A Virtual 3D Blackboard: 3D Finger Tracking using a Single Camera [*]

Andrew Wu
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
awu@uiuc.edu

Mubarak Shah and N. da Vitoria Lobo
School of Computer Science
University of Central Florida
Orlando, FL 32816
shah, niels@cs.ucf.edu

Abstract

We present a method for tracking the 3D position of a finger, using a single camera placed several meters away from the user. After skin detection, we use motion to identify the gesticulating arm. The finger point is found by analyzing the arm's outline. To derive a 3D trajectory, we first track 2D positions of the user's elbow and shoulder. Given that a human's upper arm and lower arm have consistent length, we observe that the possible locations of a finger and elbow form two spheres with constant radii. From the previously tracked body points, we can reconstruct these spheres, computing the 3D position of the elbow and finger. These steps are fully automated and do not require human intervention. The system presented can be used as a visualization tool, or as a user input interface, in cases when the user would rather not be constrained by the camera system.

1 Introduction

The expression of 3D concepts in a classroom environment has traditionally been limited to a 2D media, the blackboard. A professor who wishes to communicate the idea of a "saddle point" in Calculus must either attempt a 2D chalk rendering or find some physical analogue to display. A similar difficulty is encountered in human-computer interactions, where users often manipulate non-intuitive control devices. In both environments, standard means of expression become unwieldy when users try expressing higher dimensional concepts.

This paper describes a system that can infer the 3D location of a finger, letting users describe a geometrical object in 3D through their gestures. Besides tracking the finger, the system provides, as a by-product, a partial model of the pose of certain parts of the user's body. In particular, we are able to record the relative 3D positions of a user's finger, elbow, shoulder and head. After initial training for skin detection, the system is fully automated and does not require human intervention.

Researchers have been working on the problem of a gesture-based interface for several years now [1, 2, 4, 6, 7]. Some of the systems implemented have been able to track 3D hand position rather accurately. Glove-based systems, such as "Charade" by Baudel and Beaudouin-Lafon [1], require the user to wear a tethered glove. Vision researchers have had similar success without the use of special hardware attached to the user's hand. In GestureVR [6], Segen and Kumar use a stereo pair of calibrated cameras. For depth information, Maggioni [4] as well as Kjeldsen and Kender [2] analyze the size of the projection of the user's hand. Rehg and Kanade [5] describe DigitEyes, a model-based hand tracking system that uses two cameras to recover the state of hand.

Our system uses several techniques previously developed for hand tracking, but extends certain methods to handle a larger view window and to overcome limitations of a single camera. The system was specifically designed to avoid the need to calibrate the camera, to employ multiple cameras, or to restrain the user's hand to a limited area.

The system first automatically identifies two arms and the face using skin detection methods on each frame. Then the finger tip of the gesticulating hand is determined by identifying the pixel on the arm contour at which the dot product of two vectors along the contour is maximum. The head, shoulder, and elbow positions are determined in order to measure the length of the projections of a user's upper and lower arm.

Previous approaches typically only consider images where only one hand and a small portion of the fore-

[*]The research reported here was supported by the National Science Foundation grant EIA-9732522.

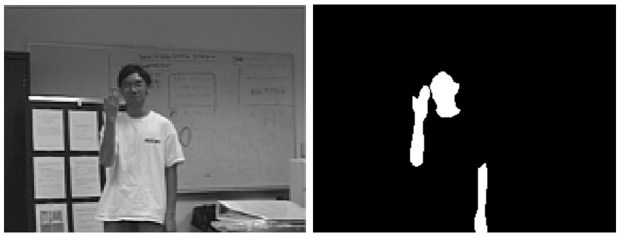

Figure 1: Example skin image and training mask pair

arm are visible, reducing complexity. By using a larger view window, we are able to use information about the rest of the body in our 3D finger tracking methods. Since bones have constant length, if one end is held motionless, the possible locations of the other end can be described by a sphere whose radius is equal to the length of the bone. From a sequence of images, if we can find certain body parts we can reconstruct these spheres and thus the 3D representation of a user's arm.

