PROBABILISTIC EVIDENCE COMBINATION FOR

ROBUST REAL TIME FINGER RECOGNITION AND TRACKING

BY

CULLEN FRISHMAN JENNINGS

B.Sc., The University of Calgary, 1990

M.Sc., The University of Calgary, 1993

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming to the required standard

.....

.....

.....

.....

THE UNIVERSITY OF BRITISH COLUMBIA

June 2002

Abstract

This thesis sets out a Bayesian approach to the robust combination of measurements from multiple sensors in different measurement spaces. Classical least squares optimization is used inside a sequential Monte Carlo approach to find the most likely local estimate. The local optimization speeds up the system, while the Monte Carlo approach improves robustness in finding the globally optimal solution. Models are simultaneously fit to all the sensor data. A statistical approach is taken to determine when inputs are failing and should be ignored.

To demonstrate the overall approach described in this thesis, the 3D position and orientation of highly over-constrained models of deformable objects - fingers - are tracked. Accurate results are obtained by combining features of color and stereo range images. The multiple sources of information combined in this work include stereo range images, color segmentations, shape information and various constraints. The system is accurate and robust; it can continue to work even when one of the sources of information is completely failing. The system is practical in that it works in real time and can deal with complex moving backgrounds that have many edges, changing lighting, and other real world vision challenges.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I owe thanks to a number of people. First, to David Lowe, who has provided ideas, encouragement, and guidance throughout. Alan Mackworth and Michael Black have provided valuable input especially on theoretical aspects of the work. Nando de Freitas provided a missing piece by helping me pull together the diverse types of particle filtering techniques. I have benefited as well from the broad perspectives of Dinesh Pai and Jim Little, the latter of whom has also always been a reliable source of help when resources needed attention.

Don Murray has been an invaluable sounding board on a range of ideas from the earliest days of this work. Rod Barman, Vlad Tucakov and Stewart Kingdom have kept the lights on and the Digiclops running. I am profoundly grateful to Lyndsay Campbell for her support, assistance and tolerance throughout. Pamela Campbell, Dominique Jennings, and Mary Ann Jennings have consistently provided encouragement and interest in estimated completion dates.

Luan Dang has gone to bat to ensure that I would have at least some of the time this project required. The National Sciences and Engineering Research Council of Canada provided a grant in the early years, which was much appreciated. And, of course, Naomi Jennings lent me her test equipment, as shown in Figure 52.

1. Introduction

As 3D vision systems acquire more practical utility, the need for robustness becomes increasingly apparent. Real world vision systems must consistently succeed under a wide variety of conditions. One approach for achieving this robustness is to combine multiple input cues. Another approach is to combine measurements from many different sensors. However, combining this information can be complicated by contradictory data. As well, the data is often difficult to combine because it comes from very different sorts of spaces. For instance, one measurement might be a simple yes or no answer to the question “is there a person in the image?” Another measurement might be the 2D location of the person’s head in the image, and a third might give the 3D location of the person in the world space. It is difficult to determine how best to combine these measurements to decide whether or not there is a person in the room. This work sets out a Bayesian approach for the robust combination of measurements from multiple sensors in different measurement spaces.

One of the techniques that has proven useful in many vision problems is segmentation. Using multiple cues, such as stereo vision, motion, and color, provides solutions to some of the difficult segmentation cases that cannot be solved by any single technique. In particular, stereo imaging has a number of advantages over imaging using only one camera. The range images from stereo simplify segmentation, and they can also stabilize the depth component of 3D positions. The system described takes advantage of multiple cameras, stereo range, and multiple cues for segmentation to help achieve robustness.

1.1 Problem Domain

This research has concentrated on developing an approach to creating robust systems and has used multiple channels of information, including stereo range imaging. A system has been developed which 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.

Natural hand gestures are fairly low bandwidth for most communications but are particularly suitable for indicating spatial relationships. A mouse - a two DOF (Degrees Of Freedom) pointer - has proven very useful for HCI (Human Computer Interface); hand gestures could provide five DOF or more. Potential applications include robot and human collaboration, virtual reality interfaces, scientific visualizations, GIS (Graphical Informations Systems), games, 3D CAD (Computer Assisted Design), and controls for machines such as those used in forestry. As computers become ubiquous and more than screens with keyboards, gestures will likely become increasingly important for user interfaces.

There are a growing number of applications through which people wish to view and manipulate 3D information. CAD, GIS, and scientific visualization are some examples. Mice make a difficult user interface for moving around view points and objects in these systems. Users might more easily use a finger on each hand as a kind of 3D mouse, and combine these with stereo viewing glasses to control these systems. A person would move both fingers in unison to perform a translation or rotation and independently to suggest scale. Similar interfaces have been investigated using data gloves, but the gloves have not been convenient to calibrate and use. These difficulties have led to things like the “Magic Board” system [10], which uses vision to detect the location of a finger that is acting as a pointer on a white board for augmented workplaces.

Games represent a large market for software. The Sony PS2 (Play Station 2) has sold over six million units in North America with the average unit selling over four hundred dollars worth of games. Logitech sales in 2001 were in the range of half a billion dollars, and a large portion of this was input peripherals for games. Vision could be used in many possible ways to allow the player to act out the game, instead of just push buttons on the game controller. It is easy to imagine the desire to use an interface such as this for games. Recent games like Dance Revolution require a custom dance pad to monitor the location of players' feet; in this game, you actually dance. In a vision based version, the whole body could be monitored. Vision-based, one-on-one combat games could involve tracking virtual swords: this would be very similar to the finger tracking described in this work.

In terms of usefulness to society, at the other end of the spectrum from game applications are medical ones. Computers are increasingly common in operating rooms. Currently, sterilization requirements mean that computers have to be kept in plexiglass boxes and disposable mice are used. A vision-based interface that permitted pointing would not require contact and could replace the mouse. Likewise, in applications where security necessitates similar arrangements, the user could interact in a very natural way, by pointing at a computer behind thick glass in a kiosk.

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. Additionally, it is unusual for these objects to move very many pixels between frames, so past motion can be used to predict the motion in the next frame. Here, the nature of the problem is quite different. The objects are smooth, and edges arise out of their silhouettes, not from any hard corners on the object. The objects are deformable and non-rigid, and they can easily move 100 pixels between two frames. Fingers' shapes, sizes and colors differ from person to person. Past frames give only a loosely constrained prediction of where fingers will be in the next

frame. The lighting conditions can be changing, the background can be cluttered, and it may contain motion. These conditions make for a complicated, challenging problem that is fairly characteristic of real world vision systems.

This system concentrates on robustness to changing lighting conditions, different skin colors, and backgrounds with motion and complex distractions. It provides accurate 3D measurements of the finger location in real time. It deals with the tracking of the finger and the initialization problem: finding the finger initially and finding it again when the tracking gets lost.

This work has two key dimensions: it describes an approach for combining multiple measurements for vision systems, and it shows a complete real time vision system for tracking fingers that demonstrates this robust approach.

1.2 System Overview

One of the key research items addressed is this work is how to combine very different measurements in a robust way. In general, robustness is achieved through the use of multiple input cues from multiple images. A model is simultaneously fit to all the input cues. A Bayesian approach is taken to combine the cues and determine what the best model is. This reduces to a least squares approach to find the locally optimal model. A search is performed over the model space to find the globally optimal model. Constraints can easily be introduced: for example, to implement a tracking approach, the search space can be constrained to the near vicinity of the image in a previous frame, once an initial model has been found.

1.2.1 Information flow

The overall flow is illustrated in Figure 1. What then follows is a synopsis of the system, the details of which are elaborated in subsequent chapters.

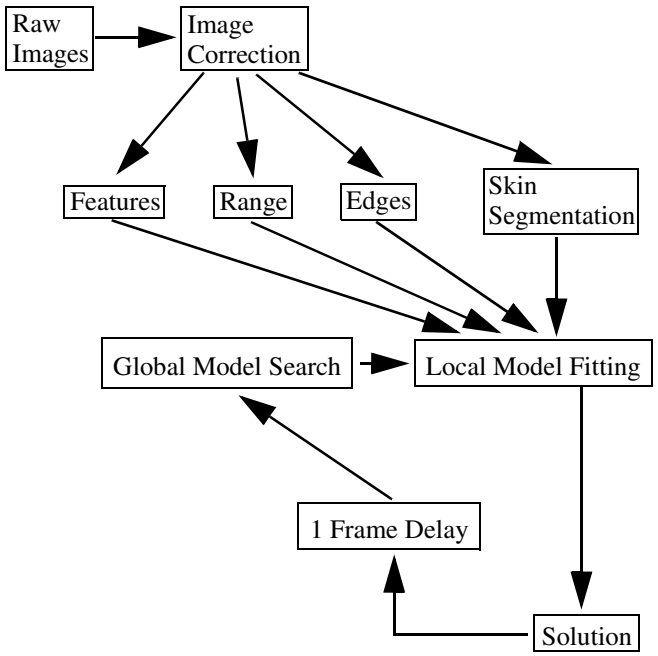

FIGURE 1. Information flow in system

1.2.2 Using multiple cues

The first stage is the gathering of the cues, a process described in more detail in Chapter 3. The raw images come from three cameras, which provide the following cues: edges, skin colored regions, and stereo range. Using multiple cues is important for achieving robustness. If only edges were used, the system would fail against a background with many edges. If only skin colored regions were used, the system would fail when it encountered backgrounds with similar colors, such as cardboard boxes or wood furniture. If image flow were used, the system would become confused upon encountering a scene with rapidly changing light, such as where light shines through a fan. Something will confuse almost any sensor, but less often will all the sensors be confused at the same time, although this will still

happen if, for example, all the lights go out. Each one of these cues is referred to as a channel and is computed by some filter.

1.2.3 Stereo range images

Range images are helpful in 3D vision systems. One challenge for such systems is deciding whether an object is small and nearby or large and far away. A related problem is telling if the camera has rotated or translated by a small amount when objects are some distance away. Range images can provide a simple way to get an initial estimate for many of these problems and also stabilize the solution.

The stereo range information helps reduce the dimensionality of the search space. Range sensors can be confused by lack of texture or the "picket fence" problem. They can, however, provide a useful way to do segmentation of images, being robust to lighting conditions and camera motion, which confuse many segmentation schemes. Range images also increase robustness in situations where the traditional background subtraction methods would be difficult to use, being difficult to initialize.

One of the goals of this research is to demonstrate how stereo can be used for both segmentation and 3D vision problems.

1.2.4 Error estimates

Combining various measurements requires error estimates for each. Imagine having one input sensor that provides latitude but gives a more or less random longitude, and another sensor that gives longitude but provides a random latitude. Only

by understanding the error characteristics of these sensors and the values they are giving can one can obtain the correct solution. Examine the following diagram :

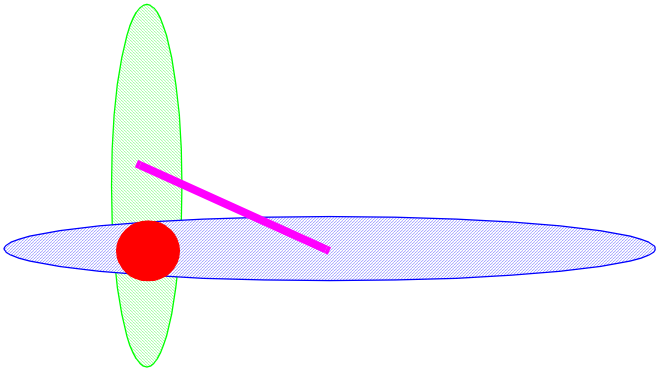

FIGURE 2. Data fusion with error knowledge

One sensor is modeled by the green ellipse and the other by the blue. If the error metrics are invoked and a model is fit simultaneously to both, the system will come up with the solid red circle as the solution. If instead each is solved in turn and then the solutions are combined without regard for the error characteristics, the solution will lie somewhere along the magenta line, which is incorrect.

Identifying the nature of error is very helpful in combining measurements. Data fusion schemes based on voting or consensus are unlikely to work in these sorts of cases.

1.2.5 Model fitting

Chapter 4 concentrates on the derivation of the model fitting and how it is locally and globally optimized. 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. A Bayesian approach is taken to determine which

model is the best fit to all the measurements. A key point of this fitting is that all the channels are fit simultaneously.

.

