spheres of common fixed radius separated by a fixed distance. The sphere dipole is a hand-held device, much like a pen. With a single calibrated camera, the 5 degree-of-freedom pose of the dipole, comprising 3 translations and 2 rotations, can be effectively tracked in real-time.

A method is derived to resolve the 3-D position of a sphere from the center and radius of its projected circle. Tracking the sphere dipole is done in two phases, locking *and* following. *The locking phase first locates the spheres by blob detection. The dipole is then followed by making use of the last known location of the two spheres. Following comprises the four processing steps of edge detection, robust circle extraction, outlier removal, and finally generating the best fit circle. Whereas the locking phase is fairly expensive, taking a few (i.e., ~ 1 or 2) seconds, following is computationally efficient and can be executed in real-time.*

Keywords: *circle detection, motion tracking, human computer interaction, visual computer interface*

1 Introduction

There has recently been a great deal of interest in the use of visual data for advanced human-computer interfaces. Many of these systems use visual tracking of a known object to determine cursor location. Some recent examples include the tracking of a hand-held sphere [1], and tracking the human nose [2].

These and other monocular systems consider only the 2-D position of the tracked object, and ignore the additional depth dimension that is perpendicular to the camera's retinal plane. The effective transmission of 3-D information to a computer has long been of interest, however, and there are devices that track the 3-D location of an object using active light sources, triangulating electromagnetic and ultrasonic triangulation ultrasonic signals, etc.

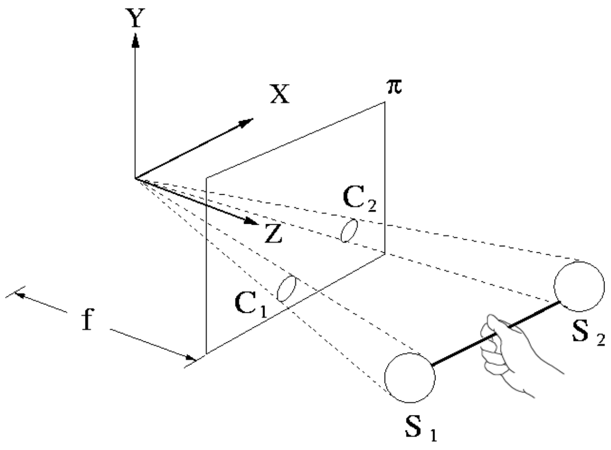


Figure 1: Projection of Sphere Dipole

In this paper we describe a new visual user interface that is based upon monocular tracking of a sphere dipole, i.e., two spheres of common fixed radius separated by a fixed distance. The sphere dipole is a hand-held device, much like a pen. With a single calibrated camera, the 5 degree-of-freedom (*dof*) pose of the dipole, comprising 3 translations and 2 rotations, can be effectively tracked in real-time.

This paper continues in Section 2 with a description of the properties of the projection of the sphere dipole onto the image plane. The processing steps for tracking are described in Section 3. Results and issues are described in Section 4, and the paper concludes with a discussion of future work in Section 5.

2 Projection of the Sphere Dipole

We consider the case of a single camera that has been calibrated so that its intrinsic parameters are known. In

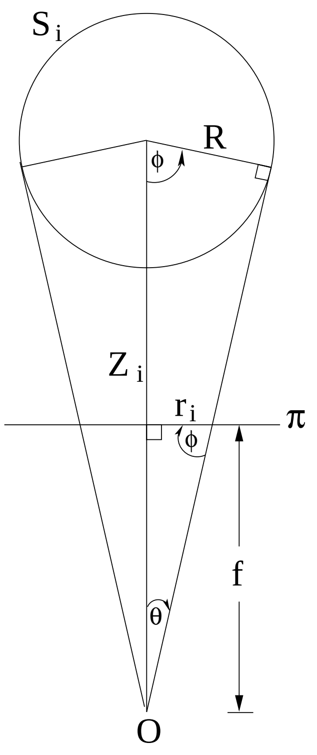


Figure 2: Conical Projection of S_i onto π

particular, the focal length f, aspect ratio s_x/s_y, and optical center (u_0, v_0) are known. We assume that the radial distortion of the optical system is negligible.

The projection of the sphere dipole onto the retinal plane is illustrated in Fig.1. Both spheres S_1 and S_2 have a common fixed radius R and are separated by a fixed distance D. Let the center of sphere S_i be denoted as $P_i = (X_i, Y_i, Z_i)$. For a given pose of the sphere dipole, the projection of S_i onto the retinal plane π produces a circle C_i centered at (x_i, y_i) with radius r_i.