The organization of the rest of the paper is as follows. In the next section, we will develop methods for arm segmentation through skin detection, and arm finding through centroid tracking. We will discuss techniques for handling occlusion and use of the contour of the arm to determine the location of the finger. Section 3 will deal with body tracking. In section 4, we will show how we can combine these data to accurately track the 3D position of a user's finger. To improve finger tracking, we apply smoothing to the 3D trajectories, which is discussed in section 5. Results will be presented in section 6, followed by discussion in section 7.

2 Arm Segmentation

2.1 Skin Detection

For the detection of skin regions, our program uses a precalculated histogram-like structure called a Color Predicate (CP). We use the method developed by Kjeldsen and Kender [3], who also applied the CP toward finding skin. We trained our Color Predicate with several color images and corresponding hand-drawn binary masks (Figure 1). After training, the CP data was stored and used repeatedly for different image sets.

The application of the Color Predicate to a novel color frame produced a binary skin image. We used a median-filter on the binary skin image to reduce noise. The resulting skin image was clear, with few gaps. Sometimes packets of noise would appear, where background pixels were mistakenly identified as skin. These pixels, however, were removed using a size filter in later processing because they always formed small regions that could safely be ignored.

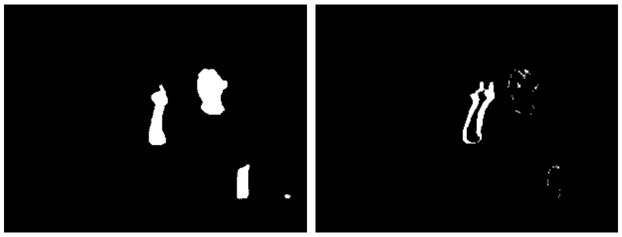

Figure 2: Skin output and a difference of two consecutive skin pictures

2.2 Identifying the Arm

In the binary skin image, we expect three regions – the head and two arms. Segmentation of the skin image uses connected components to identify large contiguous regions. The three largest regions of at least a certain pixel size are culled, and their centroids are calculated.

These largest skin regions are identified in each frame of a sequence. Next, we match centroids from one frame to nearby centroids in the next, insuring continuity. Assume there are i centroids in one frame, represented by $c_1, \ldots, c_i$, and j centroids in the next frame, represented by $d_1, \ldots, d_j$. We try to match centroids by re-ordering $d_1, \ldots, d_j$ so that we minimize $\sum_{n=1}^{min(i,j)} |c_n - d_n|$, where $|c_n - d_n|$ is the Euclidean distance between c_n and d_n. Later on we will discuss how the system handles occlusion, which changes the number of centroids found from frame to frame $(i \neq j)$.

To find the arm, we assume that the fastest moving centroid belongs to the gesticulating arm. To calculate motion, we compute the distance each centroid moves from one frame to the next. If the largest centroid delta is lower than a certain threshold, we note that there is insufficient motion to distinguish the arm, so we use information about which centroid was accurately tracked as the arm in previous frames. However, in most cases there is enough motion to be able to identify a moving arm. Figure 2 shows the motion of centroids by way of a difference picture. The difference of two consecutive skin frames shows a significant motion of the gesticulating arm, and small motion around the head and right arm.

2.3 Finding the Finger

Using the segmented arm region, we can find the most likely position of the finger in the region. We first calculate an edge contour of the arm by noting that any pixel on the arm region that has 4-connectivity with a non-arm region belongs to the boundary. Traversing this pixel outline in an arbitrary direction, we determine a list of connected contour pixels.

We use an approximation to k-curvature, the dot product. Segen and Kumar [6] compute k-curvature, whose measure is defined by the angle between two vectors $[P(i-k), P(i)]$ and $[P(i), P(i+k)]$, where k is a constant and $P(i) = (x(i), y(i))$, the list of contour points. Segen and Kumar used local curvature extrema, but we need only one single finger point. Instead, we calculate the dot product for each point on the outline.

For appropriate values of k, the value of the dot product can be used to find the finger tip. If we take a pixel near the finger and extend two vectors k pixels away along the outline, the vectors will point in similar directions and thus will have a high value of a dot product. The vectors formed at a pixel on the arm will have a large but negative dot product. Thus, we compute the dot product for all outline pixels and find the highest values. Typically, there is a unique highest value, which is the finger tip. When multiple pixels have equally high dot products, we choose the pixel that is closest to the previous finger position.

