Robust Finger Tracking with Multiple Cameras

Cullen Jennings
University of British Columbia
Vancouver, B.C, Canada
jennings@cs.ubc.ca

This paper gives an overview of a system for robustly tracking the 3D position and orientation of a finger using a few closely spaced cameras. Accurate results are obtained by combining features of stereo range images and color images. This work also provides a design framework for combining multiple sources of information, including stereo range images, color segmentations, shape information and various constraints. This information is used in robust model fitting techniques to track highly over-constrained models of deformable objects: fingers.

Introduction

Natural hand gestures are fairly low bandwidth for most communications but are particularly suitable for indicating spatial relationships. A mouse - a 2 DOF pointer - has proven very useful for HCI; hand gestures could provide 5 DOF or more. Potential applications include robot and human collaboration, virtual reality interfaces, scientific visualizations, GIS, games, controls for machines such as those used in forestry and 3D CAD. Maggioni and Kammerer suggest many potential areas of application, including medicine, advertising, harsh environments, video conferencing and virtual touch screens [19]. As computers become more than screens with keyboards, gestures will likely become increasingly important for user interfaces.

This paper describes a system that tracks the finger of an unknown user in three dimensions against a cluttered background containing motion and variations in lighting. All of these conditions could easily occur in an information kiosk-type application. This research has concentrated on developing a robust system and has used multiple channels of information, including stereo range imaging.

Much of the previous work in tracking and recognition has concentrated on rigid models of non-natural objects with corners and edges. Visual recognition systems have often exploited these features and the assumption of rigidity. Problems often arise in applying this research to natural objects, such as hands, which lack corners or edges, are flexible and deformable, and whose shapes, sizes and colors differ from person to person.

As 3D vision systems acquire more practical utility, the need for robustness becomes increasingly apparent. This system achieves robustness by combining multiple input cues, including stereo images, and using these over-constrained models.

Stereo imaging provides a number of advantages over imaging using only one camera. Range images simplify segmentation, which is one of the hardest problems for robust systems. Range images also stabilize the depth component of 3D positions and help to solve the scale problem, in which an object appears smaller in the image as it moves away.

Previous Work

Gesture recognition has attracted the interest of many researchers in recent years. It is being explored as a novel human-computer interface (a 3D mouse) [13], as a method for directing robots [15] and [4], and even as a replacement for television remote controls [9]. A review is provided by Pavlovic et al. [21].

In most of the literature, hand segmentation has been performed in one of four ways: using a controlled (uncluttered) background [15]; using a known background (i.e., background subtraction) [13]; using segmentation by motion [16]; or using color segmentation [8].

Using controlled or known backgrounds can be problematic in dynamic environments where the background can change over time. Motion cues can be difficult to apply due to the false impression of motion caused by changing light and small camera motions. Color segmentation is a fast and fairly robust approach to hand segmentation that works well under varying lighting conditions and against unknown backgrounds. Unfortunately it can easily be confused if the background behind the hand is close to the color of the skin.

Azarbayejani et al. [1] describe work going on in the Pfinder system. This system uses “blobs” - oval regions of consistent color and texture - as features to track, typically attributing one blob to the head, one to the torso and one to each hand, arm, foot and leg. The Sfinder extension to this system takes tracked blobs from two cameras and gives 3D tracking data. The system seems to rely heavily on using color to segment out the objects to track. It also has to “learn” the background image so that it can be ignored. Pfinder should therefore suffer from many of the problems that cause background subtraction and color tracing to fail.

Tracking has been facilitated through the use of special markers [7], correlation [9] and a combination of color and shape constraints [16].

Lowe’s system for model tracking [18] does edge detection on the image and then fits lines to the edges. These edges are matched to the model edges using a “best first” search approach. The model parameters are then solved for using a weighted least squares approach. The system’s careful approach to errors in matching and measurement allows it to quickly find a good match and to weight the residuals so that it is not dominated by outliers and converges even with large motions. The system is robust and real-time.

One technique that gets away from explicit models is the Eigen image idea. Black and Jepson have developed a system that tracks and recognizes simple hand gestures using what they call “EigenTracking” [2]. This system builds a representation of the hand model from training images and uses this representation to compute an Eigen image set. It then finds and tracks these images in a video stream using a pyramid approach. The major advance of this work is that it uses Eigen techniques to solve for the view transformation of the object at the same time that it finds the object’s motion.

Blake and Isard [3] describe a system whose approach to the tracking problem resembles that in “snakes” but which is based on Kalman filtering. The authors develop a finger and lip tracking system that uses a Kalman filter to estimate coefficients in a B spline. Measurements are made to find the minimum distance to move the spline so that it lies on a maximal gradient portion of the image. These measurements are used as the next input to the Kalman filter. More recent work on combining color blob information with contour information to track hands has used a condensation approach [12].