By convention, use of the uppercase (X_i, Y_i, Z_i) denotes a coordinate in the camera reference frame, whereas the lowercase (x_i, y_i) denotes the corresponding coordinate projected onto π, which necessarily has a Z-value equal to f. The symbols (u_i, v_i) denote π locations in pixel coordinates. From the perspective model;

$$\begin{aligned} x_i &= fX_i/Z_i &= -s_x u_i - u_0 \\ y_i &= fY_i/Z_i &= -s_y v_i - v_0 \end{aligned} \tag{1}$$

2.1 Perspective Projection

The problem of detecting the location of circular features from their elliptical projections has been studied by [3, 4], and analytical methods have been developed for determining the 3 *dof* translation and 2 *dof* orientation for these features. In both of these cases, the solution for the cone centered at the camera frame origin and intersecting the projected ellipse was explicitly solved. These techniques were also extended to resolve the 3 *dof* position of a sphere from its circular projection.

Here we present a simpler solution based upon constructive geometry for the case of the projection of a sphere. Consider the illustration in Fig.2, which shows a cross section of the right circular cone which has the camera frame origin as its vertex and intersects π at C_i. From the relations $\sin\Theta = R/f$ and $\tan\Theta = r_i/f$ we derive $\cos\Theta = \frac{fR}{Z_i r_i}$. Substituting these values into $\sin^2\Theta + \cos^2\Theta = 1$ and taking the positive solution for Z_i gives;

$$Z_i = R\sqrt{1 + f^2/{r_i}^2} \tag{2}$$

2.2 Weak Perspective Projection

We make the assumption that the spheres are moderately far from π, which in practise is the scenario that we encounter. In this case, the projection onto π corresponds to a great circle of S_i that is parallel to π and is centered on P_i. Under this weak perspective assumption, the center of S_i can be approximated from the center and radius of its projected circle C_i.

Let $P_j = (X_j, Y_j, Z_j)$ be some point on the surface of S_i with corresponding projected point (x_j, y_j) on the circumference of C_i. As (x_i, y_i) is the center of C_i, its radius r_i can be expressed as;

$${r_i}^2 = (x_j - x_i)^2 + (y_j - y_i)^2 \tag{3}$$

Similarly, the radius R of S_i can be expressed as;

$$R^2 = (X_j - X_i)^2 + (Y_j - Y_i)^2 + (Z_j - Z_i)^2 \tag{4}$$

Substituting Eq. 3 into Eq. 4 gives;

$$\begin{aligned} R^2 &= (Z_j x_j/f - Z_i x_i/f)^2 \\ &\quad + (Z_j y_j/f - Z_i y_i/f)^2 \\ &\quad + (Z_j - Z_i)^2 \end{aligned} \tag{5}$$

From the weak perspective assumption, the vector $\overline{P_i P_j}$ is parallel to π, so that $Z_i = Z_j$. Substituting this and Eq. 3 into Eq. 5 gives;

$$\begin{aligned} R^2 &= (Z_i/f)^2[(x_j - x_i)^2 + (y_j - y_i)^2)] \\ &= (r_i Z_i/f)^2 \end{aligned} \tag{6}$$

As $Z_i > 0$, we take the positive solution;

$$Z_i = fR/r_i \tag{7}$$

From this relation, it is interesting to note that the relative depths of the two spheres can be determined independent from the camera intrinsic parameters, i.e. $r_1/r_2 = Z_2/Z_1$.

A similar relation holds if the spheres are not of equal radius. For example, if $R_1 = kR_2$, then $r_1/r_2 = kZ_2/Z_1$. This case is of less interest, as it introduces an ambiguity in the resolution of the pose of the sphere dipole, unless other means are used to discriminate between the identity of the spheres, such as intrinsic color properties.

Once the centers of the two spheres are determined, the 5 *dof* pose of the sphere dipole can be expressed by a 3 *dof* translation component and a 2 *dof* orientation component. The translation can be defined as the point $(P_1+P_2)/2$ which is exactly midway between the two spheres. The orientation can be defined by the polar coordinates of the vector $\overline{P_1P_2}$ that connects the two sphere centers.

3 Tracking

Tracking the sphere dipole is done in two phases, *locking* and *following*. The locking phase first locates the spheres by blob detection. Locking is invoked upon initialization, or if ever the dipole position is lost. After locking, the dipole is followed by making use of the last known location of the two spheres. Following comprises the four processing steps of edge detection, robust circle extraction, outlier removal, and finally generating the best fit circle. Whereas the locking phase is fairly expensive, taking a few (i.e., ~ 1 or 2) seconds on a conventional PC, following is computationally efficient and can be executed in real-time.