Figure 3 shows an example for dot product calculations. Both vectors c and b extend outward from the darker pixel. For the example where $c \cdot b > 0$ (left), $k = 3$. For the other examples(center and right), $k = 2$.

In all images where the finger was visible in the skin image, the approach worked well. When the finger occludes portions of the hand, the method finds reasonable estimates, because a hand generally has a higher curvature than other parts of the arm. Figures 4 and 5 show some example results.

We can also use the value of the dot product $c \cdot b$ to judge the accuracy of our tracker. If $c \cdot b > 0$, we record a high level of "confidence". If $c \cdot b = 0$, we record a lesser confidence level, and if $c \cdot b < 0$, we mark the tracking done in the current frame with a low level of confidence. Most frames in our test data had $c \cdot b > 0$.

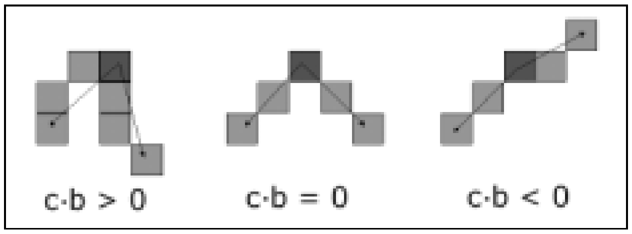

Figure 3: Example dot product calculations

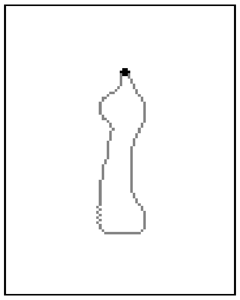

Figure 4: Contour of arm, highest dot product marked

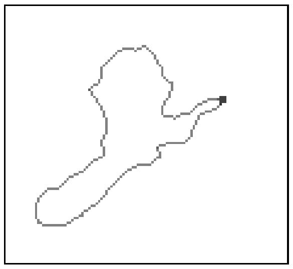

Figure 5: Contour of arm/head region, highest dot product marked

2.4 Occlusion

Previous sections assumed that there were equal number of centroids found in all frames. This is not always true, because at times a moving arm can occlude the head or the other arm. When this happens, a centroid is apparently lost, throwing off calculations that try to match centroids between two frames or that find the centroid which moved the most.

Our approach is to place the missing "ghost" centroid on top of the centroid of the largest region. Ignoring the missing centroid would cause problems when trying to find the moving hand. In figure 6, for example, the arm region intersects with the face region in the middle pair of images, causing problems in the next frame (bottom pair).

When attempting to correlate centroids between frames, if the number of centroids are not equal, we cannot calculate the distance moved for each pair of centroids since there is a lack of parity. Thus, we employ a "ghost" centroid. This "ghost" centroid is ignored for any mathematical calculations, but when two centroids appear to become three – let's say if the hand that occluded the face moves away – the matching algorithm mentioned before would work properly.

Applied to a sequence of twenty images, spanning several seconds, the program was able to output a trajectory of the recorded finger positions. Even in the presence of occlusion, the results were quite good.

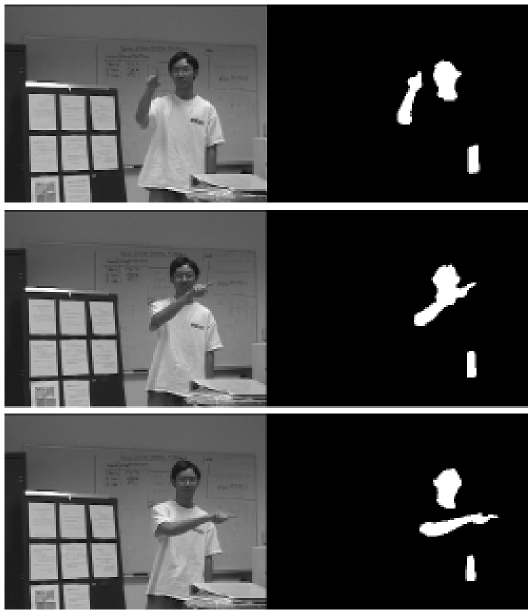

Figure 6: Example of occlusion

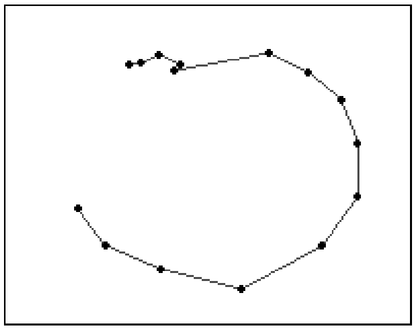

Figure 7: Circular trajectory of finger positions throughout a partial circle motion