Rehg and Kanade describe a system called DigitEyes that tracks a hand at 10 Hz, using a 27 DOF model based on cylinders [22]. They describe the implementation of a 3D mouse. The system fits a model to tracked features using a nonlinear least squares fit. The model is updated with a Kalman filter. The finger model is used to find the features in the next frame by searching orthogonally out from the centre of the finger and looking at the gradient of the greyscale image to find the edge of the finger. These contour edges of the fingers are used to establish the current feature location for the finger. The authors point out that difficult problems remain in tracking through occlusions and across complicated backgrounds.

Huttenlocher et al. track flexible objects without using explicit models [10]. First their system forms an edge image. It then matches this to previous edge images using a Hausdorff distance metric. The raw Hausdorff distance is converted into a rank order distance to improve the robustness of the system. One difficulty of a system with no explicit model is that to get the real 3D position of the finger, a model of some sort is still needed.

A similar concept is described by Darrell and Pentland [5]. This system uses view interpolation on training images to do the tracking and recognition. The images are dynamically warped to deal with nonrigid objects like hands. The computation is expensive, but the system achieves 10 Hz performance for small images. The matching of images is done using a correlation that is weighted by the variance of each pixel. More recent work by Darrell et al. make heavy use of stereo and color to track faces [6].

Overview

The overall approach in the system described in this paper is to try to detect the finger several different ways, each method being referred to as a channel and computed by some filter. A model is then simultaneously fit to all the channels. This simultaneous fitting is key. The model fitting is controlled by projecting the model into the measurement space of a particular channel and then comparing it to the measured data from the filter. The channels used are edges from four cameras, range information from stereo correlation, skin-colored regions from one camera, and areas whose convexity resembles a finger tip. There are seven channels in all.

This system is a generalized method that combines sensor readings from several different cameras and types of features. Any system that combines signals from more than one input to make a measurement needs some sort of model that gives meaning to what is being measured.

A very abstract model of a finger is used, which consists of the location of the finger tip in 3D world coordinates and the direction it is pointing. The model is nothing more than the definition of its parameters. The system looks for the set of model parameters that in some sense maximizes the likelihood of all the input channels. This requires an error function to estimate the probability of input, given the real solution specified by the current model parameters. These error functions must take a less abstract view of the model by projecting the model into a space where it can be compared to the current measurement for the specific channel. The combination of all the input channels is based purely on the abstract model.

Tracking can be used to speed up the system, but overall quality largely depends on the capacity to reacquire a lost track and initialize where there is no track. Hands may appear and disappear very quickly. Self occlusion is not addressed by this system.

Data Flow

The overall flow is illustrated in Figure 1, "Information flow in system.," on page 4. First, images are acquired from multiple calibrated cameras. From these images, stereo range images are computed, along with color segmentation images and edge images. A convexity feature that is good at detecting finger tips is also computed. The model fitter then fits the finger model to the data.

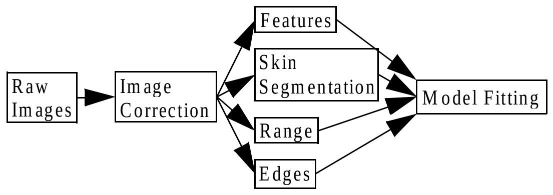

FIGURE 1. Information flow in system.

CAMERA CALIBRATION & IMAGE ACQUISITION

Camera calibration is currently done using a simple system based on a grid of targets with known positions. The Triclops stereo system comes with calibration data.

The system captures images from a color camera and a Triclops camera head with three black and white cameras. A device, described in [14], is used that takes one field from the color camera and then takes the red, green and blue channels for the next field from the three black and white cameras on the Triclops. This multiplexed signal is fed into a single RGB frame grabber. The result is synchronized images from a single RGB frame grabber at field resolution from all the cameras every 30th of a second. The fields are down sampled to 320 x 240 for processing.

The images thus obtained have considerable radial distortion due to the short focal length lenses. This distortion is corrected by using the camera calibration information to resample the image data into an ideal camera model. The image is resampled using an interpolating lookup. Example images are shown in Figure 2, “Raw image and image corrected for distortions.,” on page 5. Note that the straight lines along the ceiling are bent in the uncorrected image.

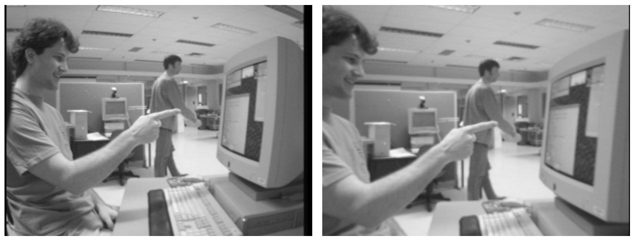

FIGURE 2. **Raw image and image corrected for distortions.**

The four types of channels used by the system for model fitting (range, skin colored regions, convexity features and edges) are described next.