3.1 Blob Detection

Prior to blob detection, a pre-processing step determines the color signature of the spheres. One sphere is placed in a defined location in the image. Once the sphere is centered in this calibration region, the user enters a keystroke to capture the image frame for processing. Statistical analysis is then performed on the region to determine its color properties. The means μ_j and standard deviations σ_j are calculated for each of the respective RGB color channels $(j = \{R, G, B\})$ within the region.

During blob detection, the color signature is used to locate areas of possible interest. A circular mask C_M of radius r_M is convolved through the image. The value of r_M is chosen to be slightly smaller than the radius of C_i when S_i is at its farthest detectable distance from the retinal plane. For every location of C_M the three channel values of every underlying image pixel are compared to the predetermined statistics. If a certain percentage of these pixel values lie within 3 σ_j of μ_j for each channel, then the center of C_M is marked as a blob location. The percentage of pixels that must pass this test to correctly identify a sphere has empirically been determined as 75%.

True spheres will often give rise to circles that are larger than r_M, so that there may be many adjacent marked pixels for a single sphere location. We therefore extract the convex contour of each connected set of marked pixels [5]. The centroid of each contour is then passed to the edge extraction step. An example of an input image and the detected blobs is shown in Fig. 3(a) and (b). Both the hand and the mouse, which have similar intrinsic color properties to the spheres, are effectively filtered out.

Whereas the model we use for color signatures has the benefit of being simple, it has a number of deficiencies. There are more effective color spaces than RGB space for object discrimination [6], and possible improvements are discussed in Section 5. Also, this model assumes that the color signature of the spheres will be nearly constant at all image locations, and does not take external lighting conditions into account. This process futher assumes that both spheres have the same intrinsic color properties. If this were not the case, then it would have to be repeated independently for each dipole sphere.

3.2 Edge Detection

It has been shown by Canny [7] that when an estimate of the edge direction is known, edge detection can be effectively accomplished by the convolution of the 1-D signal in the direction tangent to the edge. This reduces the dimensionality of edge detection from a 2-D to a more efficient 1-D problem. The center of a blob resulting from a true sphere S_i will be close to the center of C_i. A ray emanating from this center in any direction will therefore intersect C_i at close to the tangent direction, as shown in Fig.3(c). Using ray following to determine edge direction is much more efficient than the standard techniques of image sharpening, which involve 2-D convolutions.