Figure 7 shows the trajectory produced when a user moved his hand in a large circle ($radius \approx 1m$). The first discontinuity, at about the fourth point from the upper-left, was caused when the tracker loses the finger as it occludes the face. To smooth the results, we can apply an averaging computation. For every point, excluding the first and last, we calculate the midpoint of the previous and next point. The smoothed point is the average of the midpoint and the recorded point. The resulting trajectory (Figure 8) is more easily identified as a circular shape. Later in this paper we will describe how we use a model of error to determine when smoothing is necessary.

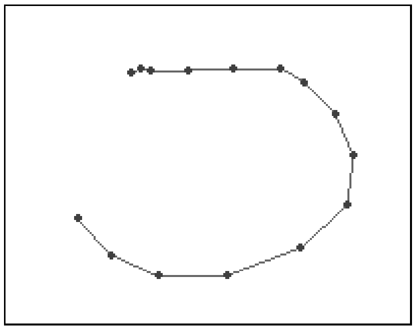

Figure 8: Circular trajectory smoothed with average filter

3 Finding Other Parts of the Body

We track other parts of the body: head, shoulders and elbow, for two reasons. First, we need more data to derive the 3D position of the finger. Second, we can use this body data to improve finger tracking when we encounter a low confidence level in tracking.

3.1 Head and Shoulders

The location of the head is only indirectly used. Thus, we do not need a sophisticated head tracker. We simply find the highest centroid (lowest Y coordinate) that is not the moving arm. When there is occlusion, we do not track the head. Rather we use previous information about the head's position.

Now that we have the centroid of the head region, we can approximate the location of the shoulder. What matters for our system is that the relative position of the shoulder, compared to the rest of the body, remains constant. To accomplish this, we notice that if the user faces forward, the shoulder maintains a fixed distance from the center of the head. We first find the "radius" r, of the head. Even though the head region is not perfectly circular, we can measure its area, a_{head}. Thus, an approximation to the radius of the head is $r = \sqrt{\frac{a_{head}}{\pi}}$.

The location of the shoulder is approximated by assuming that if the centroid of the head is at (x, y), then the location of the shoulder is at $(x - r \cdot \Phi, y + r)$, where $\Phi = \frac{1+\sqrt{5}}{2}$, the golden ratio. The shoulder's x position is $x + r \cdot \Phi$, if the user is left-handed. This approximation is fairly accurate when the user is facing the camera. If we find that the approximation is inaccurate because the shoulder point is in the background of the image, we move the shoulder point to the nearest non-background pixel.

3.2 The Elbow

Next, we find the elbow. Since we know the position of the finger and we have a skin image of the arm, if we assume the skin of the elbow is visible we

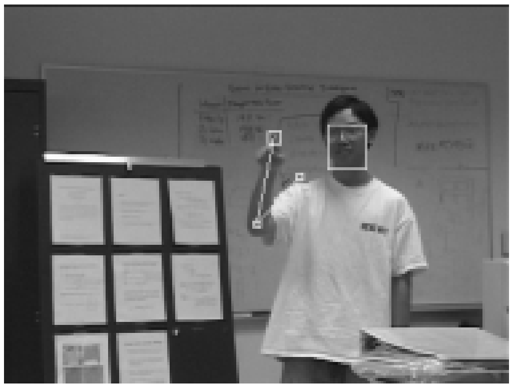

Figure 9: Skeleton, superimposed

can find the elbow's position. We erode the arm skin image several times, approximating a skeletonization algorithm. Then, we find a point on the skin image that maximizes the Euclidean distance between that point and the finger. This is most likely the elbow point.