RANGE

The Triclops/TGAP[1] stereo system is used. It has a multi-baseline algorithm that uses a sum of squared differences correlation to compute depth images. The algorithm resembles the multiple-baseline stereo algorithm described by Okutomi and Kanade [20]. Figure 3, “Reference image and depth image.,” on page 5 shows a sample range image and the associated reference image. Closer objects are darker, and black areas indicate that no match was found in the stereo algorithm.

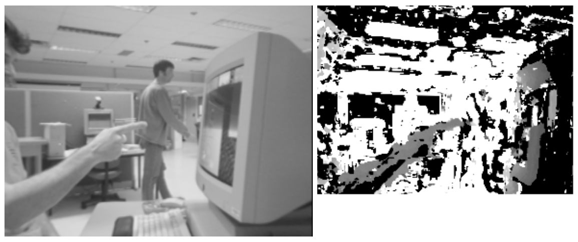

FIGURE 3. **Reference image and depth image.**

1. See www.ptgrey.com

COLOR SEGMENTATION

The color segmentation images are built by considering the hue of the skin and the color intensity. The color segmentation detects the skin reasonably well but often picks up other objects with similar color properties, such as wood furniture and doors. The segmentation is noisy but can be cleaned up with erosions and dilations to the image. Once the initial image is found, it could be used to refine the color search class for subsequent images. A final segmentation is shown in Figure 4, “Image and color segmentation,” on page 6.

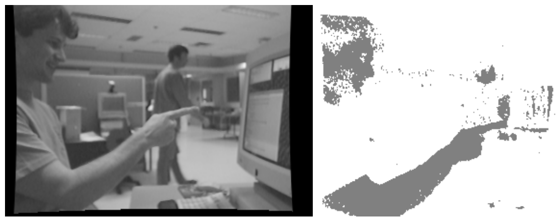

FIGURE 4. **Image and color segmentation**

FINGER CONVEXITY FEATURES

The features stage combines the segmentation cues to form good higher level hints of finger location. A strong geometric constraint is found in the fact that the human fingertip approximates a half sphere that looks the same from most orientations: segmented shapes with a certain sort of convexity are likely to be fingertips. The finger convexity points are computed by taking the skin segmentation (any reasonable segmentation could be used) of the image and finding all the points on the edge of the segmentation that are locally convex. These points are then clustered with their local neighbors to find the convexity points.

EDGES

Edge detection uses the method developed by Shen and Castan [23]. This method provides results similar to the Canny edge detector but is faster to compute. The edge detection is implemented with recursive filters.

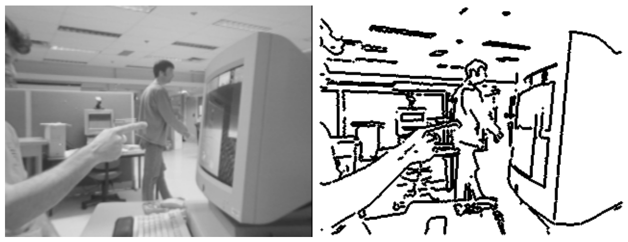

FIGURE 5. **Image and edge detection**

MOTION & BACKGROUND SEGMENTATION NOT USED

Motion segmentation and background subtraction are not used because simple implementations were easily confused by shadows, changing light conditions and small motions or vibrations of the camera.

MODEL COMBINATION

This section shows the derivation of the system's model fitting techniques, which are based on a Bayesian approach to the model and information.

Let the vector m be a model and $Bel(m)$ be the belief that m is the correct model. Let s^t be state at time t of all the sensors and S^t be the set of all measurements up to time t. So $S^t = \{s^t, s^{t-1}, \ldots, s^1\}$. Many different algorithms, or filters, can be applied to any of the sensor readings (images in this case) to implement some measurement. These filters are a function, f_i, that computes $f_i^t = f_i(S^t)$. Define the set of all measurements at the current time as $f^t = \{f_n^t, f_{n-1}^t, \ldots, f_1^t\}$, and then define the set of all measurements up to the current time as $F^t = \{f^t, f^{t-1}, \ldots, f^1\}$.

Our belief that the model is correct after the measurement at time t is

$$Bel_{post}(m) = P(m^t | S^t) \tag{EQ 1}$$

which is the same as

$$Bel_{post}(m) = P(m^t | s^t, S^{t-1}) \tag{EQ 2}$$

Applying Bayes's rule we get

$$Bel_{post}(m) = \frac{P(s^t | m^t, S^{t-1}) P(m^t | S^{t-1})}{P(s^t | S^{t-1})} \tag{EQ 3}$$