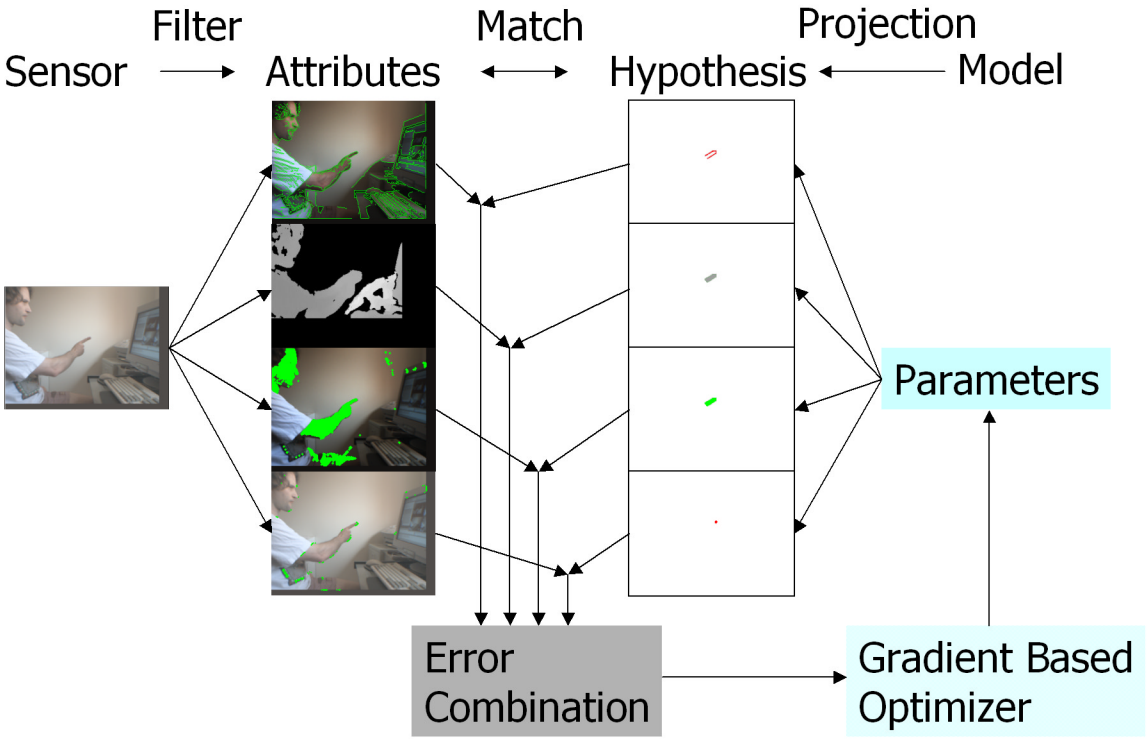

FIGURE 3. Simultaneous fitting of model to multiple measurements

This concept is illustrated in Figure 3. All the sensors have various filters, such as edge detection, applied to them to form an attribute. The model is projected into a space that can be compared with each type of attribute - this is referred to as a hypothesis. The hypothesis and attributes are matched to come up with some probabilistic rating of how well the sensor data supports the proposed model. All of the errors are combined into a single evaluation of how good the model is, which is put into an optimizer. The key here is that all the various attributes are simultaneously fit to the model, instead of each being individually fit to it in sequence.

The fitting of the model to all the measurements can be transformed into a least squares type of problem. A robust model fitting approach is taken to optimize a single model from some start location to a local minimum in the model parameter space. This process is fairly fast and converges over a reasonably wide

range of initial positions. At first glance, requiring a model might seem like a limitation of the technique, but 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. The measurement is a parameter to the model and only makes sense in the context of that model. A model does not have to be a complete physical model of some system - it can be highly abstract. In this work, an 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.

The second half of Chapter 4 describes the searching and tracking that is done subsequently to find the globally optimal model fit. Finding the globally optimal model still requires iterations over the complete search space. Two approaches were implemented. The first, a grid based method, evenly and systematically distributes the model across the whole parameter space. The distance between the models is made small enough that there will be at least one model in each local optimally basin. Each model is then fit to the locally optimal solution, and then the best of all the local solutions is chosen as the global optimum.

The second method is almost the same but instead of evenly distributing the model, a fixed number of models are randomly distributed. This approach is somewhat better for real time systems. The number of models can be chosen such that the system will compute the answer in the available time. If a faster CPU is available, the system will adapt to use more models and will automatically get better results with no modification of the program. On the other hand, if the CPU

is so slow that the system cannot search enough models to adequately cover the search space, it may fail to find the solution on one frame; but on the next frame it will search a different portion of the search space and might find the globally optimal solution. With each frame processed, the odds that the globally optimal solution has been found in at least one of the processed frames goes up.

Once an initial solution is found in one frame from the search method, it can be used to help find the solution in the next frame. A particularly effective approach has been to take the solution from one frame and, in the next frame, constrain the search space to within 10 cm or 45° of this location. The problem is then treated more or less like the previous search problem.

This second method naturally leads to a sequential Monte Carlo (SMC) approach and was adapted to a variant of particle filtering. The models are particles and can be used to estimate the probabilities. This technique improved the propagation of information from one frame to the next.

Techniques that use velocity to predict the location of the finger in the next frame turn out to work very poorly. The problem is that hands can move, change direction, appear and disappear very quickly relative to images gathered at typical camera frame rates. Once an initial model is found, 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.

Chapter 6 shows the results of the system as run on several series of images. The system is evaluated in terms of its speed and accuracy under varying conditions, and the results are very promising. Chapter 6 gives conclusions from this work and points to avenues for future work.

1.3 Key Contributions

The key contributions made through this work include:

- adapting SMC approaches to converge to modes of probability distributions;
- showing the importance of simultaneously fitting the model to all the data;
- when fitting multiple measurements, taking into consideration the types of errors that exist;
- demonstrating the increase in robustness provided by range images; and
- showing the engineering for a complete vision system.

(This page intentionally left blank.)

2. Previous Work

The past twenty years or so have produced a large body of work on problems that are, or can be, associated with human motion tracking. In surveying the state of the field, Moeslund and Granum [49] create a taxonomy of systems that sorts them according to their contributions to the development of four processes: initialization, tracking, pose estimation, and recognition. The authors observe, “we are far from a general solution to the human motion capture problem”; as key issues requiring work, they identify initialization, recovery from failure, and robustness. In a similar survey, Gravila [19] has grouped systems into 2D approaches with and without explicit shape models, and 3D approaches. Outstanding problems mentioned include initialization, occlusion, modeling, the treatment of error, and the shortage of reliance on 3D data, especially range data. In a third, earlier review of hand gesture research, Pavlovic et al. [52] identified shortcomings in the current work, focusing particularly on two different modeling strategies (3D and appearance-based). These authors found that the state of the art does not allow for the unambiguous interpretation of human gestures. They considered that 3D models showed more promise than appearance-based ones, but computational complexity was a significant problem. The field, they concluded, was still in its infancy.

In human gesture recognition, although much work has been done, much also remains. To attempt to address the various problems that have been identified in the field, this work draws on many areas of computer vision. Results are applied from research on sensors, stereo, multiple views segmentation, matching, learning, shape representation, model fitting, tracking, object recognition, video analysis, color consistency, vision systems, camera calibration, particle filtering, Bayesian approaches, robust approaches, gesture recognition, real time vision, Kalman filtering, least squares model fitting, sequential Monte Carlo, robust statistics and Markov models. The current state of the art is the work of the efforts

and insights of many. In this chapter I set out the key papers and methodologies pertaining to the development of these various techniques. The inspirations for the ideas in this thesis emerge from these.

2.1 Gesture Recognition

Gesture recognition has attracted the interest of many researchers in recent years. Maggioni and Kammerer [46] and Crowley et al. [10] emphasize the importance of vision-based gesture recognition within the larger HCI picture. It is being explored as a novel human-computer interface (a 3D mouse) [29], as a method for directing robots [37][8], and even as a replacement for television remote controls [18]. Suggested applications have included digital desks, augmented work spaces and augmented white boards [39]. Effective gesture recognition would make it possible to track a user's activities, sense the presence of people and things, or, of course, enhance video surveillance. It would also be appealing to be able to interact with real objects as input to a computer, instead of virtual objects on an electronic desktop: imagine editing a paper document on your desk while the writing is tracked and the electronic version of the document is automatically updated (see Wellner [75]).

Some of the early work on gesture recognition goes back to Badler and Smoliar [3], while one of the earliest systems involving a model driven analysis then synthesis approach similar to this one was described by Hogg [23].

Moeslund and Granum make an interesting observation in their recent review article on the tracking of full people [49]. They observe that of the few systems that deal with the initialization problem, most use multiple cameras. Similarly, Gravilla comments, "[i]t is fair to say that the results of vision-based 3-D tracking are still limited at this point. Few examples of 3-D pose recovery on real data exist in the literature and most of these introduce simplifications ... or limita-

tions ... that still require improvement with respect to robustness. Robust 3-D tracking results have been particularly scarce for approaches using only one camera. The benefits of using multiple cameras to achieve tighter 3-D pose recovery has been quite evident" ([19], page 23).

2.2 Color Segmentation

In most of the literature, hand segmentation has been performed in one of four ways: using a controlled (uncluttered) background [37]; using a known background (i.e., background subtraction) [29]; using segmentation by motion [38]; or using color segmentation [17]. It is not always possible to use a controlled or known background: many environments are dynamic and have backgrounds that 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. The work described here builds on the successes and learns from the limitations of these various techniques, one of the most useful of which is color segmentation.

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. A good survey on face detection that includes much of the work on skin detection is by Yang et al. [82]. Jones and Rehg [33] took a huge sample set of images and marked the skin regions so as to assess the distributions of skin color. Because the images were from non-calibrated cameras, the results are not directly applicable here, but they provide a good insight into the shape of the color distribution. A serious look at why skin is colored the way it is with the actual spectrophotometric data for skin across some races in provided in [1].

The work of Raja, McKenna, and Gong [54] on adapting the color distribution when tracking people highlights some of the difficulties in this area. Their approach uses a mixture of gaussians in the color space and slowly evolves. They use EM to fit the mean and covariance of the gaussians to the data set. The results are not great under conditions involving rapid motion or lighting changes.

Future work could extend the work in this thesis using an approach similar to the work by Sigal et al. on dynamically updating color distribution [66]. They use a second order Markov model to predict the motions of the skin color histogram in HSV space over time. Markov models are developed using maximum likelihood estimation. These authors used a 64x64x64 RGB to HSV table, which I found to lack sufficient resolution to distinguish some colors that should be skin (or not skin). Their system also has problems if the initial skin histogram contains pixels that have been erroneously classified as skin. A related problem made dynamic color updates difficult in this work: as "bad" colors leaked into the skin classification, the results degraded rapidly. The authors build their initial classification using a static classification drawn from the hand labeled data set collected by Jones and Regh [33].

2.2.1 Infrared

Several systems (e.g. [59]) have used thermal infrared images as an alternative to color segmentation. Thermal infrared would likely form an excellent alternative input for the work in this thesis. The evidence it provides is quite different from the information gathered by the other cameras and is likely to fail at different times, which would make it complementary. Detecting skin by combining measurements made in normal color space with infrared measurements seems like it would be an interesting approach that is as yet not used in the people tracking literature.

2.3 Tracking Systems

Tracking has been facilitated through the use of special markers [14], correlation [18] and a combination of color and shape constraints [38]. The work described in this thesis combines the findings and tracking attempts of a large number of other authors.

2.3.1 Color blobs

Azarbayejani et al. [2] describe work using 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 tracking to fail.

Several systems have concentrated on where the user points. The Perseus system [34] uses a combination of edge, range, color, and motion inputs to track an arm well enough to tell where someone is pointing in 2D.

The work of Jojic et al. [32] takes advantage of stereo range images to help with background segmentation and then fits a gaussian model to the range data to track where a user is pointing. The range data does a good job of background subtraction for textured backgrounds, as it is not confused by shadows and changing lighting conditions.

A fairly simplistic tracking system that uses only skin color to find the 3D location of an arm is described by Wu et al. [78]. An interesting aspect of this system is its ability to do 3D tracking from a single camera by assuming the length of a user's arm.

2.3.2 Image based approaches

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” [4]. 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.

Huttenlocher et al. track flexible objects without using explicit models [24]. 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 [12]. 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. [13][11] makes heavy use of stereo and color to track faces. The combination of stereo and color gives excellent results.

The work of Toyama and Hager [71] takes several simple, image-based techniques, uses multiple simultaneous trackers, and then applies a sensor fusion approach to get a robust tracking system. The individual trackers are all computed over small windows so that the system can run in real time. This paper identifies the issues that make it desirable to fuse various trackers.

2.3.3 Snakes

Blake and Isard describe a system whose approach to the tracking problem resembles that in "snakes" but which is based on Kalman filtering [5]. The authors develop a finger and lip tracking system that uses a Kalman filter to estimate coefficients of 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 [27] on combining color blob information with contour information to track hands has used a condensation approach (see section 2.4), which improves the results.

2.3.4 Model fitting with least squares

Lowe's system for model tracking [41][42] 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. A very similar scheme is used in the least squares fitting in this work (see Chapter 4). Lowe's stabilization approach clarified how to get the least squares model fitting to converge without excessively moving the models.

Rehg and Kanade describe a system called DigitEyes that tracks a hand at 10 Hz, using a 27 DOF model based on cylinders [56]. They describe the implementation of a 3D mouse. The system fits a model to tracked features using a non-linear 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 center 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.

Kakadiaris and Metaxas [36][35] describe a system that models a human torso and uses the projected silhouette of the 3D model to fit with the image data from multiple cameras. The results are very accurate 3D positions. By having the cameras view the person from very different views, the system can cope with complex occlusions. The authors' decision to use only silhouettes introduces problems with separating the background from the person, but that is not the focus of their research.

2.3.5 Kalman Filter Approaches

Many tracking systems have made use of Kalman filters in various form. Generalizations of these led to the particle filtering and SMC approaches discussed later in this chapter. One classic approach is to run a Kalman filter on the pose estimates of the model being tracked. Harris [21] discusses a system that does this. It uses edges to match to a known model of a rigid object. Systems taking this sort of approach have been very successful at tracking well defined geometric shapes that have nice hard edges, such as aircraft and tanks. These systems have been most successful when there have been few objects to track and limited motion between frames.

A fairly different type of tracking based on Kalman filters appears in other work by Harris [20], in which many features are tracked simultaneously and each has its own Kalman filter. In this system, the corner features of many stationary objects are tracked, and the system attempts to derive the 3D geometry of the scene and ego motion of the camera. One of the difficulties with these types of systems is knowing when to remove tracks and when to introduce new ones. This way of thinking about problems led to the invention of particle filtering approaches that are a form of SMC.

2.3.6 Learning

Some other systems have taken a training approach. Heap and Hogg [22] describe a system that tracks a hand. They use a deformable point distribution model of a hand captured from MRI data. Real time tracking is obtained with some degrees of freedom in the hand. The work of Rosales et al. [58] uses a large training set of images, along with ground truth captured with a data glove, to treat recognizing hand poses as a learning problem.