Combining the body points tracked and estimated, we can form a skeleton of the user's upper body pose. Figure 9 shows an example image. The finger and body tracking is robust enough for our purposes, and has worked for varying sets of image inputs. When the finger is pointing towards the camera, the system occasionally mistracks the elbow as the finger. To handle such worst-case scenarios, we apply a simple rule, checking if distance from the shoulder to the elbow exceeds the distance from the elbow to the finger, and if the finger is closer to the shoulder than the elbow. If the rule is satisfied, we swap the coordinates of the finger and the elbow. In test runs on real data, the above condition was rarely satisfied.

4 Deriving 3D Information

Armed with body data, and with the accurate tracking results obtained as described above, we can begin to derive 3D information. We first assume that in each frame, the user will be more or less facing the camera directly. This assumption is valid if the user knows about the camera and faces it. The assumption also holds for normal usage, as the user will most likely be communicating toward an audience. In that case, we can identify the vector of communication and place the camera system near the audience so that the camera sees the front of the user's body.

We also assume an orthographic projection. For perspective projection, the location of a point (x, y, z) in the image plane is (x', y'), where $x' = \frac{f}{z}x$, $y' = \frac{f}{z}y$, and f (focal length) is a constant. For two points (x, y, z) and $(x + \Delta x, y + \Delta y, z + \Delta z)$, when z is large and Δz is small, $x \approx cx'$ and $y \approx cy'$ because $\frac{x}{z \pm \Delta z} \approx \frac{x}{z}$. c is another constant. For our system, z, the distance between the camera and user, is much larger than Δz, the length of the user's arm, so an orthographic projection model remains fairly justifiable.

Before we can interpret the pose of the body, we need to know the true pixel length of the arm. A calibration image is not required if we assume that in at least one image of our sequence, the arm is fully extended (Figure 10). We store the longest pixel lengths of the Humerus (upper arm) and Radius (lower arm), as true pixel length of the arm.

4.1 Spherical Kinematics

Human bones have consistent length. Since the distance from the elbow to the shoulder is fixed, the elbow can only move tangent to a certain sphere. The center of this sphere is the shoulder, and its radius is the length of the upper arm. Ignoring the half-space behind the plane of the body, we are left with the surface of one hemisphere. If we look at the hemisphere along the polar axis, we can see the entire surface of the hemisphere. That is, along the polar axis, we can collapse the hemisphere from 3D to 2D without a loss of information, as well as recover the hemisphere from its 2D projection.

In algebraic terms, we can describe all possible locations (x, y, z) of the elbow if we know r, the length of the upper arm. The shoulder is placed at the origin. l is the measured length of the 2D projection of the upper arm in the current image.

$$x^2 + y^2 + z^2 = r^2 \tag{1}$$

$$z^2 = r^2 - (x^2 + y^2) \tag{2}$$

$$z = \pm\sqrt{r^2 - l^2} \tag{3}$$

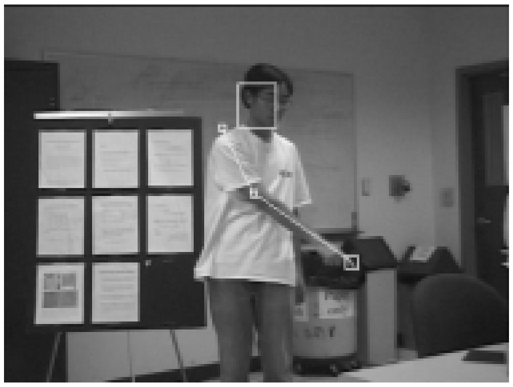

Figure 10: Fully extended arm

Ignoring the half-space behind the body plane,

$$z = +\sqrt{r^2 - l^2} \tag{4}$$

Since we can directly measure l and have a reasonable estimate for r, we can easily calculate z. This z is a relative Z. If we set $z_{shoulder} = 0$, then $z_{elbow} = 0 + z$. Using a similar line of reasoning, we see that the finger can only move in a certain sphere around the elbow. We calculate relative Z as above, using the length of the forearm and the length of its 2D projection. Therefore $z_{finger} = z_{elbow} + z_{relative}$, where $z_{relative}$ is solved from equation 4. Applying these equations to one frame of a sequence, we derive 3D information about the user's arm (See Figure 11).