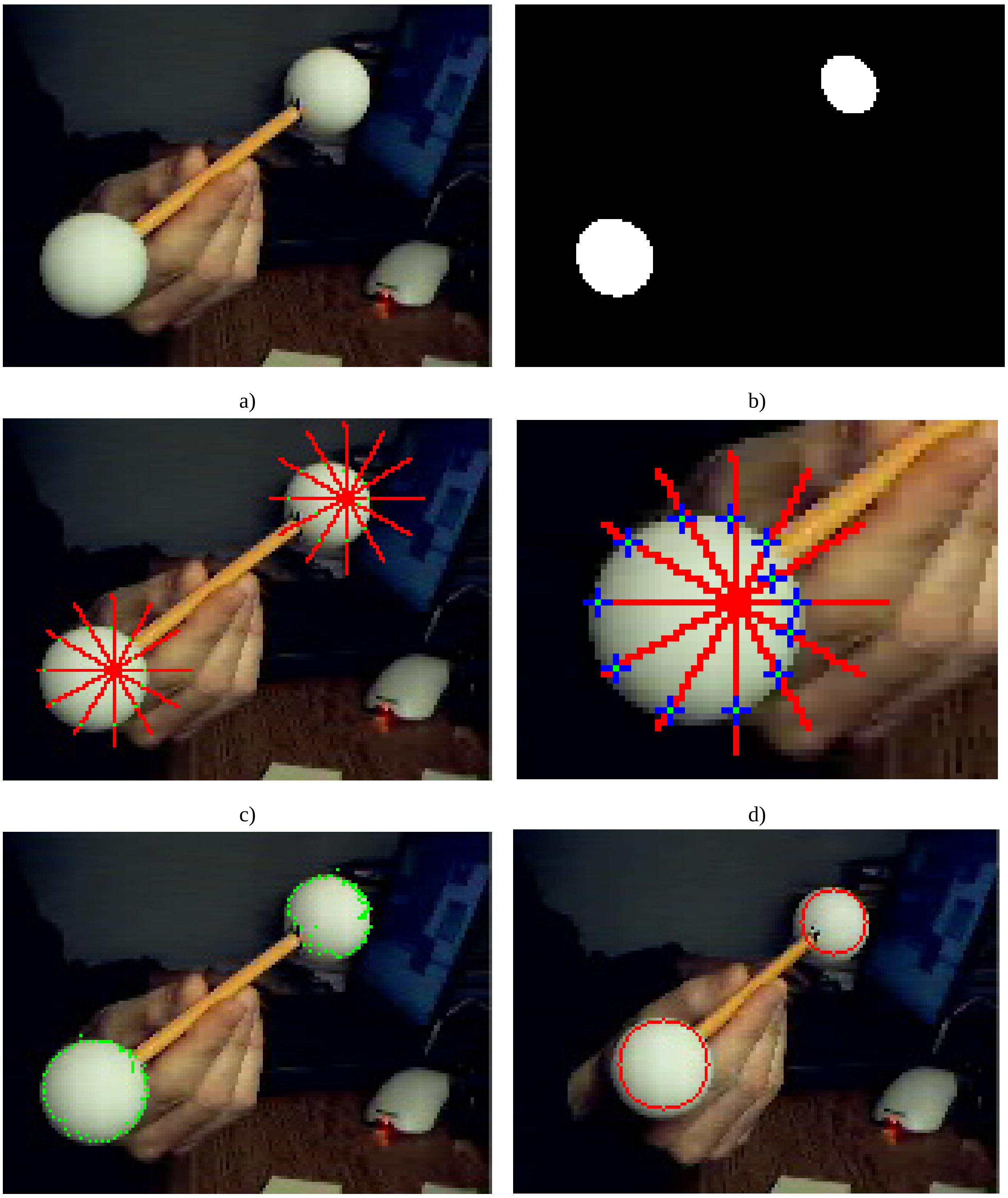
For each blob, rays are emanated from the center at regular intervals which we have chosen to be 5^o. The total number of detected edge points for each blob will therefore be 72. For each ray, the sequence of underlying image pixels form a 1-D signal.

Starting from the blob center, the image is sampled at pixel increments along the ray direction to form the signal value. At each sampling step, the value m_j for each channel j of the underlying pixel is compared to the color signature mean μ_j that was determined during preprocessing. The signal value $g(i)$ at each step i is set to be;

$$g(i) = \sum_{j=R,G,B} (\mathit{max} - (\mu_j - m_j))^2 \tag{8}$$

where *max* is the maximum channel value (i.e., 255). In this way, the signal is a measure of similarity to the color signature where the highest value (3×255^3) indicates perfect similarity, and low values indicate dissimilarity.

To detect a transition, the signal for each ray is convolved with a 1-D mask that is based upon the derivative of the Gaussian, which has been shown by Canny to be close to the optimal for step edges [7]. The peak resulting from this convolution identifies the point that borders the edge and lies within the sphere. This point is extended by



a) b)

c) d)

e) f)

Figure 3: *a) image, b) extracted blobs, c) rays (every 30^o), d) ray edge points (left sphere), e) all edge points, f) tracked circles*

another half-pixel in the ray direction to provide a more accurate estimate of the edge location.

An example of the rays emanating from a blob representing a true sphere is illustrated in Fig. 3(c). For clarity, only every 6th ray at 30^o increments are illustrated. The 1-D signals and their convolutions corresponding to the 3 rays oriented at $2:00, 3:00$, and $8:00$ are plotted in Fig. 4. A derivative of Gaussian mask of size 5 was used, i.e., $[3, 5, 0, -5, -3]$. The edge points resulting from the peaks of the convolved signals are shown in Figure 3(d). Of the 3, it can be seen that only the edge point extracted from the $3{:}00$ ray is correct. The signal at $2{:}00$ changes abruptly at radial distance 6, which causes a false peak in the convolved signal. Conversely, the ray at $8{:}00$ changes very gradually, and exhibits a false peak due to a shadow effect. This shadow effect was common in the sphere regions that where oriented obliquely from the lighting source. Fig. 3(e) shows all 72 detected edge points from each ray. While some of these points are noisy and do not correspond to the true edge, most of them are quite true. This was often the case under reasonable lighting conditions.

The length of the emanated rays may affect the success of the edge detection. If they were too short, then false points would be detected, as the true edge would not actually be encountered. Conversely, if they were too long, then a peak that corresponded to the image background could dominate, resulting again in a false edge point. To limit these effects the ray length is based upon a predefined limit to the a maximum sphere movement between frames, which is proportional to the length of the emanated ray. In the initial detection, the ray length is limited to be slightly larger than the radius of C_i when S_i is positioned at its nearest detectable location to the image plane.