Everywhere this equation is used we will be trying to compare different models. Since the denominator is independent of m, it can be considered a constant normalizer. So

$$Bel_{post}(m) = \eta P(s^t | m^t, S^{t-1}) P(m^t | S^{t-1}) \tag{EQ 4}$$

The last term in this expression is just the prior belief in the given model, so

$$Bel_{post}(m) = \eta P(s^t | m^t, S^{t-1}) Bel_{prior}(m^t) \tag{EQ 5}$$

Now making the standard Markov assumption that the given state of the sensors depends on the current model and is independent of previous sensor readings, we get

$$Bel_{post}(m) = \eta P(s^t | m^t) Bel_{prior}(m^t) \tag{EQ 6}$$

The filters, f_i, must together be sufficiently statistic that they preserve the information relevant to the model in the current sensor reading. This sufficiently statistic assumptions results in $P(f^t | m^t) = P(s^t | m^t)$. So

$$Bel_{post}(m) = \eta P(f^t | m^t) Bel_{prior}(m^t) \tag{EQ 7}$$

It should be noted that the measurements are clearly not completely independent, since they come from images of the same scene; but they are not completely correlated either, because each does provide additional information. For simplicity, all the covariance information between the measurements is ignored. This strategy greatly simplifies the problem and amounts to assuming that all the measurements are independent. So

$$P(f_1^t, f_2^t, \ldots, f_n^t | m^t) = P(f_1^t | m^t) \ldots P(f_n^t | m^t) \qquad \textbf{(EQ 8)}$$

Equation 7 thus becomes

$$Bel_{post}(m) = \eta \left[\prod_{i=1}^{n} P(f_i^t | m^t) \right] Bel_{prior}(m^t) \qquad \textbf{(EQ 9)}$$

DEALING WITH THE $Bel_{prior}(m^t)$ TERM

At each time increment in the processing of the input channels, if the previous time step has yielded a position estimate, the system uses a prior that is a constant near where the finger was in the previous time step and zero elsewhere. The vicinity is specified in terms of the model, rather than sensor coordinates, and is considered to be a cube in model space that is 100mm in the x, y and z directions and 70° in rotation. It is centered at each of the likely solutions from the previous time step. When the system lacks an estimate for the previous time step, such as at the first frame or when tracking has been lost, the prior is set to be a constant over the complete feasible model space.

This can be considered a Kalman filter approach. The previous track is projected into the current time frame using trivial dynamics and then diffused to represent the uncertainty in the forward prediction. It is then combined with the new measurements to get a new estimate.

The condensation method of Isard and Blake [11] provides a successful approach to propagating information forward from a previous step. The method deals with the initialization, multiple solution and loss of tracking issues that complicate all tracking systems. It would deal more cleanly with the priors. My initial experiments with condensation approaches suggested that adequately representing the prior distribution would require a number of sample points that would be too huge for computational feasibility in a real time system. These limitations may yet be overcome; future work will explore this further.

DEALING WITH THE $P(f_i^t | m^t)$ TERM

Computing the individual channel probabilities means knowing something about the distribution of the f_i measurements. This distribution is different for each filter function. The parameters required for the distribution are empirically measured in a calibration procedure.

The errors are evaluated by projecting the model into the measurement space, with the projection functions, described in Section , then finding the nearest feature in the channel image. An error metric is formed from a distance metric on this feature and the projection of the model. For the edge images, the error metric is simply the sum of the distances from several sample points around the silhouette of the predicted model to the nearest edge in the image. For the convexity features it is the distance from the projected location of the finger tip to the nearest convexity feature in the image. For the range image, the error metric is the projected range along the center of the finger compared to the range values in the measurement channel. For the color segmentation, it is the number of skin colored pixels that are inside the projected model.

The convexity, range image, and color features produce a fairly Gaussian looking distribution. The error metric of a single edge feature is not Gaussian, so several are added together. The resulting distribution does look Gaussian.

The filter functions have been constructed so that they can be approximated by a normal distribution. One difficulty arises because the tails of a normal distribution are very thin. If a filter detects a feature nowhere near the model, the belief in the model's correctness is not much affected by just how far away the feature actually is. Somehow the normal distribution needs to be modified to reflect this. A mixture of normal distributions could be a good way to model this. This system models it as a probability that the measurement is just plain wrong plus a probability (with normal distribution) of the amount of error in the measurement if it is not plain wrong. The maximum error of the measurement that gets fed into the normal computation is limited, so that, for instance, even if the error is greater than 5σ it will be considered the same as a point at 5σ. The tails of the normal distributions are thus forced to take on constant values at some point. The fact that the distribution no longer integrates to a value of one is taken care of by a normalizer used later.

All the distributions used require a standard deviation. These were found in calibration procedure that was run once when the system was developed. Initially a value of 1.0 was used for all variances; the system easily tracked a finger over a simple background that was not the color of the skin, had few edges and lacked motion. From this sample all the variances for this training run were directly calculated. These newly calculated variances were used for future runs of the program. The algorithm is relatively insensitive to changes in the variance; they can be doubled or halved with little effect on the results.