When we ignore the negative solution for equation 4, we imply that the finger will always be in front of the elbow, and that the elbow will always be in front of the shoulder (from the user's point of view). The latter is almost always true for all gestures, while the former may not be true. In most gestures, the finger points away from the body, but there are a few gestures where this is not the case. The assumption of an orthographic projection also has an effect, but since we are concerned with the relative 3D positions of the finger, a slight warping is unimportant and can be ignored.

5 Smoothing

To improve the results of our finger tracking, we can apply smoothing to the 3D trajectories to reduce the effect of noise. If we can estimate error, we can adjust the level of smoothing based on this estimated error. The model of error we used was based on the linear combination of four parameters. We record confidence, which is inversely proportional to error.

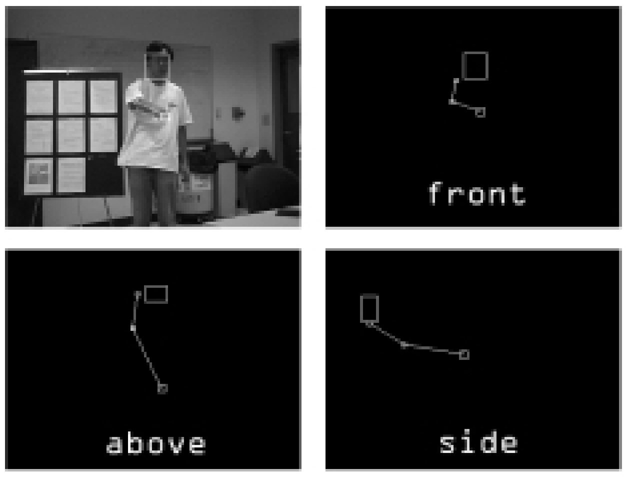

Figure 11: Various views of a user's 3D pose

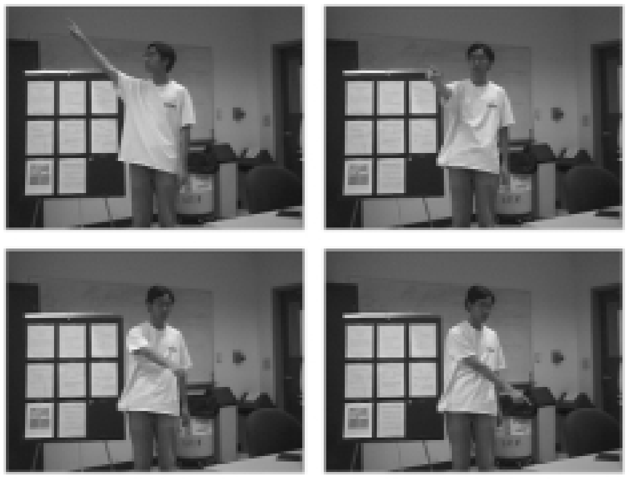

Figure 12: Frames from semi-circle motion

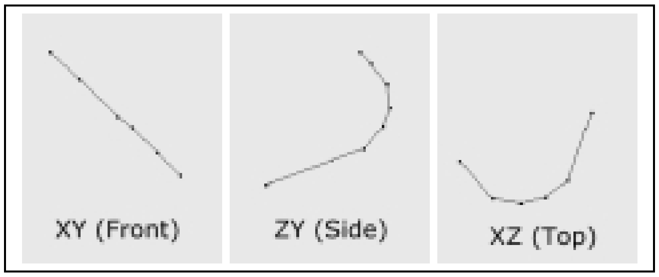

Figure 13: 3D Trajectory output from semi-circle motion

The first parameter was the value of the dot product. Larger dot products signify a larger value of confidence, as high dot products are found when a finger is visible.

The next parameter we look at is the probable occlusion factor. That is, we can use information about how a 3D pose makes finger tracking difficult to improve the 2D tracking. When the finger points towards the camera, it becomes harder to find the finger based on the contour of the arm's skin image. When the finger is pointing in a direction orthogonal to the eye vector of the camera, finger tracking is easier as less of the finger is apparently occluded by the rest of the arm. Thus, the confidence we record will be proportional to the measured length of the user's forearm.