The output of edge detection are sets of 72 edge points, one for each possible sphere location, which are next passed to the circle extraction step.

3.3 Circle Extraction

There exist a number of standard circle extraction techniques, such as the Hough Transform [8], the UpWrite [9], and recently the Randomized Circle Detection [10] and [11]. These techniques are designed for use in complex scenes which contain clutter and occlusions, where there may be many outliers and multiple circles. Our scenario is much simpler, as there is only one possible circle per blob. As well, if the blob is a true circle, then the outliers are mostly due to poor lighting conditions and will only comprise a small portion of the 72 edge points.

For these reasons, it is not necessary to employ such general techniques as those noted above, and a straightforward random sampling and consensus (RANSAC) algorithm [12] is sufficiently powerful and efficient. At each

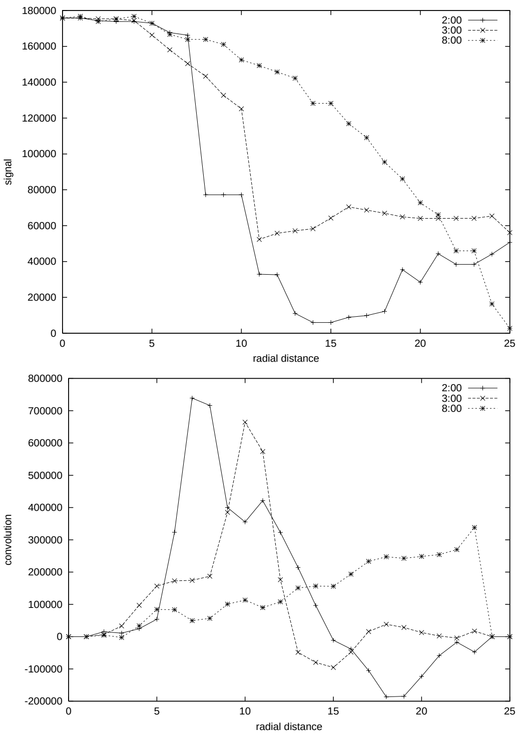


Figure 4: Radial Signal and Convolution

iteration, three edge points are randomly selected, and the center and radius of the unique circle defined by these 3 points is determined. A score value is set to zero, and the distances between all edge points and the circle are calculated. Every point that is closer than a certain maximum distance, chosen to be $\sqrt{2}$ pixels, is deemed to fall on the circle, and the score value is incremented. In this way, the arbitrary selection of the three points to generate the circle model is the *random sampling* stage of the algorithm, and the calculation of the score is the *concensus* stage. The process iterates by randomly selecting three new points and repeating. At the end of a predefined number of iterations, the circle with the largest score is returned.

As we know the exact number of edge points, we can determine the number of iterations required to identify a circle within a certain likelihood of success. The number of ways to select subsets of 3 points from a set of N is $\binom{N}{3}$, which for $N=72$ is 59640. If all of the points are inliers (i.e., lie on the circle) then it is necessary to select just one of these subsets to extract the true circle. Alternately, if any of the points are outliers, then it is necessary to generate and test all of the subsets to identify the true circle with 100% certainty.

Let e be the ratio of outliers in the set of N points. The number K of subsets of 3 points that has at least one outlier is;

$$\begin{aligned} K = \frac{1}{3!}[\ (N(1-e))(N(1-e)-1)(Ne) \\ +\ (N(1-e))(Ne)(Ne-1) \\ +\ (Ne)(Ne-1)(Ne-2)\ ] \end{aligned} \tag{9}$$

The probability of selecting M such subsets consecutively is then $(K/\binom{N}{3})^M$. Conversely, the probability of selecting M subsets such that at least one subset does not have an outlier is;

$$1 - \left(\frac{K}{\binom{N}{3}} \right)^M \tag{10}$$

An application of Eq.10 shows that, for $N = 72$ and assuming the proportion of outliers to be 25%, only $M = 3$ random samples of 3 points are required to identify a true circle with 99% confidence. For 50% outliers, M=5 samples are required. In practise a single iteration of the random sampling is quite efficient, so we tend to use a value of $M > 50$ iterations so that detection of the true circles is almost a certainty, even when the proportion of outliers is large.