PROBABILITY REPRESENTATION

The software stores the negative log of the probability instead of the actual probability. This causes equation Equation 9 to become

$$\begin{aligned} -\log(Bel_{post}(m)) &= \\ -\log(\eta) - \log(Bel_{prior}(m^t)) &+ \sum_{i=1}^{n} -\log(P(f_i^t | m^t)) \end{aligned} \quad \textbf{(EQ 10)}$$

Another function projects the model into the measurement space to provide an expected value of the function. This projection function is called μ_i. For a filter function with a normal distribution, this gives

$$P(f_i^t | m^t) = \frac{1}{\sqrt{2\pi\sigma_i}} e^{-\left[\frac{(f_i^t - \mu_i(m^t))^2}{\sigma_i^2}\right]} \quad \textbf{(EQ 11)}$$

The leading term is independent of m^t so it can be replaced by a normalizer, resulting in

$$-\log(P(f_i^t|m^t)) = \eta_i + \left(\frac{f_i^t}{\sigma_i^2} - \frac{\mu_i(m^t)}{\sigma_i^2}\right)^2 \qquad \textbf{(EQ 12)}$$

Equation 10 can now be written in the form

$$-\log(Bel_{post}(m)) = \\ \eta - \log(Bel_{prior}(m^t)) + \sum_{i=1}^{n}\left(\frac{f_i^t}{\sigma_i^2} - \frac{\mu_i(m^t)}{\sigma_i^2}\right)^2 \qquad \textbf{(EQ 13)}$$

Finding the model that maximizes our belief that it is the correct model is equivalent to minimizing

$$-\log(Bel_{prior}(m^t)) + \sum_{i=1}^{n}\left(\frac{f_i^t}{\sigma_i^2} - \frac{\mu_i(m^t)}{\sigma_i^2}\right)^2 \qquad \textbf{(EQ 14)}$$

This is almost in the form of a least squares problem, except for the $\log(Bel_{prior}(m^t))$ term. Since $Bel_{prior}(m^t)$ is constant in a feasible region and zero outside it, the $\log(Bel_{prior}(m^t))$ term constrains the solution to be inside the feasible region but has no other effect. This constraint is embedded into a least squares solver that finds a solution to

$$\sum_{i=1}^{n}\left(\frac{f_i^t}{\sigma_i^2} - \frac{\mu_i(m^t)}{\sigma_i^2}\right)^2 \qquad \textbf{(EQ 15)}$$

The stabilization methods described by Lowe [17] are used in solving this least squares problem. These stabilization methods provide a trade-off between moving the current solution as little as possible and reducing the overall error in the model fit. Constraints can be added to the least squares solution in two different ways. One is to simply add a linear equation representing the constraint to the system of equations being solved. The other is to set the partial derivatives to zero outside the feasible region and rely on the stabilization terms to keep the solution from moving around randomly when the solution is not in the feasible region. This second method is used here to constrain the search to the feasible configuration space.

The problem is nonlinear because the μ_i projection functions are nonlinear. This nonlinearity means that the solution will only converge on the global minimum for some neighborhood in the feasible model space. The program must search the feasible space to find the global solution to Equation 15. The size of the convergence neighborhood determines how closely the search points need to be located. Empirical results show that the system easily converges over a 10° rotation and 15mm motion, resulting in a very coarse sampling of the search space that runs quickly.

TRACKING & SEARCHING

One the problems with tracking fingers is that they move very quickly. Most tracking systems allow the track to be no more than a few pixels off the predicted motion from one frame to the next. This just does not work for fingers. Most people can easily move their fingers at 4 m/s; a baseball pitcher can reach 10 times that speed. The accelerations are also very impressive. Even at 30 fps, the motion will be several times the diameter of the finger. The model from the previous frame may not even overlap with the finger in the current frame. Unless the frame rate is much higher than 30 fps, no reasonable upper limits on the finger's acceleration or velocity can confine the possible location of the finger to a small area. For this system, the finger can move up to 5 cm in any direction between two frames and can change its orientation by 35°. If no finger has been found in the previous frame, the whole work space is searched. The radius of convergence for the model is about half the diameter of the finger. This makes sense: if the model is not on the finger, it is difficult to match up the correct features and do a least squares fit that gets closer to the correct model. This means that I step across the search space (normally 100 mm by 100 mm by 100 mm by 35°) in steps of 15 mm and 10°. Fortunately each model is quick to evaluate.

The model evaluation consists of doing a least squares fit on the model to solve Equation 15. The system numerically computes partial derivatives of each error metric with respect to each degree of freedom in the abstract model. An important twist here, however, increases the robustness of the whole system. The conditional operabilities, $P(f_i^t \mid m^t)$, for each channel are examined. The two least likely channels are ignored and not used for forming the equation to be solved in the least squares fit. This simple-minded approach to robust statistics works well and greatly improves the performance of the system. Often one of the channels is completely wrong; this process helps to keep a wrong channel from corrupting the end result.