2.3.7 Graph matching

The work of Triesch and von der Malsburg [72][73] examines the difficulties of real world situations, compared to the images used in much of gesture research. They use flexible graph matching to track fingers and hands and to try to obtain reasonable results in real world situations. The underlying features used are Gabor jets and variants of them that they call Color Gabor jets. These are formed into bunch graphs, which are fed into a classic elastic graph matching algorithm. The approach is different from the one in this work but the results are very good, and it works in images with complex backgrounds.

2.3.8 Bayesian approaches to tracking

There is a developing body of literature on using a Bayesian approach to derive a model fit and combining this with some sampling method, as is done here. Much of the work in this thesis was originally published in 1999 [30]. Werman and Keren [76] have taken a similar approach to fitting circles and line segments to data points.

Though not directly related to vision, Thrun's work on the localization of mobile robots was a source of inspiration for the ideas and probabilistic formulations for the localization of fingers described in this thesis (see [70]). The work

here follows the Bayesian formulation of the problem in Thrun's work, and his incremental localization algorithm led to ideas about combining sensor readings in a meaningful way.

Sherrah and Gong [61] use this sort of Bayesian formulation to track hands and faces. Cham and Rehg [6] use it to track a human limbs model of a person. They also have a multiple hypothesis approach that leads well into particle filtering.

2.4 Sequential Monte Carlo

The traditional SMC and algorithms closely related to it have shown up in the vision community under several different names, including particle filtering, condensation, temporal Bayesian filtering, Monte Carlo filters, and the bootstrap algorithm. Iba [25] provides a good survey of sequential Monte Carlo (SMC) techniques and tries to unify some of the various names from different disciplines that have developed similar algorithms. Doucet et al. have detailed the algorithm in a good overview paper [16]. Convergence issues are well covered by Crisan, who covers different ways of viewing the formulation of the algorithm and describes briefly its origins and how it has been used in different fields [9].

The condensation method of Isard and Blake [26] got many people in the vision community interested in SMC. Condensation is a successful approach to propagating information forward from a previous step. The method deals with certain situations that challenge all tracking systems: initialization, multiple solutions and losing the track.

MacCormick and his colleagues [43][45][28] have done substantial research on tracking with particle filtering methods, including tracking fingers. Isard and MacCormick [28] use particle filtering with blobs to track people. This

would likely also work reasonably well in tracking hands. The resulting tracking is fairly fast and robust.

Wu and Huang take a particle filtering approach to track heads with ellipses [80] and, with Lin [81], take a similar approach to hands. This provides better results than earlier work [79] that was purely least median squares.

The work of Sidenbladh, Black, and Fleet [64] on tracking people walking takes an approach very similar to the Bayesian formulation of the problem used in this thesis. In addition to motion, which was used in [64], other work by Sidenbladh and Black [63] added ridges and the edges of limbs as features for computing likelihoods. Related work by Sidenbladh, Torre, and Black [65] uses principal component analysis to learn a visual model for the appearance of each limb. More recent work by Sidenbladh and Black [63] takes Bayesian tracking and particle filtering and adds a learned model that describes the likelihood that edges and ridges are being generated by people being tracked or by the background. All of this work is covered in more detail by Sidenbladh [62].

2.4.1 Unscented SMC

The unscented particle filter algorithms [48][47][57] provide some important insights into how it may be possible to integrate adjustments of the particles into the system. In traditional particle filtering, the choice of where to pick the samples ignores the current measurements. This can result in samples being placed in locations that are very unlikely, given the current measurements. Unscented SMC looks at techniques to take into account more information, including the current measurements. It can result in SMC techniques that require fewer particles to still correctly approximate the probability density function. In different papers, Van der Merwe et al. have provided both an excellent brief introduction to the ideas [47] and substantially more detail [48]. Rui and Chen [57] show how to use unscented particle filtering for head tracking using an oval head model that is fit to Canny edges in the image. The authors show that a better distribution of parti-

cles results in better tracking than does the traditional condensation algorithm. Applying these approaches to vision is very recent in the research, and it is not clear exactly how best to do it, but it is clear that doing something with these approaches can improve on traditional particle filtering.

2.4.2 Combining least squares and particle filtering

The approach taken in this thesis - simultaneously fitting the model to all the inputs - is similar to the work of Rasmussen and Hager [55]. These authors start their model fitting with points in the state space sampled from the prior distribution. They use a gradient approach to drive the samples toward a more likely solution. Thresholds are then used to select the best samples. The inputs they use are homogeneous regions (oriented rectangles of constant color), textured regions (areas that are matched with sum of squared differences correlation), and snakes-based Sobel edges. They use this to track heads, arms, and chess pieces. The key focus of the work is dealing with two objects that overlap and occlude one another. The authors define a Joint Probabilistic Data Association Filter that results in masks in the sensor space so that a given bit of pixel data is associated with one of the objects and the occluded area does not distract the tracking of the other object. They show that these types of approaches can be used in a variety of tracking problems. Similarly, the work of Sullivan and Rittscher [69] to track heads also combines a local optimization with particle filtering. However, neither the work of Sullivan and Rittscher [69] nor that of Rasmussen and Hager [55] deals appropriately with correcting the importance weighting. This is discussed in more detail in section 4.12.

The work in this thesis is similar to that of Rasmussen and Hager [55] in the overall use of particle filters and optimization, but their system does not deal with the errors model of the sensors that helps in combining them in a robust way. Some training data about the sensors' measurements is needed.

2.4.3 Modified SMC

Other work that combines a traditional optimization technique with particle filtering is the work with annealed particles by Deutscher et al. [15]. This work uses an annealing schedule to change the amount of random noise applied in the particle filters as they slowly narrow the peaks of the fitness function. This allowed the system to successfully track objects in a higher dimensional space than is possible with classical particle filtering, with significantly fewer particles. They track a full body model of a person walking with 30 degrees of freedom.

Choo and Fleet [7] have combined Markov chains with the sampling step in SMC to improve the convergence in high dimensional cases. They tracked a 28 DOF body model with it. This is a very large number of DOF to deal with using particle filtering, but this work shows that it is possible in some constrained cases.

2.5 Robust Approaches for Vision

There is not a huge amount of work applying the results from robust statistics to computer vision problems. The best survey I have found is by Stewart [67]. Some common approaches that have been applied included Least Median Squares (LMS) and variants of it, as well as various M-estimators including the Huber, Cauchy, and Beaton and Tukey. Two approaches developed in the vision community are Hough transforms and RANSAC. The work in this thesis is closely related to the M-estimator type of approaches. The M-estimators apply an importance function that reduces the importance of outliers and then usually does some sort of maximum likelihood estimation. In this work a sensor model is chosen which results in an importance function that reduces the influence of outliers.

2.6 Advancing the Field

As the previous sections suggest, there has been much interest in recent years in issues pertaining to gesture recognition and tracking people. Sophisticated methods for tracking, dealing with outliers, and accumulating evidence over multiple frames have developed that have made it possible to attempt to do robust real-time tracking in complex environments. This thesis combines the various types of input for tracking - such as color, stereo range, edges, and image appearance - with the model fitting approach to tracking and the statistical techniques from Bayesian work, Kalman filtering, and SMC.

In general, state-of-the-art systems are described as able to do tracking in specific cases, instead of being able, more generally, to work almost all the time in almost all cases. The work in this thesis takes on this situation, in the context of a robust, practical, real world system, by looking at how to combine evidence from multiple sensors over time so as to find the best answers given the evidence.

2.6.1 Open Problems

Techniques are required for addressing the many significant problems still awaiting good solutions, such as:

- modeling flexible, deformable 3D objects and acquiring the parameters for a given object;
- initializing at the start of a sequence;
- tracking;
- occlusion;
- realizing that other instances of the model may appear or disappear;
- achieving robustness with respect to the environment;
- making systems real-time and affordable; and
- analyzing and understanding errors in a system.

3. Sources of Evidence

Overall, the system works by using multiple cues to detect the finger's location and then robustly combining these cues and fitting a finger model to the data, using a Bayesian approach to select the best model. This chapter discusses the cues used and how they are computed. Chapter 4 describes how a model is fit simultaneously to all the channels to provide a model that is locally optimal, and then how a globally optimal model is determined. The information flow is illustrated in Figure 4

.

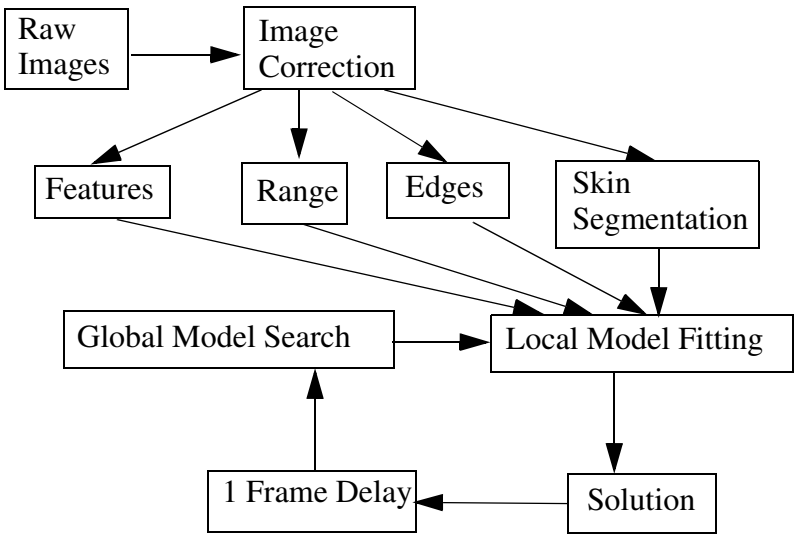

FIGURE 4. Information flow in system

3.1 Image Acquisition & Camera Calibration

The system captures images from a color Digiclops camera made by Point Grey Research and shown in Figure 5.The Digiclops has three cameras which each cap-

ture a 640 x 480 progressive scan color image at over 15 frames per second and send it via a firewire interface to the host computer.

FIGURE 5. Digiclops

Although the actual CCD arrays have 640 x 480 pixels, the pixels are sampled on a Bayer's grid. This grid has a checkerboard-like color filter, as shown in Figure 6.

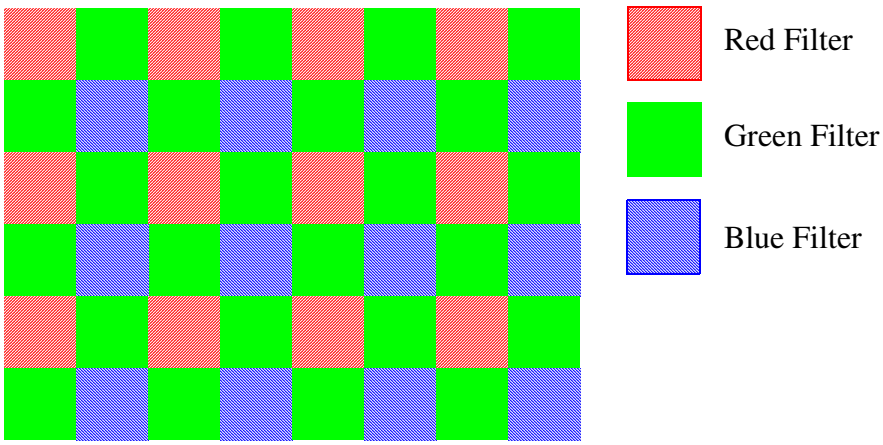

FIGURE 6. Bayer filter

There are only 320 x 240 red pixels and a similar number of blue pixels. There are twice as many green pixels, so in theory the resolution of the color image is somewhat higher than 320 x 240 but much less than 640 x 480. It seems like it should be possible to get better results by taking advantage of the higher resolution of the

green channel. However, simplistic efforts to do this did not seem to make much difference to the accuracy or resolution of the whole system, compared to just using the sample 320 x 240 images; and the system ran slightly faster with the 320 x 240 images, so these were used.

The image data from the camera is never put into an NTSC format, which would substantially compress the color information. Compared to the images from a typical camera, therefore, Digiclops images have less noise in the color hue space and fewer color artifacts. Some raw images are shown in Figure 7.

FIGURE 7. Left and right raw image

This version of the Digiclops does not do automatic white balance. Because color is actively used, the system needs to be calibrated whenever there is a change in the lighting conditions under which it is operating.

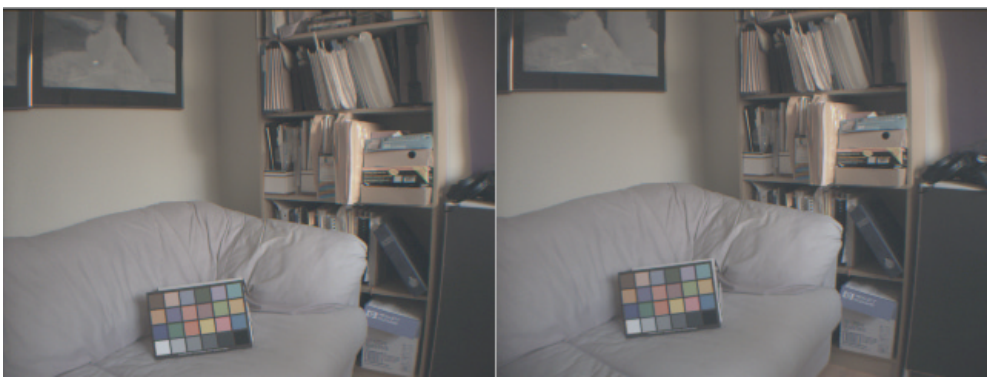

FIGURE 8. Left and right image after color correction

This is done by holding up a 18% gray card and adjusting the linear gains on each color channel until the card is gray in the RGB space. The resulting color corrected images are shown in Figure 8. In producing this thesis, no attempt was made to calibrate the color of the printing process because this was not seen as fundamental to the ideas presented.