Outliers are identified as those points that do not lie within $\sqrt{2}$ pixels of the extracted circle's circumference. The circle is then verified by determining the number of inliers that satisfy the color signature, and it is discarded if greater than 10% of the inliers are not within $3\sigma_j$ of the mean color μ_j for each channel j. The inliers of the remaining circles are then passed to the optimal circle fitting step.

3.4 Optimal Circle Fitting

For each subset of 3 points chosen from the inliers, there will in general be a unique circle. It still remains therefore to determine the equation of the circle that best describes these data. It has been shown in [13] that an exact (albeit biased) solution to the best-fit circle can be derived by minimizing the error of the circle area. The center $(\bar{x}, \bar{y})$ and radius r are defined as;

$$\begin{aligned} \bar{x} &= \frac{c_1 b_2 - c_2 b_1}{a_1 b_2 - a_2 b_1} \\ \bar{y} &= \frac{a_1 c_2 - a_2 c_1}{a_1 b_2 - a_2 b_1} \end{aligned} \tag{11}$$

$$r^2 = \frac{1}{n}(\textstyle\sum_{x^2} - 2\sum_x \bar{x} + n\bar{x}^2 + \sum_{y^2} - 2\sum_y \bar{y} + n\bar{y}^2) \tag{12}$$

Here, the summations are taken across all n inliers, and the parameters are defined as;

$$\begin{aligned} a_1 &= 2({\textstyle\sum_x}^2 - n\textstyle\sum_{x^2}) \\ a_2 = b_1 &= 2(\textstyle\sum_x \sum_y - n\sum_{xy}) \\ b_2 &= 2({\textstyle\sum_y}^2 - n\textstyle\sum_{y^2}) \\ c_1 &= (\textstyle\sum_{x^2}\sum_x - n\sum_{x^3} + \sum_x \sum_{y^2} - n\sum_{xy^2}) \\ c_2 &= (\textstyle\sum_{x^2}\sum_y - n\sum_{y^3} + \sum_y \sum_{y^2} - n\sum_{x^2 y}) \end{aligned} \tag{13}$$

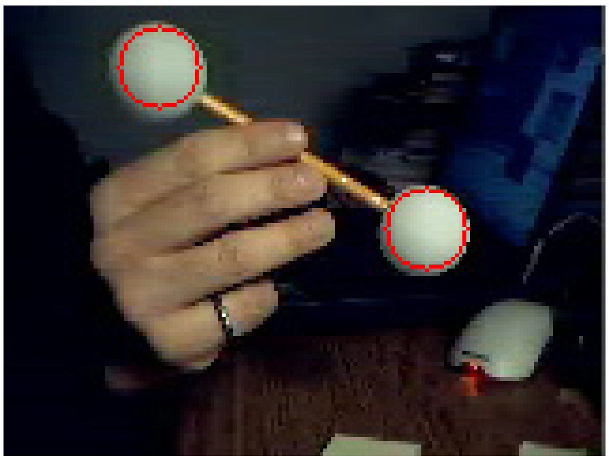
4 Discussion

The method was implemented on a PC platform using the Intel OpenCV computer vision library [14]. The sensor used was an Intel Pro Video PC Camera, which is an inexpensive commodity Webcam, which produced images of 320×240 pixel resolution. Our sphere dipole was constructed from two table-tennis balls glued to the end of a pencil. The sphere dipole was not constructed to any measurement specifications, and this limited the accuracy of the system, as did the limited accuracy of the camera calibration. The routines supplied in the OpenCV library were used for camera calibration, but the accuracy of the resulting calibration was not determined. Also, the radial distortion parameters were ignored.

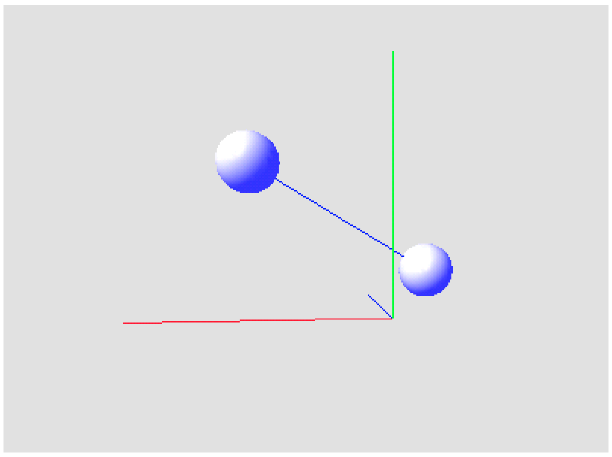
Qualitatively, the system was believed to work well, tracking the sphere in real-time for a number of motions under a variety of reasonable lighting conditions. Some examples of the tracked dipole are illustrated in Fig. 5. Whereas the achievable measurement accuracy is of interest, it is not crucial for the main anticipated applications. The system is very inexpensive, comprising only a single Webcam and the sphere dipole. Consequently, this technology is perfect for consumer applications where the accuracy is not that important of a factor, but cost is, such as games, 3-D graphic design, and animation. Systems that require high accuracy, such as image-guided medical systems and AR, are likely to have a higher tolerance for costlier solutions based upon active and multiple sensors.