Next, we measure occlusion another way by looking at the number of skin regions we find. During occlusion, the number of skin regions decreases and our estimated confidence also decreases. So, confidence measured will be proportional to the number of skin regions. The last parameter we use is motion disparity. We expect that the motion of the finger will be similar to the motion of the centroid of the arm. If the magnitude of the difference between these two vectors is high, then it is likely the finger was not correctly tracked. Confidence recorded, then, would increase as motion disparity decreases.

Our final estimate of tracking confidence is a linear combination of these factors, with distinct weights for each factor. When the estimated confidence of a certain tracked point is high, we apply very little smoothing to that point. For median levels, we apply normal smoothing, and for low levels of confidence we smooth with a large smoothing factor.

Figure 14 shows an unsmoothed and a smoothed version of the same trajectory. The large dots indicate points at which recorded confidence was low. Looking at the figure to the right, we see that only points of low confidence are smoothed.

As an optional improvement for tracking, the user may choose to extend a thumb while gesturing. In worst cases when the index finger is pointing directly at the camera, it is better to find the thumb than to find another part of the arm. In normal cases, the index finger has a higher curvature so it is tracked properly.

6 Results

Figures 12 and 13 show graphical output from an example data set, where a user moves his hand in a 3D semi-circle motion over a period of several seconds. Figure 12 shows example frames while Figure 13 shows the projection of a 3D trajectory onto several different axes. In the XY view, the points form a straight line because the camera sees the user moving his finger in a similarly straight line. In the other views, we see that the finger's recorded Z coordinates are changing as the finger follows the semi-circle, first moving away from the body, and then returning.

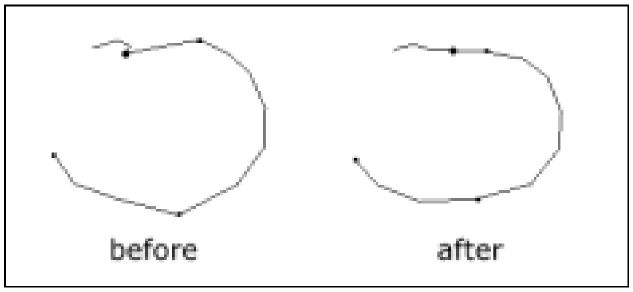

Figure 14: Unsmoothed and smoothed trajectories

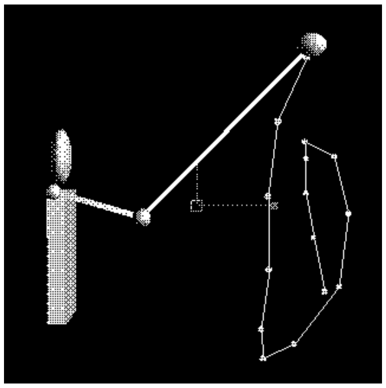

Figure 15: Spiral motion trajectory

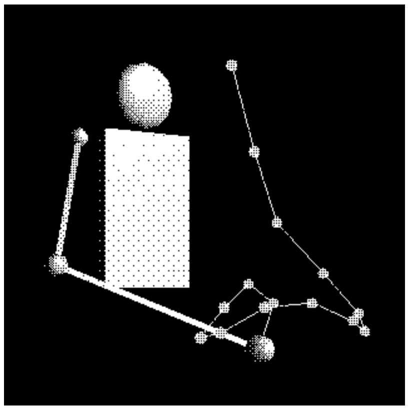

Figure 16: Tetrahedron gesture

In Figures 15 through 18, a true human gesturer's motions are tracked by our algorithm and are shown re-played with a graphical doll. The re-played positions of the trajectory confirm that our 3D tracker is correct. Figure 18 shows a 3D rendering of the doll's resulting path for the 3D trajectory data recorded for the semi-circle motion of Figures 12 and 13. The 3D graphing program uses the actual unmodified parameters derived by our 3D tracker. In this figure, there are more points because the system was given more images to process.

Several other examples are shown in Figures 15, 16 to 17. Figure 17 shows the result of one user's saddle point gesture. Figure 16 shows a 3D trajectory that illustrates the tetrahedral geometry found in chemical compounds such as NH_4.