The images obtained have considerable radial distortion due to the short focal length lenses. This distortion is corrected by using the camera calibration information to re-sample the image data into an ideal camera model. The image is re-sampled using an interpolating lookup. To reduce the alias effects in the resampling, super sampling is used. (A good discussion of warping, interpolating, and resampling is found in [77].) When the area covered by one pixel in the destination image is mapped onto the source image, its area is still roughly one pixel and it is not so distorted that it will touch more than four pixels on the source image. Super sampling by a factor of nine will therefore provide excellent results. Other values were tried, but four is too low and values above nine gave essentially the same results.

When resizing images, various interpolation filters were tried. Simply using a nearest neighbor approach resulted in poor images and reduced the accuracy of the results. For the system to achieve sub pixel accuracy, image sampling is important. The sinc filter is ideal, and it was the metric to which other results were compared, but it was quite slow. The lanczos2 is a good approximation to it that is faster to compute. An integer approximation to the lanczos2, known as the Gabriel decimator, was a bit faster than the floating point lanczos2 filter. The results it gave had the same accuracy as the sinc filter, and since it was faster it is used for image downsizes. All of these filters are described in detail in [74].

Sample images after this correction are shown in Figure 9. Note that the straight lines along the bookcase are bent in the uncorrected image. At the same

time the geometry of the images is adjusted so that the epipolar lines align with the rows or columns of the images.

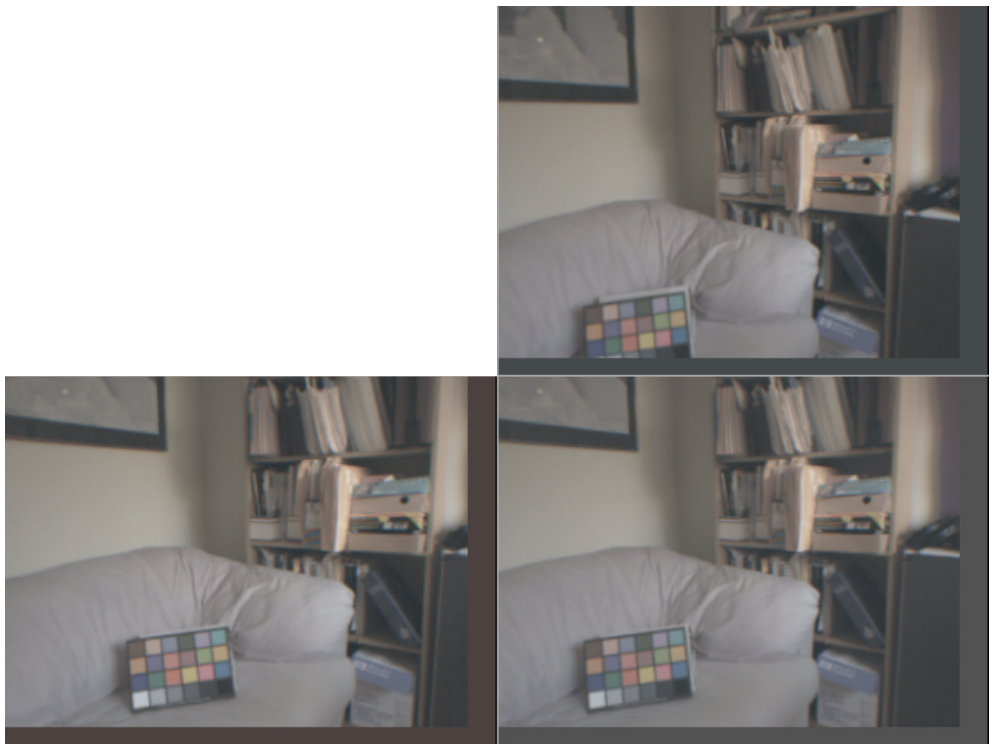

FIGURE 9. All images corrected for color and lens distortion

3.2 Image Channels

As noted, much of the system's robustness comes from its use of four types of channels: range, skin colored regions, convexity features, and edges. These are described next.

3.2.1 Range evidence

The Triclops stereo system is used to compute range images. 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 [51]. Figure 10 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. Traditionally range has been displayed by having the grayscale value represent the disparity between the two images. Black is infinity, and white is very close. This practice results in a very non-linear scheme that makes the noise in distant measurements appear much less than it is. I represent range images by making the grayscale value be the distance from the camera to the object, in centimeters. Objects immediately in front of the camera are dark, and objects 2.5 metres away are white. I found that this linear representation made it easier to visualize the position and angle of the finger relative to the camera.

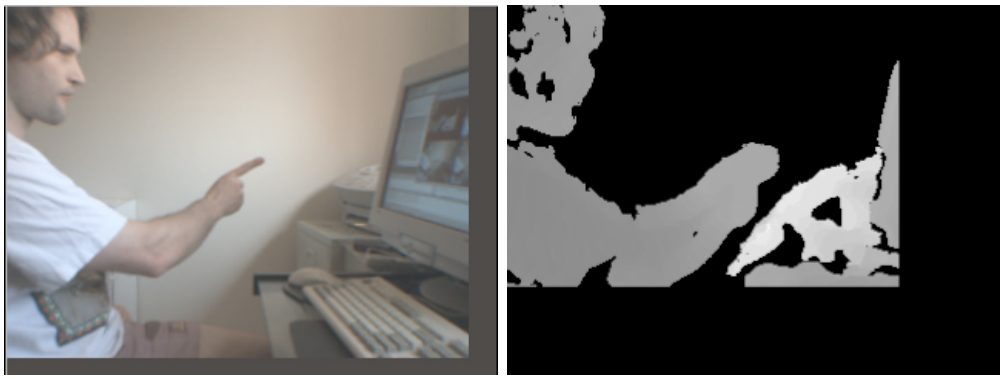

FIGURE 10. Reference image and depth image

The range data from this system is quite good compared to other real time stereo systems, but it tends to have more noise than non-real time test sequences used in other systems for people tracking, such as the work of Lin [40].

3.2.2 Color evidence

The color segmentation images are built by finding skin colored image regions. This can work well because the camera has already been white balanced to compensate for the current lighting. The color segmentation detects the skin reason-

ably 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.

Skin colored regions are detected by comparing the color of the pixel to a set of known reference skin colors. The comparison is done in a CIE Lab color space, and the distance between the pixel color and the nearest color in the reference set is used to form a probability that this pixel represents a skin colored region.

The probability is computed by assuming that the distance from the reference skin colors forms a gaussian distribution. The standard deviation of this distribution is learned in the training sequence in the same way as for all the other channels. (This training step is discussed in more detail in section 4.10.) The reference set was constructed by hand selecting some skin colored pixels from various images, and then varying their intensity in an HSV color space to generate a few extra skin colors. A better approach to selecting reference colors would be to pick colors that cover the space found in the huge skin color study done by Jones and Rehg [33].

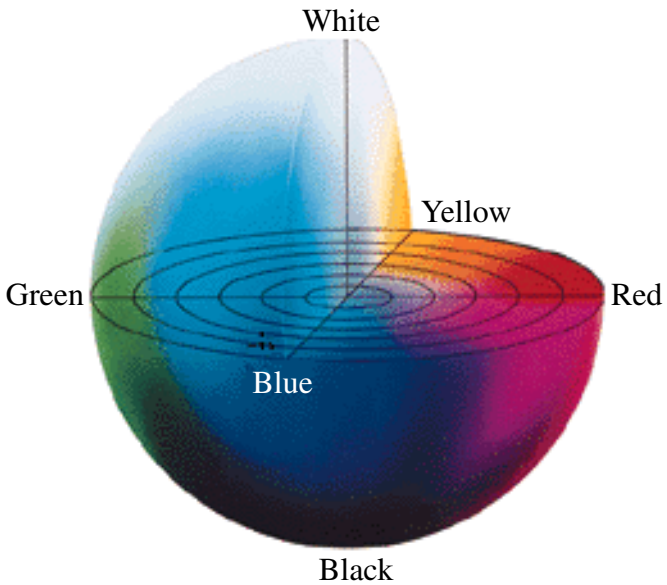

FIGURE 11. CIE Lab color space

The Lab space is perceptually uniform. This means that the distance between two colors in the space is proportional to the rough human perception of how different the colors are. This may not be the best space for this application, but it is much better than RGB for performing relative comparisons of the difference between colors. Because it is based on human perception, it is fairly good in the region of colors that represent skin tones, as people are good at distinguishing these. Figure 11 represents the color space. The vertical axis, l, shows luminance, the a axis represents the green/red component, and the b axis represents the blue/yellow component.

The probability image formed from the distances in the color space is segmented into a binary image of skin colored regions. Additionally, an edge detector is run on this skin probability image to find likely edges of skin colored regions.

After the skin segmentation is done, an erosion and dilation step of one pixel is performed to clean up the speckle noise. The result is shown in Figure 12.

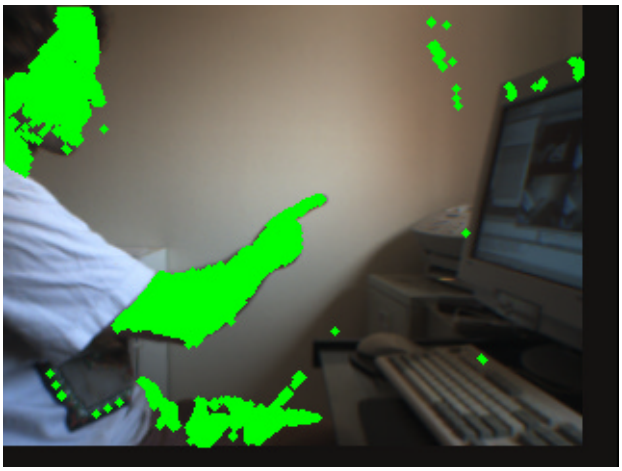

FIGURE 12. Image with color segmentation

Once the initial image is found, it could be used to refine the color search class for subsequent images - this is not done in the current system but would likely improve the results in some situations. Some work was done in attempting to refine the color class on-line, but the difficult problem that arose was slow corruption of the color class in long image sequences. At first, the detection results

would start to improve as the system narrowed the color class for the current user and lighting conditions, but as time went on and the occasional noise pixel snuck in, the color class would get corrupted and start to include things that were not skin colored. Abrupt changes in lighting also cause significant problems for on-line refinement of the color class.

3.2.3 Convexity evidence

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. Clearly this feature will not be as widely applicable as range, color, and edges and will work well only for certain types of tracking problems, such as fingers. I have not seen it in other finger tracking work and believe it is a contribution to the field.

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. In computing convexity, first the edges in the grayscale image (see section 3.2.4) and the skin segmentation (see section 3.2.2) are found. All the edge pixels that are within two pixels of a skin colored region are found. This is done by dilating the skin image twice and performing an "and" operation with the edge image. These edge locations that are close to the skin colored regions are candidates to be convex points. They are on an edge near a skin colored region. Each candidate point is checked to see if it is convex. This is done by looking for any other skin colored pixels on lines radiating out from the candidate pixels. Sixteen lines, each 22.5° apart and 16 pixels long, are radiated out. The number of consecutive lines that have no skin

colored pixels on them are counted. If nine or more of the lines are clear, then the candidate pixel is considered convex.

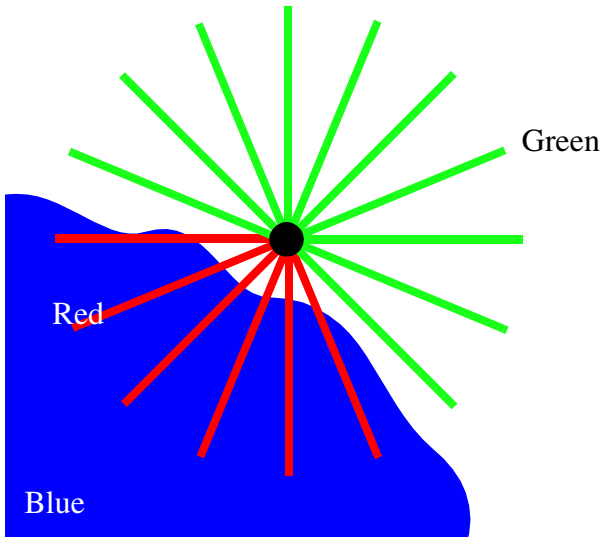

FIGURE 13. Convex point calculation

An illustration of this process is shown in Figure 13. The sixteen lines are shown radiating from the black candidate point in the center. The blue region represents a skin colored region. The lines that intersect it are colored red and the ones that do not are green. Ten consecutive lines do not intersect the skin colored region, so this candidate point would be considered convex. An example result is shown in Figure 14.

FIGURE 14. Convexity points

3.2.4 Edge evidence

Edge detection uses the method developed by Shen and Castan [60]. This method provides results similar to the Canny edge detector but is faster to compute. The edge detection is implemented with recursive filters. It is run on the grayscale images formed from the color images. Edge images from each of the three cameras is used as an separate input channel.

FIGURE 15. Image and edge detection

3.2.5 Oriented edge evidence

Oriented edges help the models converge from much farther in the search space. The concept is fairly simple: by trying to match, say, the edge that corresponds to the top of the finger only to those edges that have their skin colored side below them, we reduce the number of edges that need to be matched, and we ensure that the model does not latch onto a local minimum and place the top edge of the finger model on the bottom edge of the finger in the image.

The oriented edges are computed by looking at the eight directional derivatives formed by the Sobel operators. The operator that returns the largest result for a particular location is considered the normal for the edge. This gives a rough edge direction that covers a 45° sector.

For each of the eight directions, we can now use a distance transform algorithm to form a distance map for each pixel, which shows the perpendicular distance to the nearest edge. Another map is made to keep track of the location of the edge pixel that is closest to any given location. Since the end algorithm spends substantial time finding the edge nearest to a given model location, this map can greatly speed up the matching procedures that are used later in the system.