5 Conclusions

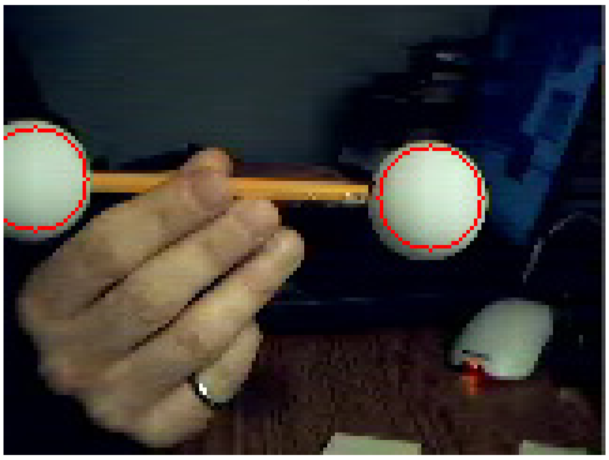
In this paper we have presented a simple technique to extract the 3-D locations of a pair of spheres. This is done by analyzing the centers and radii of their projected circles onto the image plane. Using the intrinsic parameters of the sensor, this allows 5 dofs to be extracted when the spheres are connected in a dipole device. This information is then used to develop an algorithm for real-time tracking of a sphere dipole. Our implementation of these techniques used commodity hardware, and the success of the system so far suggests that these techniques have significant potential for 3-D consumer interfaces. This is particularly promising for CAD, 3-D graphic design, and game applications.



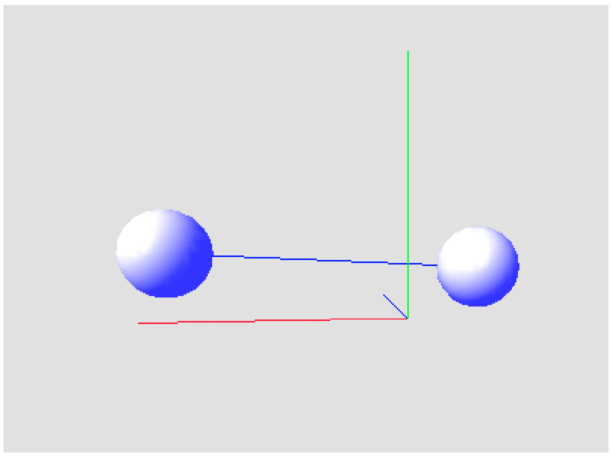
a)



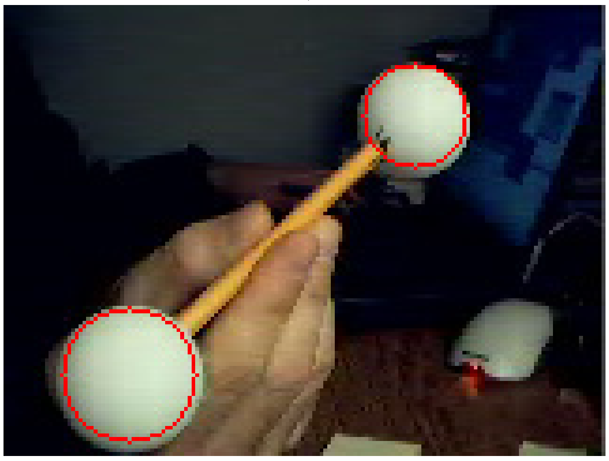
b)



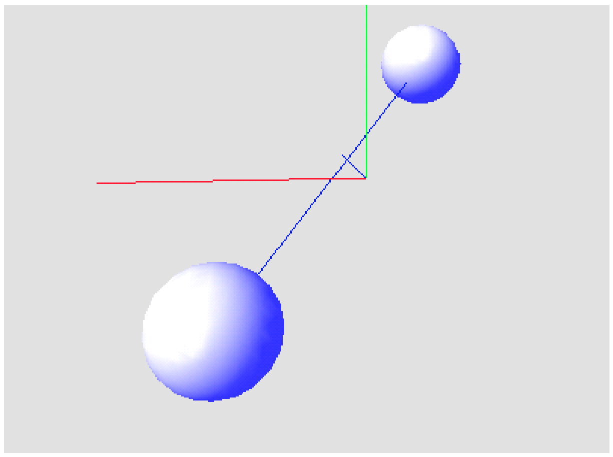
c)



d)



e)



f)