7 Discussion

Our approach makes several assumptions to make the system faster and less complex. The use of an orthographic projection and the inability to always determine the location of an occluded finger add a

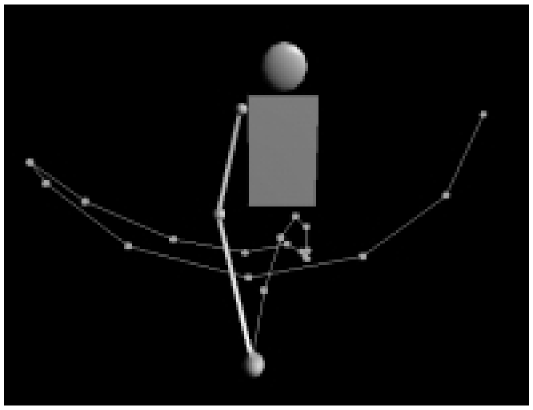

Figure 17: Saddle point motion trajectory (front)

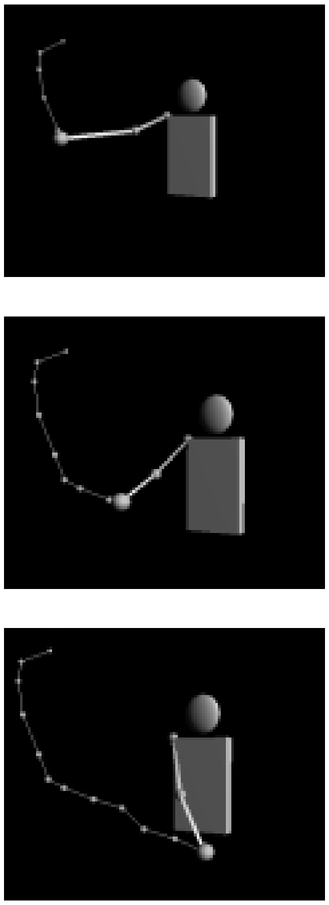

Figure 18: Animation of trajectory in 3D

certain amount of error to the tracking. As such, the tracking is not as precise as other systems that employ more cameras and use a smaller field of view. On the other hand, our system need not be pre-calibrated and can be set up quickly.

The system reported so far could be applied toward many different possible applications. Although it was generally tested on simple 3D geometric motions, the system could just as easily be applied toward recording other types of gestures where relative position is important.

Another possible use for the system would be as a virtual mouse for a very large screen. With some knowledge of the environment, the system would be able to determine what the user is pointing at. In a similar way, the system could be used as a game controller for certain types of fighting and shooting arcade games.

The original intent of the system was to give professors the ability to gesture a 3D geometrical object. We have shown that the system has this capability. Although the system does not automatically classify the gesture, the recorded trajectories and their graphical representations seem useful enough by themselves.

8 Conclusion

We have presented a system that accurately tracks a user's finger in 3D, using only a single camera. Without requiring the user to wear a glove or stand near the camera system, our approach lets the user communicate a 3D geometrical motion to a camera several meters away. This system could easily be used in a teaching environment, or as an intuitive gesture interface at a distance.

References

[1] T. Baudel and M. Beaudouin-Lafon, "Charade: Remote control of objects using free-hand gestures," *CACM*, vol. 36, no. 7, pp. 28-35, 1993.

[2] R. Kjeldsen and J. Kender, "Towards the use of Gesture in Traditional User Interfaces," *Proc. International Conference on Automatic Face and Gesture Recognition*, pp. 151-156. October 1996.

[3] R. Kjeldsen and J. Kender, "'Finding Skin in Color Images," *Proc. Intl. Conf. on Automatic Face and Gesture Recognition*, pp. 312-317. Oct 1996.

[4] C. Maggioni, "GestureComputer - New Ways of Operating a Computer," *Proc. International Conference on Automatic Face and Gesture Recognition*, pp. 166-171. June 1995.

[5] J. Rehg and T. Kanade, "Visual Tracking of High Tracking of High DOF Articulated Structures: An Applation to Human Hand Tracking," *Proc. European Conf. on Comp. Vis.*, pp. 35-46, May 1994.

[6] J. Segen and S. Kumar, "Human-Computer Interaction using Gesture Recognition and 3D Hand Tracking," *Proc. ICIP, Chicago*, pp. 188-192, 1998.

[7] M. Krueger, *Artificial Reality II,* Addison-Wesley Publishing, 1991.