The partial derivatives are put into a matrix to solve Equation 15 for the least squares solutions. Lowe's stabilization method [17] is highly advantageous here: if the system lacks constraints in some direction, the solution does not move all over the place. The least squares solution usually converges after only a few iterations. Constraints can be embedded in the system simply by adding additional equations to the matrix to be solved.

At the end of this search stage, each model has a certain likelihood of correctness. Models below a certain threshold (10%) are thrown out, and a non-maximal suppression step is applied so that if two models have converged to the same solution, only one is kept.

Results

ROBUSTNESS

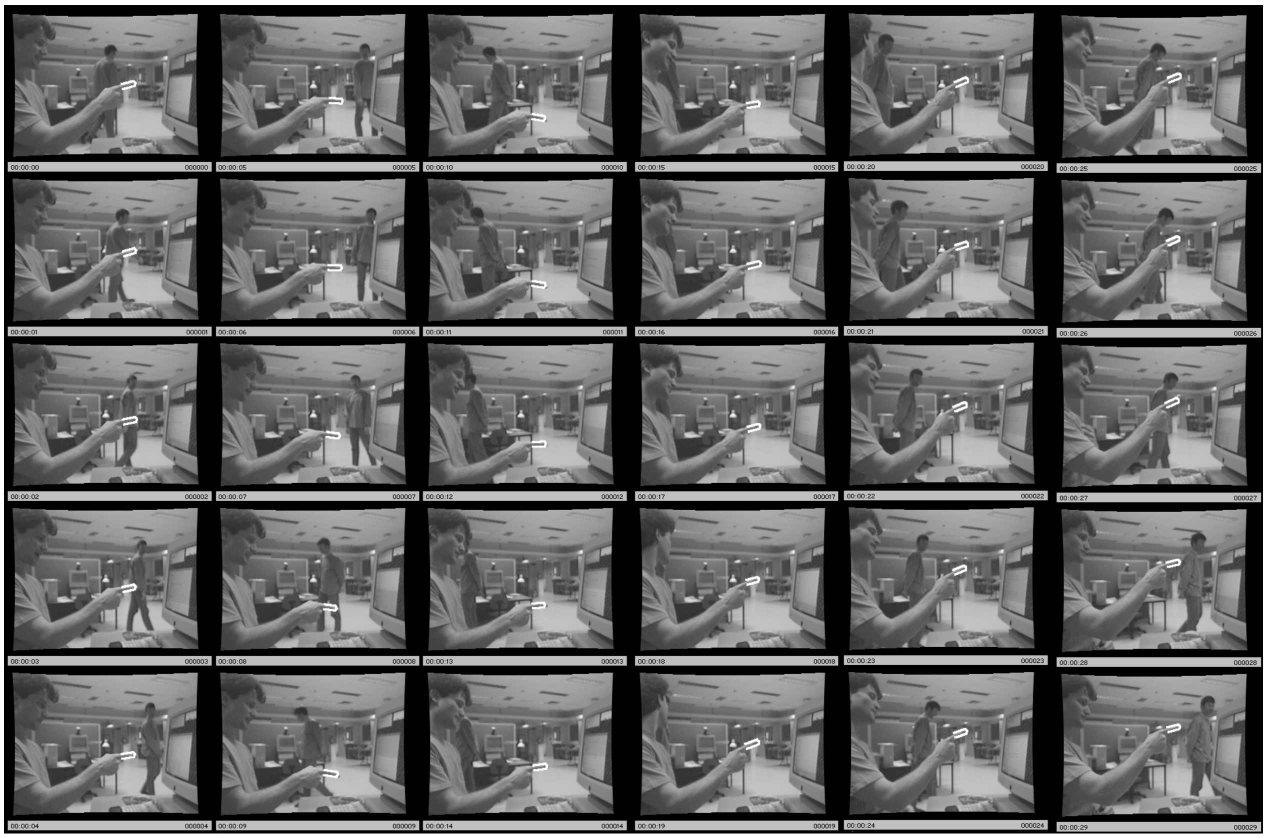

FIGURE 6. **Image sequence with tracks (view downward and then left to right)**

Figure 7, “Difficult image (all four camera views),” on page 13 shows a challenging image from a sequence containing a cluttered background with many edges and a person walking around. The door directly behind the finger is wood and shows up as the same color as the skin. As shown, the system correctly tracks the finger. (A large number of images from the sequence are shown in Figure 6, “Image sequence with tracks (view downward and then left to right),” on page 12**.)**