3.3 Finger Model

The model of the finger used is simply the position of the finger tip and the direction the finger is pointing. This model is very abstract - it does not say anything about what a finger must look like. It is the simplest model that could capture all the information the system gathers. In future work the model could also include non-geometric quantities like the current skin color.

The ability to compare this model to a measurement from one of the channels comes from the projection function, which projects the model into a space where it can be compared with a measurement. This comparison needs to be able to return a probability that the model is correct, given the measurement. Furthermore, the projection function needs to be constructed such that the distribution of this probability can be approximated by a normal distribution or a mixture of a normal distribution and a uniform distribution. A uniform distribution helps deal with the problem of outliers that plagues many normal distribution systems. It prevents the tails of the distribution from getting too small.

The next chapter describes the model fitting, which uses the basic input cues just described in this chapter. To recap, these inputs are:

- Range images from stereo correlation matching from three cameras;
- Skin colored regions from three cameras;
- Edges from three cameras; and
- Convexity feature points from one camera.

4. Probabilistic Model Fitting

This chapter shows the derivation of the system's model fitting techniques, which are based on a Bayesian approach to the model and information. The formal derivation that leads to the algorithm begins in section 4.2. In section 4.1, I provide, informally, some simplified, one-dimensional examples to introduce and provide the motivation for some of the ideas that are then described beginning in section 4.2.

4.1 Introduction

This section is mathematically informal and is meant only to make the rest of the chapter easier to understand. There is no normative material in Section 4.1 and the reader is free to skip straight to Section 4.2, which contains the mathematically rigorous material that is covered in this chapter.

Figure 16 shows two sensors, each of which has made an observation. Consider that we want to match a model whose one degree of freedom can range from -2 to 4 and is designated the variable m. The top red line graph in Figure 16 shows the relative probability of a given model being correct given the measurement, s_1, that the sensor has made. The horizontal axis of this figure represents the value of m. The vertical axis of the PDF does not have any units but the values of these probability density function graphs have been normalized (as required by the definition of a PDF) so that the integral of the function across the whole workspace is 1. Since they all have the same scale, it is possible to compare relative likelihoods between the graphs. The value of the parameter for the model is the horizontal

axis, and the relative chance of correctness is the vertical axis. The probability that the model, m, is correct may be expressed as $P(m|s_1)$.

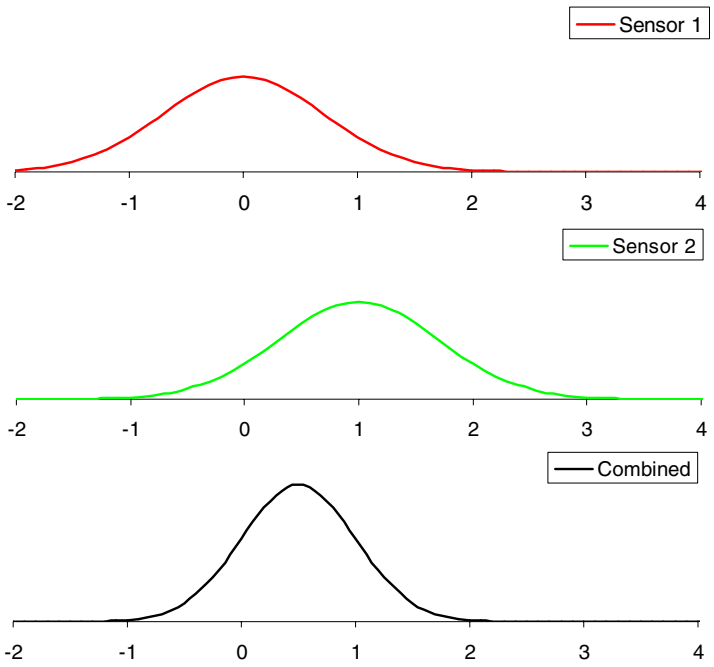

FIGURE 16. Simple combination of two normal sensors

Likewise, the second, green line graph describes the measurement from a second sensor. The graph of $P(m|s_1, s_2)$ is the black graph on the bottom of Figure 16. This shows a fairly simple and nice result. The estimate of the correct value using s_1 would be a model with a parameter of 0, while s_2 would give 1 and the combination would have 0.5 as the best model.

When two (or more) normal distributions are combined, the result is never a multimodal distribution. This becomes significant in the next example. Suppose we have a third sensor, s_3, which may have provided a bad measurement. Its graph is shown in Figure 17. The combined result of $P(m|s_1, s_2, s_3)$ shown in Figure 17 is correct, but s_3 has made any measurement somewhat dubious; the peak is at 2, even though no individual sensor thinks a value of 2 is very likely. The possibility

that a single outlier may be able to destroy the chance of finding the correct answer is a common impediment to making these types of systems robust.

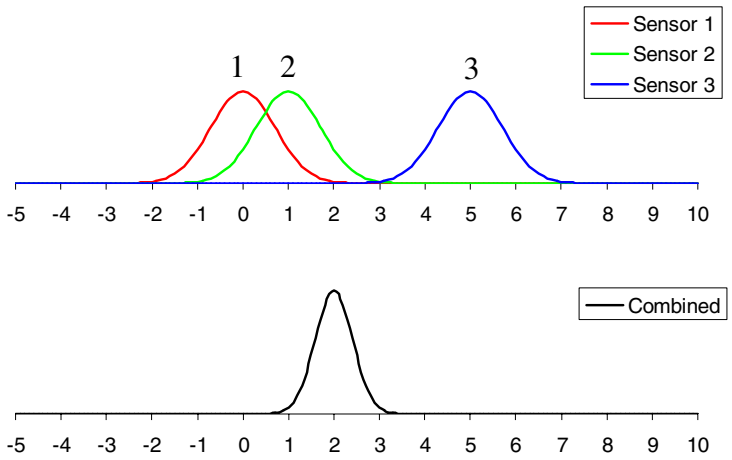

FIGURE 17. Simple combination of three normal sensors

This work explicitly models the possibility that the sensor could be wrong. I change the sensor model to the following: it is either right or wrong. If it is right, then it has a normal distribution. If it is wrong, it has a uniform distribution over the whole workspace. This is shown in Figure 18. The uniform part across the workspace has changed the scaling of the non-uniform part, because the graph is scaled to keep the total area of the probability density function at one.

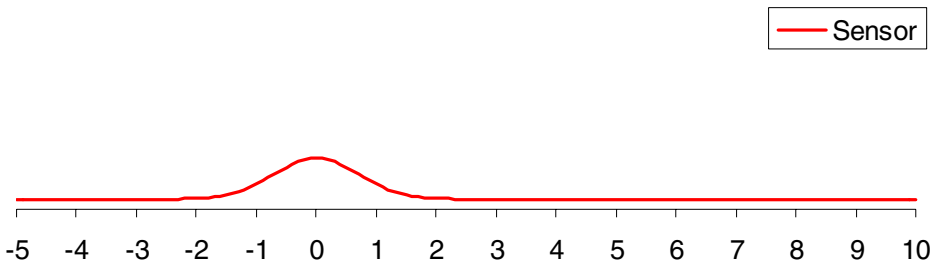

FIGURE 18. Sensors with failure model

This function can be described as the sum of the probability the sensor was wrong and the probability the sensor was correct, multiplied by the probability from a gaussian sensor. A slight approximation is needed to simplify the computation described later. Instead of using this sum, the sensor is approximated as the maxi-

mum of the uniform distribution and the gaussian distribution. This approximation is shown in Figure 19.

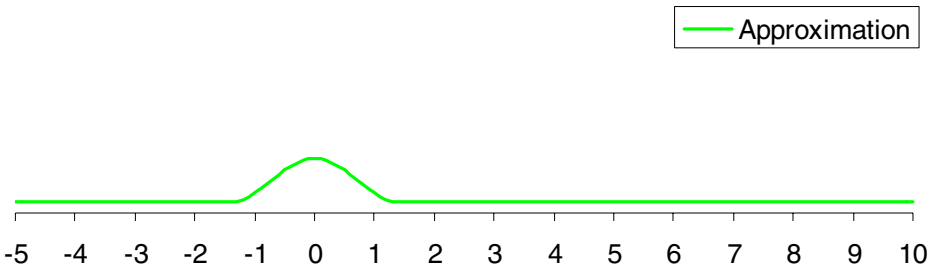

FIGURE 19. Approximate sensor model

In Figure 20, this approximation is overlaid as a green line onto the original sensor model. The approximation has a slightly higher peak than the original sensor model, but they are very close.

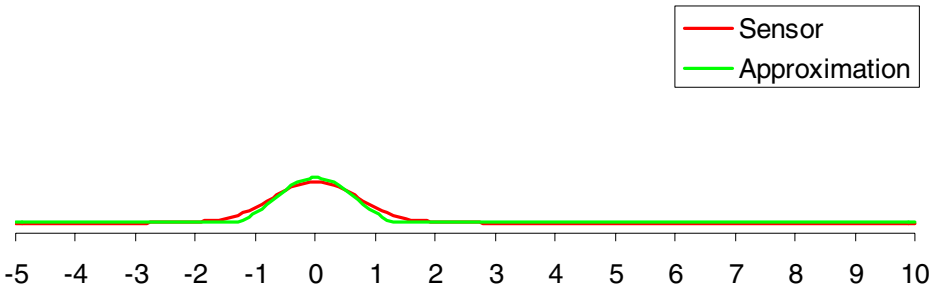

FIGURE 20. Approximation overlaid on original sensor model

It is important to note that this sensor model was a choice made in implementing the system. It increases the robustness, allows multimodal distributions (as shown later), and, because the math works out, still allows a fast implementation. This model is a reasonable, simple representation of the sensor errors described in the previous chapter.

The decision whether a sensor is right or wrong is done by the normal confidence test of whether a sample belongs to the normal distribution the sensor is proposing. Where a normal distribution is unlikely, it becomes a uniform distribution.

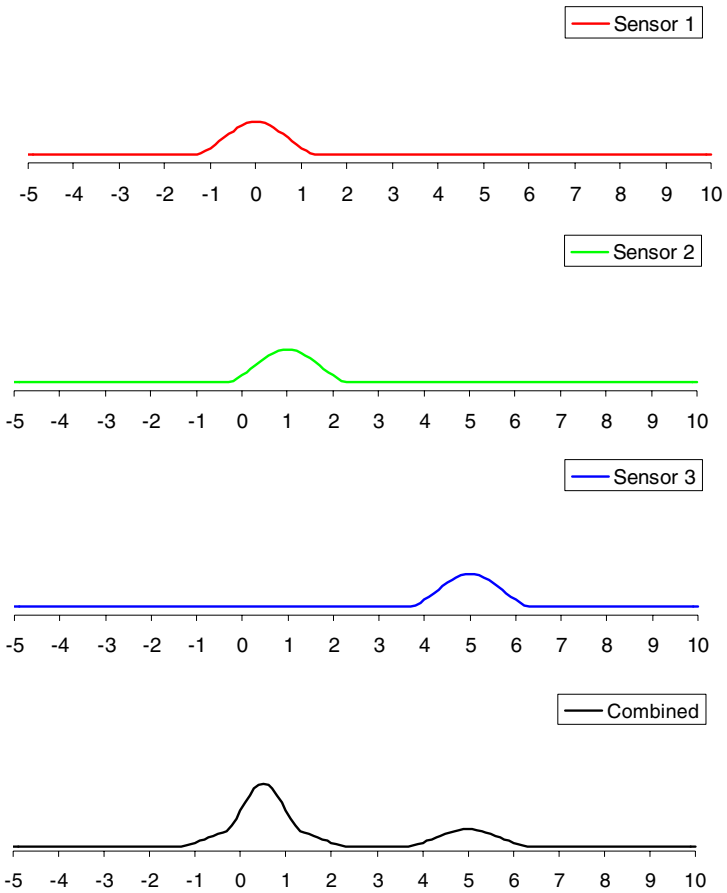

FIGURE 21. Combination of three sensors with failure model

When the three sensors are now combined, as shown in the bottom black graph of Figure 21, two possible answers emerge for consideration: 0.5 and 5. The 0.5 corresponds to s_3 being wrong, and 5 corresponds to s_1 and s_2 being wrong. It is now possible to get multimodal distributions.

Another concept used is prior belief. Suppose we know that at a previous time the correct model was at 1.0 and we also know that it can only have moved ± 1.0 since then. This gives us a belief, prior to any sensor measurements, that the solution is in the range shown on the top magenta graph of Figure 22. The sensors s_1, s_2, s_3 are drawn together on the middle graph of Figure 22. When these are combined, we get the result shown on the bottom black graph of Figure 22. This effectively adds in the previous information to the current solution and helps con-

strain the result to the correct solution. The prior completely eliminates any possibility of a solution over a large part of the workspace and thereby increases the probability in the part of the workspace where the solution is feasible. Since the areas of the graphs are scaled to one, this results in the peak from Figure 22 being more likely than the peak in Figure 21, which did not consider the priors.