Figure 5: Examples of Tracked Sphere Dipole

Future research will be focused towards improving the overall robustness of the system. By changing the color space to a more representative color model, the blob detection and edge detection will become more invariant to the lighting conditions. This will improve functionality in the widely different lighting conditions expected in an uncontrolled office or home environment. A good candidate for this is Hue, which is a representation of the intrinsic color of the object. In Hue space white is a high intensity highly saturated color, the color of the spheres should be changed to provide a more definitive color signature.

The color signature is currently determined by sampling the sphere at one location in the scene. It may be beneficial to sample the sphere at multiple locations to account for the changing illumination conditions throughout the scene.

The convolution mask for the edge extraction step is currently based upon the derivative of the Gaussian. This has been shown to be near optimal for step edges, but tends to underestimate the sphere edge, as can be seen in Fig. 5. It may be possible to design a mask which is particularly tuned to the signal that is encountered along a ray, thereby increasing the accuracy and robustness of the edge extraction.

Robustness of the initial locking phase will be improved by adding a distance pruning method. Currently, in blob detection, false regions may be extracted from background features that happen to match the color signature. In the current process, these false regions will result in circles, so that the number of detected circles can be greater than 2, which confuses the dipole localization. An improvement would compute the relative distance of all pairs of extracted spheres, and discount (prune) those pairs that are outside of the true sphere dipole separation distance D.

Experiments will also be conducted to determine the accuracy and repeatability of the system. This could be done by having a robotic arm hold the dipole and repeat a series of movements. The robot provides a repeatable series of movements which can easily be compared to the output of the system. Included in this is the development of representative measures to quantify the effectiveness of the system.

We also plan to interface and apply the system to a number of example applications, such as games.

Acknowledgements

The authors would like to thank the reviewers for their helpful comments, and NSERC for their support of this work.

References

[1] David Bullock and John Zelek. Real-time tracking for visual interface applications in cluttered and occlusing situations. In *Vision Interface 2002*, pages 93–100, May 2002.

[2] D.O. Gorodnichy, S. Malik, and G. Roth. *Nouse*‘use your nose as a *mouse*’- a new technology for hands-free games and interfaces. In *Vision Interface 2002*, pages 354–361, May 2002.

[3] Y.C. Shiu and Shaheen Ahmad. 3d location of circular and spherical features by monocular model-based vision. In *IEEE Intl. Conf. Systems, Man, and Cybernetics*, pages 567–581, 1989.

[4] Reza Safee-Rad, Ivo Tchoukanov, Kenneth Carless Smith, and Bensiyon Benhabib. Three-dimensional location estimation of circular features for machine vision. *IEEE Trans. Robotics and Automation*, 8(5):624–639, Oct. 1992.

[5] S. Suzuki and K. Abe. Topological structural analysis of digital binary images by border following. *CVGIP*, 30(1):32–46, 1985.

[6] T. Gevers and A.W.M. Smeulders. Color based object recognition. *Pattern Recognition*, 32:453–464, March 1999.

[7] John Canny. A computational approach to edge detection. *TPAMI*, 8:679–698, 1986.

[8] Raymond K.K. Yip, Peter K.S. Tam, and Dennis N.K. Leung. Modification of hough transform for circles and ellipses detection using a 2-dimensional array. *Pattern Recognition*, 25(9):1007–1022, 1992.

[9] Robert A. AcLaughlin and Michael D. Alder. The hough transform versus the upwrite. *TPAMI*, 20(4):396–400, 1989.

[10] Teh-Chuan Chen and Kuo-Liang Chung. An efficient randomized algorithm for detecting circles. *Computer Vision and Image Understanding*, 83:172–191, 2001.

[11] Euijin Kim, Miki Haseyama, and Hieo Kitajima. A new fast and robust circle extraction algorithm. In *Vision Interface 2002*, pages 439–446, May 2002.

[12] M.A. Fischler and R.C. Bolles. Random sample consesus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[13] Samuel M. Thomas and Y.T. Chan. A simple approach for the estimation of circular arc center and its radius. *CVGIP*, 45:362–370, 1989.

[14] http://www.intel.com/research/mrl/research/opencv/.