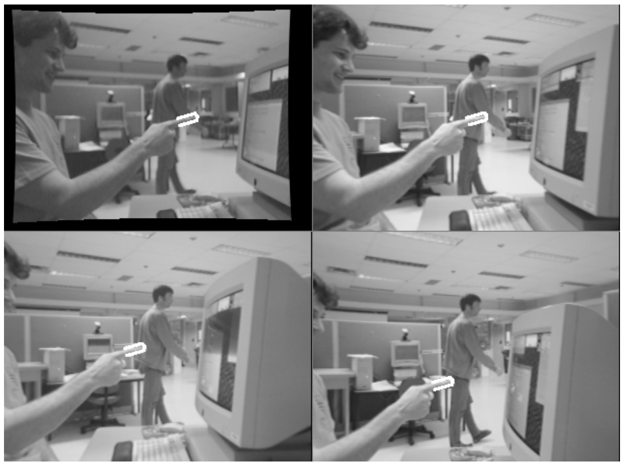

FIGURE 7. **Difficult image (all four camera views)**

The system has also been tested under changing ambient lighting conditions and where the camera moves. Neither situation causes any problems, as is expected because the system does not carry any state information about lighting from one frame to the next. Similarly, as long as the camera motion does not cause an apparent motion of the finger that is larger than the track region, there is no state information about the finger. This system will therefore work even if the cameras are sitting on a desk that vibrates. On the other hand, in the situation in Figure 7, "Difficult image (all four camera views)," on page 13, a shift in the main color of the monitor, which casts a considerable amount of light on the finger, will drastically change the lighting and color consistency of the finger. The system has been tested with screen colors of white, red, blue, green and black in a setting with about 50% of normal ambient light. The skin detection colors are set wide enough that this does not cause any problems.

The system also works well where there is no texture for stereo detection. In these cases, the range information is poor but the edge information tends to be good because of the absence of background clutter.

The model finding for the initialization sequence takes about three times as long to compute as a normal tracking frame, but it is important that this be robust and reasonably easy to compute because a finger can disappear from a scene at any time and must be quickly reacquired when it reappears. The initialization model fit was run on each frame and compared with the tracked results. In almost all cases the result was the same.

The two possible error types are that the system may find the wrong thing or may find nothing even though there is a finger in the scene. These errors are basically controlled by setting a single probability threshold for throwing out models. This test was run with a threshold that ignored any solution with a likelihood of less than 5%. No errors of either type occurred.

The system performs poorly in two situations. The first is when the finger is occluded by the arm or hand. Occlusion is possible because the cameras are mounted close together. In future work this problem may be solved by mounting the cameras so that they view the scene from very different angles. The other failure case is when the finger points almost straight at the cameras. This could be dealt with by positioning the cameras to the side of the user, so that such an action is awkward or, again, separating the cameras so that a user could not point to them all at once.

ACCURACY

The system was tested with the hand moving over a volume 500mm high, 500mm wide and 800mm deep, where the depth axis moves away from the cameras. This volume corresponds approximately to the work space in front of a computer monitor. A large number of hand rotations were tested in this volume. Ground truth was gathered by putting a ruler in the scene, moving the finger to a particular point on the ruler and taking measurements. The distance between two measurements from the system was compared with the ground measurements from the ruler.

This method of measuring ground truth data is not especially accurate - the ground truth is probably accurate only to 5mm. The accuracy in the plane of the camera image is quite good, usually less than the 5mm limit of the ground truth data. Occasionally the system does not track all the way out to the end of the finger and causes a finger tip displacement of up to 10mm. No data in the test sequence exceeded 15mm and almost all the data was under 5mm.

On the depth axis (moving away from the cameras) the accuracy is worse. The average error was 25 mm while the maximum error was 40 mm. Given the baseline of the cameras, the 40mm error represents an error of about 1.5 pixels in the image. It will be hard to improve this beyond about a 0.5 pixel error; much better depth resolution will require a wider baseline or higher resolution images. It is difficult to obtain ground truth for the rotation angles. Examining the model overlaid on the images reveals that none of the rotation angles exceed 5°.

Given the size of the work space, the system's accuracy is about 5% for the worst case data and about 3% for most of the frames. If the depth component is ignored, the average accuracy is around 1%.

SPEED

The system was run on a Pentium Celeron 400 Mhz processor running Linux. The time varies up to 10% for a given frame, but the average time per frame is shown in Table 1, "Average computation time per frame," on page 14. Frames where the track is lost and the system must reinitialize the tracking take three times as long.

TABLE 1. **Average computation time per frame**

Task	**Time (ms)**
Capture frame	40
Scaling, unwarping, and image correction	80
Stereo processing	80
Edge detection	25
Convexity features	10
Total	**300**

TABLE 1. **Average computation time per frame**

Task	**Time (ms)**
Skin detection	15
Model fitting and search	15
Miscellaneous	35
Total	**300**

The current speed of about 3 fps is not great real time, but with some tuning and a dual 500Mhz Pentium III processor, the system could probably achieve 15 fps.