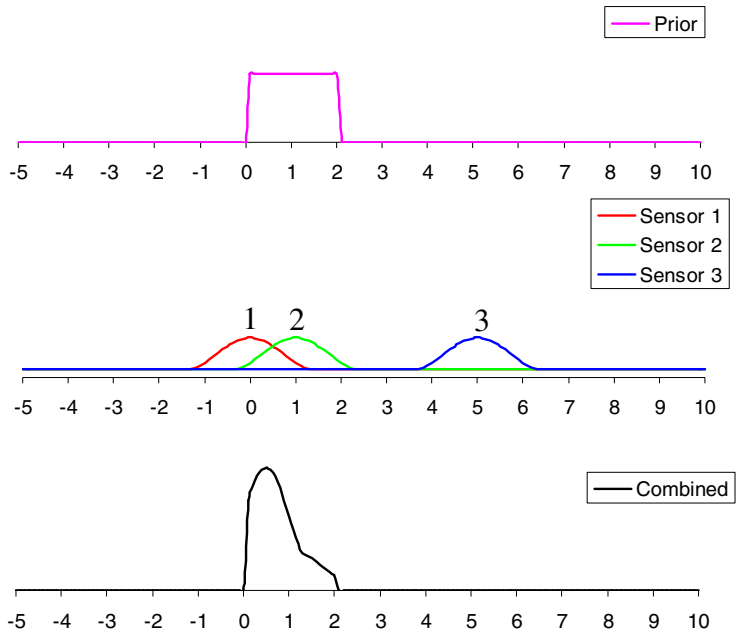

FIGURE 22. Combination of sensors and prior belief

In all the examples so far, the combined result function was computed for all points in the model space. This is not particularly efficient because the only points that are relevant for this work are the local maxima. The fact that only the maxima are interesting allows us to pick a random starting point and apply an optimization technique to migrate this starting point to the local maxima. Since the function might have several local maxima, we need to start at enough start points that at least one of the start points converges into each of the local maxima. By comparing all the local maxima we can find the global maximum, or the top n maxima, or all the local maxima with a posterior probability over a certain threshold.

This approach provides the benefits of an optimization to find the exact local solution, and the benefits of a particle filter-style search to find the global solution. Because only one starting point, or particle, needs to be in the local maxima region to get the correct solution, far fewer particles are required than are used in traditional particle filtering systems.

4.2 Definitions

This section describes the theoretical model that is used to combine all the sensor inputs discussed in Chapter 3.

Let the vector m be a model and $B(m)$ be the belief that m is the correct model. $B_a(m^t)$ is the belief, *after* a measurement has been taken at time t; and $B_p(m^t)$ is the belief *prior* to the taking of the measurement. 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 $S^t = \{s^t, s^{t-1}, \ldots, s^1\}$.

For the finger tracking example, the vector m contains the position and orientation of the finger. The state, s^t, is the images at time t from all the cameras.

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 $F^t = \{f^t, f^{t-1}, \ldots, f^1\}$.

In the finger tracking example, the filter functions f_i are the algorithms that we use to generate input cues. They are all the various cues discussed in Chapter 3 applied to the current set of images from all the cameras.

4.3 Filter Combination

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

$$B_a(m^t) = P(m^t | S^t) \quad \text{(EQ 1)}$$

which is the same as

$$B_a(m^t) = P(m^t | s^t, S^{t-1}) \quad \text{(EQ 2)}$$

Applying Bayes's rule we get

$$B_a(m^t) = \frac{P(s^t | m^t, S^{t-1}) P(m^t | S^{t-1})}{P(s^t | S^{t-1})} \quad \text{(EQ 3)}$$

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

$$B_a(m) = \eta P(s^t | m^t, S^{t-1}) P(m^t | S^{t-1}) \quad \text{(EQ 4)}$$

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

$$B_a(m) = \eta P(s^t | m^t, S^{t-1}) B_p(m^t) \quad \text{(EQ 5)}$$

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 $P(s^t | m^t, S^{t-1}) \cong P(s^t | m^t)$. Now Equation 5 becomes

$$B_a(m) = \eta P(s^t | m^t) B_p(m^t) \quad \text{(EQ 6)}$$

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

$$B_a(m) = \eta P(f^t \mid m^t) B_p(m^t) \qquad \text{(EQ 7)}$$

4.3.1 Independence assumption

To simplify the problem, the assumption that the measurements are independent is made. Ignoring the covariance information might not be without consequences, so I will discuss these here.

First, since all the measurements come from some processing of the same basic images, they are not completely uncorrelated. However, they each do provide additional information, so they are not completely correlated either. Imagine a case in which the same measurement is taken twice and presented as two independent channels. If each had a probability of x, the combined result would be x^2. This understates the result; if the result were low - say 1% - it would be presented as 0.001%, and if it were high - say 90% - it would be presented as 81%.

Much of the use of probabilities in this system is to compare two models and see which is best. Say the true probabilities are x and y for two different models. If they are understated because covariance is ignored and the system thinks they are x^2 and y^2, the system will still choose the correct model because if $x > y$, then $x^2 > y^2$.

It is difficult to formally analyze the effects of ignoring covariance information, but experimentally the system still works well even with this approximation.

Making the independence assumption, therefore, gives

$$P({f_1}^t, {f_2}^t, \ldots, {f_n}^t \mid m^t) = \prod_{i=1}^{n} P({f_i}^t \mid m^t) \qquad \text{(EQ 8)}$$

Equation 7 thus becomes

$$B_a(m) = \eta\left[\prod_{i=1}^{n} P(f_i^t \mid m^t)\right] B_p(m^t) \qquad \text{(EQ 9)}$$

4.4 Computing the Conditional Probabilities

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. This is described in more detail in Section 4.10.

The errors are evaluated by using the projection functions described in section 4.6 to project the model into the measurement space, and 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.

In the finger tracking example the error metrics are fairly simple. 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 images, 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 is close to gaussian.

4.5 Computing the Prior Belief

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 (see section 4.8 regarding this usage of priors). 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. A different distribution of how the finger has moved from one frame to the next may be desirable but this choice in distribution allowed for an implementation that could run in real time. If all that is known about the motion of the finger is that it is random and moves at less than a certain velocity, this could be the best distribution to choose. This ability to use the same framework for both the tracking and the initialization of the tracking is an important part of the system.

4.6 Probability Representation and Least Squares

The software stores the negative log of the probability instead of the actual probability. Doing so simplifies the calculations and better suits the large range that the probabilities cover. This causes Equation 9 to become

$$-\log(B_a(m)) = -\log(\eta) - \log(B_p(m^t)) + \sum_{i=1}^{n} -\log(P(f_i^t \mid m^t)) \quad \text{(EQ 10)}$$

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

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

The distribution of the filter function is normal if we feel the channel is working and uniform otherwise. In some cases there might be extra information that suggests that the channel is working. For example, if 90% of the pixels in the image are apparently skin colored, skin detection is probably not working and should just be ignored. Testing with this type of extra information was not used in this work but could, in future work, extend it in ways that would improve the robustness of the system. This work sets aside the issue of extra information, performing the classic confidence test to decide if the data looks like it could belong to the current model.

The classical normal test equation, when $\bar{x}$ is the mean sample value, a is the distribution mean, σ is the distribution standard deviation, and n is the number of samples, is

$$z = \frac{\bar{x} - a}{\sigma/(\sqrt{n})} \quad \text{(EQ 12)}$$

If $|z_i| > c$, then the sample is assumed not to have come from that normal distribution. An arbitrary confidence level is chosen, here 5%, which gives $c = 2.576$. It may be possible to learn a value of c for each different sensor, but this approach

was not tried. When the normal test equation is written in terms of the variables used here, it becomes

$$z_i = \frac{f_i^t - \mu_i(m^t)}{\sigma_i} \qquad \text{(EQ 13)}$$

and if

$$|z_i| > c \qquad \text{(EQ 14)}$$

then the channel is assumed to be failing and a uniform distribution is used. Otherwise the normal distribution is used. This normal test equation approach relies on a small proportion of the data being outliers.

So now Equation 11 can be written as

$$P(f_i^t \mid m^t) = \begin{cases} \alpha_i \frac{1}{\sqrt{2\pi\sigma_i}} e^{-\left(\frac{f_i^t - \mu_i(m^t)}{\sigma_i}\right)^2} & if \left|\frac{f_i^t - \mu_i(m^t)}{\sigma_i}\right| \le c \\ \alpha_i \frac{1}{\sqrt{2\pi\sigma_i}} e^{-c^2} & otherwise \end{cases} \qquad \text{(EQ 15)}$$

where α_i is the appropriate constant such that the integral of $P(f_i^t \mid m^t)$ over the workspace is 1. The constant term

$$\alpha_i \frac{1}{\sqrt{2\pi\sigma_i}} e^{-c^2} \qquad \text{(EQ 16)}$$

in the "otherwise" case results in the $P(f_i^t \mid m^t)$ function being continuous. In a case where f_i is one dimensional, the graph of $P(f_i^t \mid m^t)$ looks very much like Figure 18. Here $\bar{f}_i$ is constructed so that $\bar{f}_i$ can be used as a robust estimator of f_i and Equation 15 can be written to look like a normal distribution in terms of $\bar{f}_i$. Specifically we define $\bar{f}_i$ to be

$$\bar{f}_i^t = \begin{pmatrix} f_i^t & if \left| \frac{f_i^t - \mu_i(m^t)}{\sigma_i} \right| \le c \\ (c\sigma_i + \mu_i(m^t)) & otherwise \end{pmatrix} \quad \text{(EQ 17)}$$

I will now work towards an alternative way of expressing the function shown in Equation 15. I will eventually arrive at Equation 21 and have shown that it is equivalent to Equation 15. Note that by looking at the distance between $\bar{f}_i^t$ and the expected value of the model, we can see that

$$\left| \bar{f}_i^t - \mu_i(m^t) \right| = min(\left| f_i^t - \mu_i(m^t) \right|, \sigma_i c) \quad \text{(EQ 18)}$$

so from an implementation point of view, the robustness added by doing the normalizing test reduces to bounding the value of any measurement.

We can see that when $\left| \frac{(f_i^t - \mu_i(m^t))}{\sigma_i} \right| > c$ then

$$c^2 = \left(\frac{\bar{f}_i^t - \mu_i(m^t)}{\sigma_i} \right)^2 \quad \text{(EQ 19)}$$

and when $\left| \left(\frac{\bar{f}_i^t - \mu_i(m^t)}{\sigma_i} \right) \right| \le c$ then

$$\left(\frac{f_i^t - \mu_i(m^t)}{\sigma_i} \right)^2 = \left(\frac{\bar{f}_i^t - \mu_i(m^t)}{\sigma_i} \right)^2 \quad \text{(EQ 20)}$$

So now Equation 15 can be written as

$$P(f_i^t \mid m^t) = \alpha_i \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\left(\frac{\bar{f}_i^t - \mu_i(m^t)}{\sigma_i} \right)^2} \quad \text{(EQ 21)}$$

Taking the negative log of Equation 21, we get

$$-\log(P(f_i^t \mid m^t)) = -\log\left(\alpha_i \frac{1}{\sqrt{2\pi\sigma_i}}\right) + \left(\frac{\bar{f}_i^t}{\sigma_i} - \frac{\mu_i(m^t)}{\sigma_i}\right)^2 \quad \text{(EQ 22)}$$

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

$$-\log(P(f_i^t \mid m^t)) = \eta_i + \left(\frac{\bar{f}_i^t}{\sigma_i} - \frac{\mu_i(m^t)}{\sigma_i}\right)^2 \quad \text{(EQ 23)}$$

Equation 10 can now be written in the form

$$-\log(B_a(m)) = \sum_{i=1}^{n} \eta_i - \log(B_p(m^t)) + \sum_{i=1}^{n} \left(\frac{\bar{f}_i^t}{\sigma_i} - \frac{\mu_i(m^t)}{\sigma_i}\right)^2 \quad \text{(EQ 24)}$$

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

$$-\log(B_p(m^t)) + \sum_{i=1}^{n} \left(\frac{\bar{f}_i^t}{\sigma_i} - \frac{\mu_i(m^t)}{\sigma_i}\right)^2 \quad \text{(EQ 25)}$$

This is almost in the form of a non-linear least squares problem, except for the $\log(B_p(m^t))$ term. Solving this is discussed in detail in section 4.7; it is a sequential process in which the $\bar{f}_i^t$ and the $\mu_i(m^t)$ are computed at each step in the iteration. Since $B_p(m^t)$ is constant in a feasible region and zero outside it, the $\log(B_p(m^t))$ term will be 0 inside the feasible region and $-\infty$ outside of it. This forces the solution to be inside the feasible region but has no other effect. This constraint that it must be inside the feasible region is embedded into a least squares solver that finds a solution to

$$\sum_{i=1}^{n}\left(\frac{\bar{f}_i^t}{\sigma_i}-\frac{\mu_i(m^t)}{\sigma_i}\right)^2 \qquad \text{(EQ 26)}$$

Since the solution is in the feasible region, we know that the solution to Equation 26 is also the correct solution to Equation 25.

4.7 Stabilized Least Squares Solution

The problem of finding the locally optimal model reduced to finding the solution to the least squares problem in Equation 26 while keeping the constraint that the solution be in the feasible region. The stabilization methods described by Lowe [41] 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. This section describes how Lowe's approach is used in this system but does not extend that work.

Since the μ_i functions are nonlinear, they will require an iterative solution. This is done by linearizing the functions $\mu_i(m^t)$ around the current estimate of m^t, which will be called m_0. Now define the partial derivatives of $\mu_i(m^t)$ with respect to each element of the column vector m^t to form a row vector j_i, where the jth element of the vector is given by

$$j_{ij} = \frac{\partial}{\partial m_j}\mu_i(m^t)\Big|_{m_0} \qquad \text{(EQ 27)}$$

Now define the column vector x to be the change in m^t from m_0 so

$$x = m^t - m_0 \qquad \text{(EQ 28)}$$

This will result in the linearization of $\mu_i(m^t)$, giving the approximation

$$\mu_i(m^t) \approx \mu_i(m_0) + j_i x \quad \text{(EQ 29)}$$

From this Equation 26 can be approximated as

$$\sum_{i=1}^{n} \left(\frac{(\bar{f}_i^t - \mu_i(m_0))}{\sigma_i} - \frac{j_i x}{\sigma_i} \right)^2 \quad \text{(EQ 30)}$$

A column vector b is defined, in which the ith element is given by

$$b_i = \frac{(\bar{f}_i^t - \mu_i(m_0))}{\sigma_i} \quad \text{(EQ 31)}$$

A matrix A is also defined, in which the element of the ith row and jth column are given by

$$a_{ij} = \frac{j_{ij}}{\sigma_i} \quad \text{(EQ 32)}$$

Now Equation 30 can be written in the familiar form of

$$\|Ax - b\| \quad \text{(EQ 33)}$$

Following Lowe's logic, we would also like to stabilize these equations, meaning we do not want the value of m^t to change unnecessarily from our current estimate of it, which is m_0. This results in the following term which we would like to minimize along with the other terms:

$$\left\| m^t - m_0 \right\| \quad \text{(EQ 34)}$$

The question is how to appropriately weight this term. The values in Equation 33 have been correctly weighted by the σ_i so that they are normalized to each have an expected variance of 1. The terms in Equation 34 also need to be weighted

to give them an expected variance of 1. This is done by dividing the jth element of the m^t and m_0 by the standard deviation of the jth element of the model, which I will call σ_j. Note that the σ_j are not the same as the σ_i, which are the standard deviations of the measurements. They are both learned in the training stage described later in Section 4.10.