If the frame rate doubles, the distance the finger can move between frames halves, and the volume of the search space diminishes by a factor of 8. Each frame requires much less computation. Because of the large overhead in capturing, unwarping frames and stereo processing, the system does take more CPU cycles as the frame rate increases, but the relationship is less than linear.

Conclusion & Future Work

This paper has demonstrated a highly robust system that can accurately track a finger in three dimensions. It can deal with motion in the camera and with complex backgrounds containing motion, skin colored objects and many edges. The same techniques can be applied to a wide range of tracking problems. The system deals quickly with the initialization case and recovering when tracking has been lost. The technique is suitable for real time systems.

Future plans include investigating how to update parameters, such as variance, as the system runs. Skin color, for example, could be measured in the current frame once the hand was found and could then be used for finding similarly colored regions in subsequent frames. Using condensation offers important possibilities.

References

[1] A. Azarbayejani, C. Wren, and A. Pentland. “Real-Time 3-D Tracking of the Human Body”, M.I.T Media Lab. Tech. Report #374. Appears in Proc. IMACE’COM 96, Bordeaux France, May 1996.

[2] M. Black and A. Jepson. “EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation”, ECCV’96.

[3] A. Blake and M. Isard. “3D position, attitude and shape input using video tracking of hands and lips”, SIGRAPH’94.

[4] R. Cipolla and N.J. Hollinghurst. “A human-robot interface using pointing with uncalibrated stereo vision”, in Computer Vision for Human-Machine Interaction. Ed. R. Cipolla and A. Pentland. Cambridge University Press, 1988.

[5] T. Darrell and A. Pentland. “Space-Time Gestures”, CVPR’93, pp. 335-340.

[6] T. Darrell, G. Gordon. J. Woodfill, M. Harville, “A Virtual Mirror Interface using Real-time Robust Face Tracking”, Proceedings of the Third International Conference on Automatic Face and Gesture Recognition, April 1998, Nara, Japan.

[7] J. Davis and M. Shah. “Visual Gesture Recognition”, IEEE Proc. Vision Image Signal Processing 141(2), April 1994, pp. 101-106.

[8] M. Fleck, D, Forsyth, and C. Breggler. “Finding Naked People”, ECCV’96, April 15-18, 1996, pp. 593-602.

[9] W. T. Freeman and C. D. Weissman. “Television control by hand gestures”, IEEE International Workshop on Automatic Face and Gesture Recognition, June 1995.

[10] D. Huttenlocher, J. Noa, and W. Rucklidge. “Tracking Non-Rigid Objects in Complex Scenes”, Cornell University Computer Science Dept. Tech Report, CUCS TR-92-1320.

[11] M. Isard and A. Blake, “Contour tracking by stochastic propagation of conditional density”, ECCV-96, 1996, pp. 343-356.

[12] M. Isard and A. Blake, “Icondensation: Unifying Low-Level and High-Level Tracking in a Stochastic Framework”, ECCV-98, 1998, pp. 89 908.

[13] K. Ishibuchi, H. Takemura, and F. Kishino. “Real Time Hand Gesture Recognition using 3D Prediction Model”, International Conference on Systems, Man, and Cybernetics, Volume 5, Le Touquet, France, October, 1993, pp. 324-328.

[14] C. Jennings. “Video Switch”, Circuit Cellar Ink, No 5, April 1999, pp. 32-38.

[15] A. Katkere, E. Hunter, D. Kuramura, J. Schlenzig, S. Moezzi, and R. Jain. “ROBOGEST: Telepresence using Hand Gestures”, Technical report VCL-94-104, Visual Computing Laboratory, University of California, San Diego, December 1994.

[16] I. J. Ko and H. I. Choi. “Extracting the hand region with the aid of a tracking facility”, Electronic Letters 32(17), August 1996, pp. 1561- 1563.

[17] D. Lowe. “Fitting Parmeterized 3-D Models to Images”, PAMI 13(5), May 1991, pp. 441-450.

[18] D. Lowe. “Robust Model-based Motion Tracking Through the Integration of Search and Estimation”, IJCV 8(2), 1992, pp. 113-122.

[19] C. Maggioni and B. Kammerer. “GestureComputer - history, design and applications”, in Computer Vision for Human-Machine Interaction. Ed. R. Cipolla and A. Pentland. Cambridge University Press, 1988.

[20] M. Okutomi and T. Kanade. “A Multiple-Baseline Stereo”, PAMI 15(4), April 1993, pp. 355-363.

[21] V. Pavlovic, R. Sharma, and T. Huang. “Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review”, PAMI 19(7), July 1997, pp. 677-695.

[22] J. Rehg and T. Kanade. “DigitEyes: Vision-Based Human Hand Tracking”, Technical report CMU-CS-93-220, School of Computer Science, Carnegie Mellon University, December 1993. Also see ECCV’94.

[23] J. Shen and S. Castan. “An optimal linear operator for step edge detection”, GVGIP 54(2), March 1992, pp. 112-133.