Given the definition x in Equation 28, Equation 34 can be written as:

$$\|Ix - 0\| \quad \text{(EQ 35)}$$

At this point I am going to introduce a new matrix W which contains the appropriate weights to correctly weight the stabilization terms. The matrix W is defined to be zero except for the diagonal entries, where the jth diagonal elements w_{jj} will have the value

$$w_{jj} = \frac{1}{\sigma_j} \quad \text{(EQ 36)}$$

Now Equation 34, correctly weighted, can be written as

$$\|Wm^t - Wm_0\| \quad \text{(EQ 37)}$$

which is

$$\|Wx\| \quad \text{(EQ 38)}$$

This can be combined with Equation 33 and written as

$$\left\| \begin{bmatrix} A \\ W \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\| \quad \text{(EQ 39)}$$

The value that minimizes this is solved using the normal equations technique. As shown by Strang [68], the normal equation approach gives the solutions that minimize Equation 39. The solution is

$$x = \left(\begin{bmatrix} A \\ W \end{bmatrix}^T \begin{bmatrix} A \\ W \end{bmatrix} \right)^{-1} \begin{bmatrix} A \\ W \end{bmatrix}^T \begin{bmatrix} b \\ 0 \end{bmatrix} \quad \text{(EQ 40)}$$

This solution to this is the same as solving the system

$$\begin{bmatrix} A \\ W \end{bmatrix}^T \begin{bmatrix} A \\ W \end{bmatrix} x = \begin{bmatrix} A \\ W \end{bmatrix}^T \begin{bmatrix} b \\ 0 \end{bmatrix} \quad \text{(EQ 41)}$$

which reduces to

$$(A^T A + W^T W)x = A^T b \quad \text{(EQ 42)}$$

This equation provides the solution for the stabilized least square solution to find the best value of x, which then gives the solution to m^t. It is solved using Gauss Jordan with maximal column pivot selection.

Since we linearized the $\mu_i(m^t)$ functions, they will not get the exact answer the first time. This equation needs to be solved iteratively, which requires recomputing the partial derivatives $\frac{\partial}{\partial m_j}\mu_i(m^t)$ at each step. The partial derivatives are computed using finite differences. This iteration converges very quickly, typically after two or three cycles. The iteration is stopped when the model stops moving significantly or after 10 iterations. On rare occasion, the solution gets worse from one step to the next. In this case the iteration is stopped and the previous result is used. (This happens when a step size was too large; approaches to search for the right step size just slowed down the system and did not improve robustness.)

Constraints can be added to the least squares solution in two different ways. Keep in mind that for the result to be the correct solution to Equation 25, we still need to add in the constraint that the solution stay in the feasible region when solving Equation 42. One possibility 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. This approach to constraints makes it very easy to add constraints like “Pay attention only to fingers with an orientation between -45 and +45 degrees.”

Some texts point out that the least squares problem often has bad rounding errors when solved this way and it would be better to use SR factorization or SVD. This was not a problem - it may be that just moving to double precision solves this problem or it may have to do with the stabilization factor. The adding of the stabilization factors ensures that even after subtracting other rows that are almost identical, each row has one value that is not close to zero. Since I work on processors that do all floating point arithmetic using an internal representation that is slightly larger than a double, there is no speed gain to using a float. The doubles give the same stability using Gauss Jordan elimination as the factorization type techniques using floats, and Gauss Jordan only requires half the operations.

4.8 Propagating the Prior Belief

Having a uniform prior belief within a particular region of the solution space with the probability dropping to zero elsewhere may seem strange, but it is consistent with the belief that the finger will likely not move more than 7 cm, but within that area, it could be anywhere. It would be possible to do some training for a particular finger tracking problem and learn the motion distribution. However, any distribution that was not uniform over some space would require a different approach to the least squares problem than just embedding it as a constraint. The current scheme works well and results in a fast implementation.

Future work could include looking at adaptive search region sizes. If a small region is used, it can be computed much faster. Because it is computed faster, the finger will not have moved as far since the previous result. Picking the

best size of region to search so as to get the fastest tracking possible while ensuring the finger remains in the search region is a problem that was not addressed.

The previous position estimate can be used to form a rectangular region of interest (ROI) for each input sensor to reduce the amount of the image that needs to be processed for each of the input channels. This works fairly well when only one answer is propagated forward from each frame, but when multiple answers are portaged forward, the ROI often expands to be most of the image.

4.9 Tracking & Searching

The approach described here gives a way to evaluate the probability of a given model being correct. It eventually derives into a least squares solution for finding the locally optimal solution, but it still needs an initial starting solution and a search across the solution space to find the globally optimal solution.

One of 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 does not work for fingers. Most people can easily move their fingers at 4 m/s; a baseball pitcher can reach ten times that speed. The accelerations are also 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 limit 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. In simple images the convergence region is quite large, but in difficult images, 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.

4.9.1 Randomized search

In an earlier phase of this research, the system was implemented using a systematic search that stepped over the whole space in same-sized steps. When the finger had not been found in the previous frame and the whole work space needed to be covered, it took about two seconds to evaluate all the models. The current system improves on this approach. Say 10,000 models would have been evaluated in the systematic approach. Instead, 2500 models are randomly selected and evaluated. Because the radius of convergence is fairly large, there is a reasonable chance that the system will find the finger with these 2500 samples. If it does not, another 2500 random samples are selected in the next frame. Even if no solution is found for this frame, searching it takes only half a second, and in less than two seconds the same number of locations have been evaluated. It is likely that one of the frames will yield a solution as good as that which would have been found through the systematic approach, because over four frames a similar number of sample points are evaluated. By varying the 2500 number, the algorithm can dynamically adapt to achieve the desired frame rate but have better odds of finding the correct solution on a more powerful computer. This solution is quite nice for a real time system, in that it adapts to provide an answer at the desired time and provides the best possible answer for the power of the computer on which it is running.

The dimensionality of this search space can be greatly reduced by some pre-filtering techniques. These make the assumption that certain channels of information are correct or are at least failing in a certain way.

The first pre-filter assumes the range information is correct. Instead of choosing random models out of the x, y, z space, a random pixel on the range

image is chosen. From it we compute the x, y, z coordinates that the pixel corresponds to and use these for the location of the model. This effectively ends up only picking models that lie on the surface generated by the range information.

The second pre-filter used is color. For a given model, its x, y, z position is projected into the image and if no skin colored pixel lies nearby, the model is thrown out. This assumes that the color channel is saying the image is skin colored wherever there is a finger. (No problem arises if it also picks up some things that are not fingers.)

These pre-filters speed up the system but reduce the overall robustness by assuming that some particular channel is not failing. They were not used in the test described in Chapter 5 or in the final result sequence shown in Chapter 6.

4.10 Training

The system needs an estimate of the standard deviation, σ, for each of the projections functions and model parameters. This is found by initially assuming the value is 1 and running the system on a simple training sequence. This simple sequence can easily be tracked and the values of the σ computed. The values found in this experiment are then put back into the algorithm and used for future tracking. At this point, much more complex sequences can be correctly tracked. The algorithm is relatively insensitive to changes in the variance; they can be doubled or halved with little effect on the results.

The assumption here is that the training set is simple enough that it has no outliers and the value of σ found will not be corrupted with outliers. The training set was manually checked to make sure that the system correctly tracked every frame. Future work could look at using the training data to learn the values to use for c in Equation 14.

4.11 Non Maximal Suppression

At the end of this search stage, each model has a certain likelihood of correctness. Models below a certain threshold (5%) 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. Two models are considered to be the same if they have converged to within 5 mm of each other in all x, y, z directions and to within 5° in rotation. These numbers were picked because no two fingers in the scene could be this close together. In practice, models that were this close seemed to lie in a single local minima basin and converged, so that their locations were extremely close. The likelihood weights of all the models that have been combined are summed into the one model that is kept.

At this stage there may be more than one "good" model - particularly if there are two or more fingers in the scene. Either all the models can be propagated forward to the next frame and used as starting points, or just the best can be used. The ability to find an unknown number of fingers in the scene is very useful for some applications, but in others, such as for a 3D virtual mouse, only the best solution would be selected.

4.12 Comparison to Sequential Monte Carlo

The classic form of the sequential Monte Carlo (SMC) algorithm as shown in many papers, including one by Doucet et al. [16], is illustrated in Figure 23. I first review this traditional form and then describe adaptations done in this work.

In the first step of Figure 23, there are several random samples of possible solutions each referred to as a particle. In step two, the likelihood of each of these is evaluated and they are weighted accordingly. In step three, a new set of parti-

cles is chosen with replacement from the previous set adjusted for the weights. This will cause particles that are not very likely to be unlikely to be chosen at all and particles that are very likely to be chosen multiple times. In step four, the dynamics of the system are applied to each particle. If there is a known expected motion of the particle, it is applied here, much as it is with a Kalman filter. In addition, the random motion that is characteristic to the motion of the solution is also applied. This spreads the particles to new random locations to which the particle could reasonably have moved in one time step. These particles form the input for the next cycle.

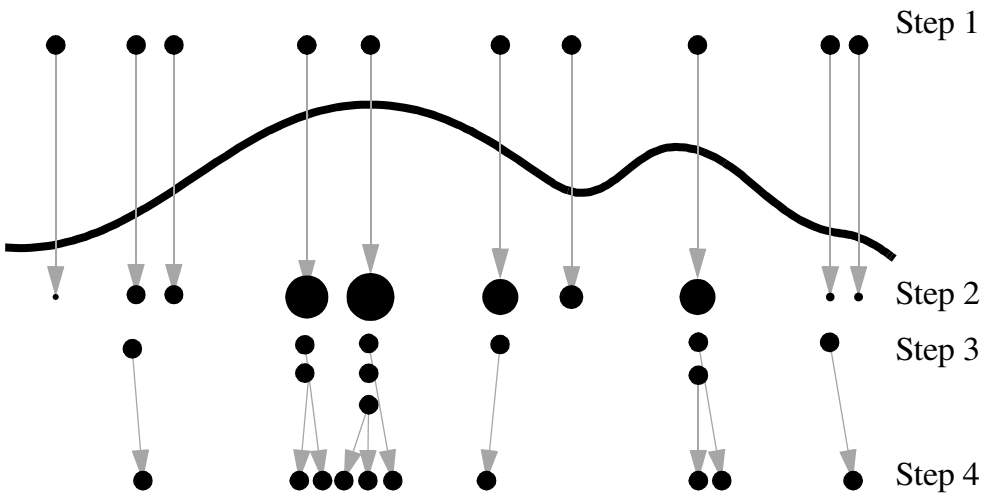

FIGURE 23. Traditional particle filtering

A nice proof that particle filters actually converge is covered in the work of Crisan [9] and Moral and Jacod [50], while practical evidence that these systems work for many vision cases is found in the growing interest in particle filtering in the vision research community.

My initial experiments with traditional 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. Recent work [45][44] on using factorization techniques to reduce the

dimensionality of problems so that condensation can be used may provide a solution; this is an interesting area for future work

Instead of the classical particle filtering approach, I combined an SMC-type approach with a least squares optimization. The process used is shown in Figure 24. There is an extra step "A" where the particles go through a least squares process to bring them to the local optimal of the likelihood function.

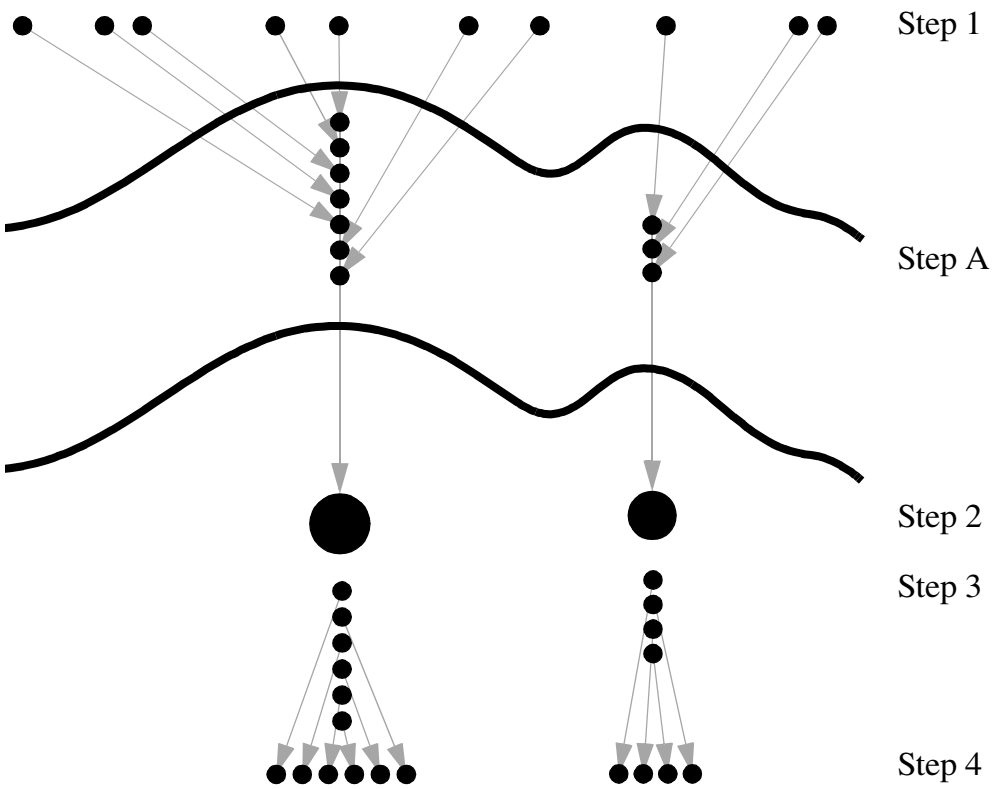

FIGURE 24. Modified approach

The advantage of this approach is that it results in particle convergence in substantially fewer steps with far fewer particles. Unfortunately, it also introduces a large bias in the sampling by moving the sample in a non-random way.

Nando de Freitas[1] and some of the work on unscented particle filters [48][47][57] suggest a way to compensate for this bias. (More information about

1. Personal communication, January 2002.

unscented particle filters is provided in section 2.4.1.) These authors point out that the importance weights are

$$\omega_i^t = \frac{P(f_i^t \mid m^t) P(m^t \mid m^{t-1})}{q(m^t \mid f_i^t, f_i^{t-1}, \ldots, f_i^1, m^{t-1}, m^{t-2}, \ldots, m^1)} \quad \text{(EQ 43)}$$

Importantly, the q is a choice and any importance function that includes the support of $P(m^t \mid m^{t-1})$ can be used. The classical choice for q is the prior, $P(m^t \mid m^{t-1})$, which results in the importance weights simply being the likelihood functions, so that

$$\omega_i^t = P(f_i^t \mid m^t) \quad \text{(EQ 44)}$$

The problem with this choice is that it ignores the value of the most recent measurement, f_i^t, which can result in particles being placed in locations that are very unlikely, given the most recent measurement; very few particles will therefore survive the resampling. As van der Merwe et al. observe, "it is therefore of paramount importance to move the particles towards the regions of high likelihood" [47]. This is, of course, what the least squares fitting ends up doing.

In the case of least squares optimization, each step could be considered a resampling using a choice of q that is a function of the gradient of the likelihood function and using the prior as a starting point for the optimization. This alternative q function has not been implemented and the detailed form of it is not specified but it is an interesting possibility for correcting for the biasing. So in this case q from Equation 43 would be of the form

$$q(m^t \mid f_i^t, m^{t-1}) \quad \text{(EQ 45)}$$

Now the least squares can be embedded in the q function, but the importance weights need to be compensated for this. The current system does not make this adjustment, but it is an interesting area for future work. Other authors are

moving toward this type of approach [15][55][7][69][47]. The hope is that if the weights were not biased, the system would get better results, possibly with fewer particles.

4.13 Summary of the Process

In summary, the overall process is as follows. First, some random model locations are chosen. These are constrained by using information about the location of the finger in the previous frame and, optionally, constraints arising from color and range images. Each one of these models is used as a starting point for a least squares local optimization. This least square fit iterates and attempts to fit the model simultaneously to all the different information available. This fit takes into account a probability that any given channel is failing and simultaneously combines all the information from various measurement channels. After all the models have been fit, the best one or set of best ones is chosen as the global solution.

5. Results

This chapter shows some of the initial image processing steps that form the evidence that is used for the tracking, and also the intermediate results of the algorithm for tracking a frame. Other tracking sequences then show the system's robustness to certain hard conditions. The end of this chapter examines the errors, accuracy, and speed of the system.

5.1 Intermediate Images

Figure 25 shows the input image from the cameras after the correction as described in Section 3.1.

FIGURE 25. Corrected images from three cameras, with range images

The top left is the stereo range information, while the others represent the views from the top, left, and right cameras. The stereo matching searches over 64 disparities, so there is a 64-pixel-wide area on the bottom and right sides of the range image where no result can be computed. This area is drawn in green in Figure 25. If the system were changed to do two camera stereo in the areas where three are not available, there would be only a small green square in the bottom left where range data is absent. This would give a 61% increase in the number of range pixels available and expand the usable work space without reducing accuracy.

A grayscale version of these images is computed and used to find the edges as described in Section 3.2.4. The resulting image is shown in Figure 26.

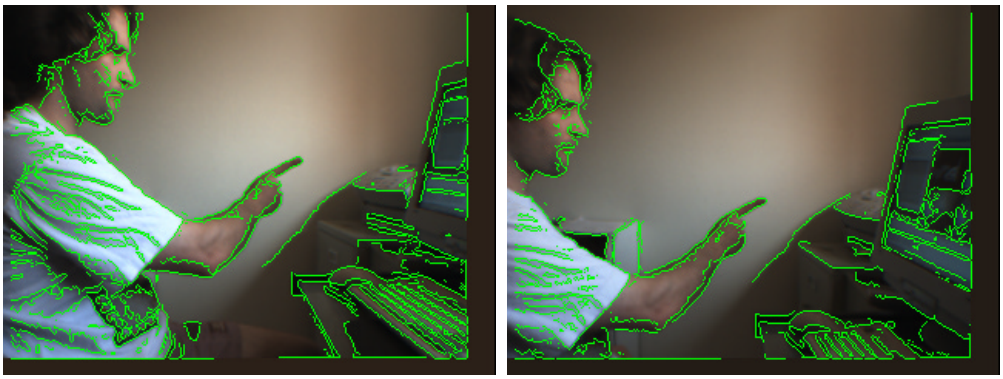

FIGURE 26. Left and right edge images

The edges are fairly consistent between the left and right images. Like the Canny edge detector, this Shen and Castan method [60] needs a gaussian blurring factor that represents the basic scale of the edges sought. Edge images from all three cameras are used as separate input channels to the robust combination.

The hue is shown in Figure 27 to illustrate the signal to noise ratio of the color sensors. Some of the NTSC cameras used early in this work produced far

more noise. The digital camera currently in use has a bit more noise than high end 3CCD RGB output cameras.

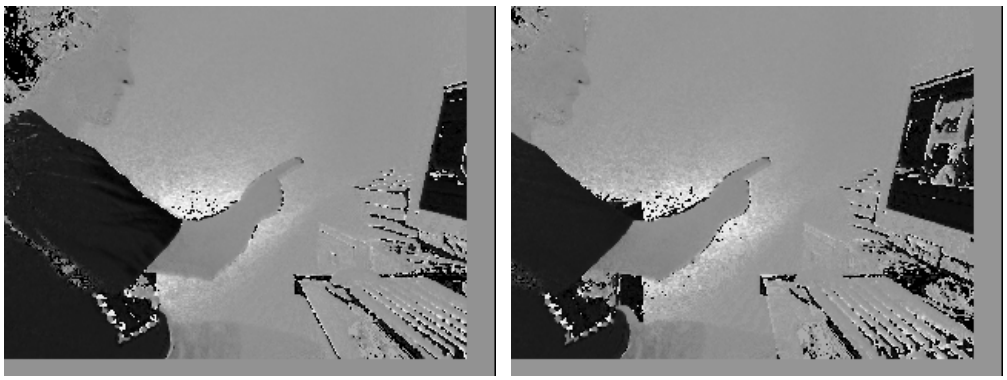

FIGURE 27. Left and right hue

The color detection process described in Section 3.2.2 detects the skin colored regions, as shown in Figure 28.

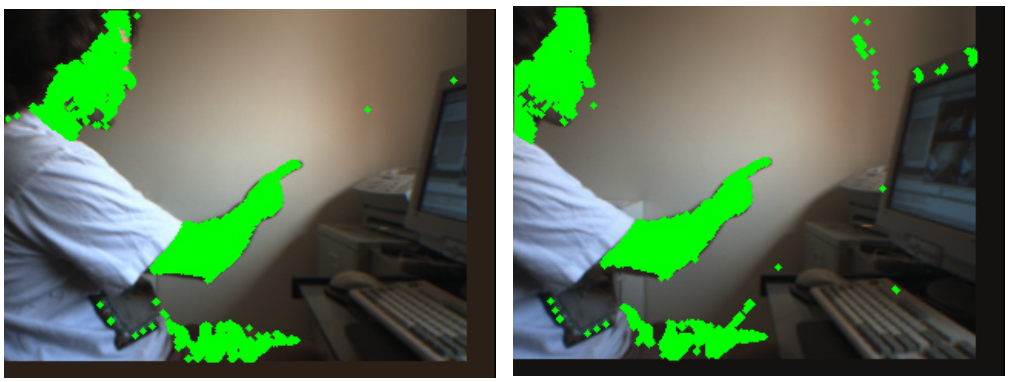

FIGURE 28. Skin detection

Here it has done a reasonable job, but there are some small noise points on the wall and the shirt. In the left image, a few points near the eyes have been missed. Typically, as here, skin detection is largely correct but has some noise.

The convexity point detection algorithm described in Section 3.2.3 gives the results shown in Figure 29. Noise points in the skin detection are usually fairly isolated and turn up as convex points in this image. Skin detection evidence is good in that as long as the color detection isolates the finger tip, there are very high odds that some convexity points will be found around it. On the other hand,

skin detection also always finds several convexity points around the image that are not associated with a finger tip, such as the tip of the nose.

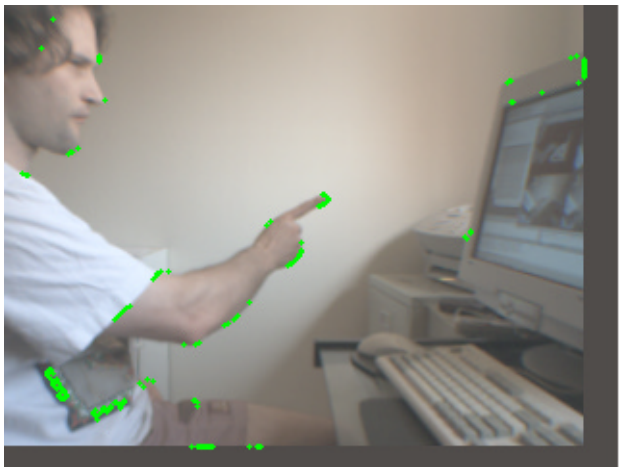

FIGURE 29. Convexity points

The initial set of 1000 randomized models that the system attempts to fit is shown in Figure 30.

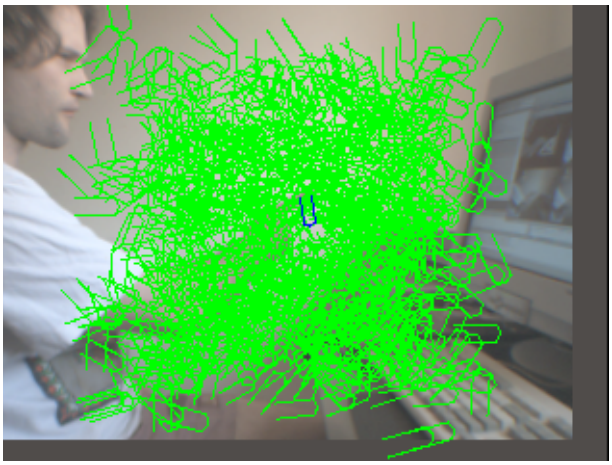

FIGURE 30. Randomized models

This corresponds to the initial particles in the first step of the particle filtering algorithm described in Section 4.12. More models are used but only 1000 are shown here because the image becomes too cluttered with more. These models could be pre-filtered, as described in Section 4.9.1, if we were willing to assume that a particular evidence channel was not failing. This would greatly speed up the

results, but since maximum robustness was more desirable than speed for this test set, pre-filtering was not used.

The next step in the algorithm is to do the least squares fit of each model. Each of the models is optimized by solving the equations set up in Section 4.6. The result is that the models shown in Figure 30 move to become the set of models shown in Figure 31.

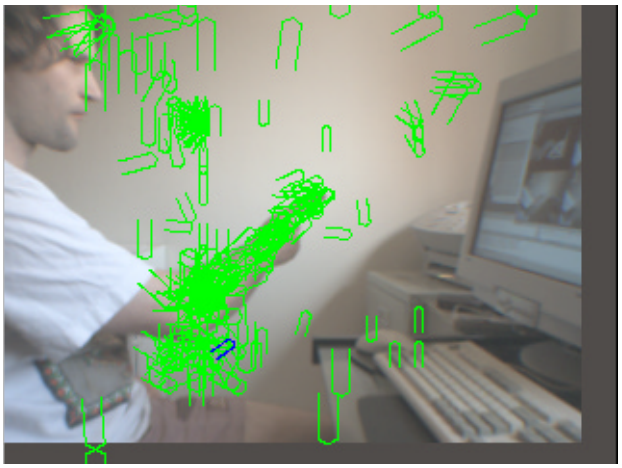

FIGURE 31. Initial least squares fit of models

The models in the flat region against the wall are caused by the local optimization of models in Figure 30. Nothing guides them, so they just converge or randomly move around. Since they are such poor fits, they are very unlikely and get thrown out at the next stage.

The next steps of the algorithm are resampling and maximum likelihood suppression. The models that are unlikely or equivalent to other models are thrown out to get the results shown in Figure 32. In this case, every model selected is on the finger. Many models very close to the same location are drawn on top of each another in Figure 32. These models will act as the starting point for the dynamics and diffusion to be applied to form the initial models for the next time step.

FIGURE 32. Adjusted least squares fit of models (left & right image)

From these possible models the best model is selected. In Figure 33 the final model is shown overlaid on the images from all the cameras.

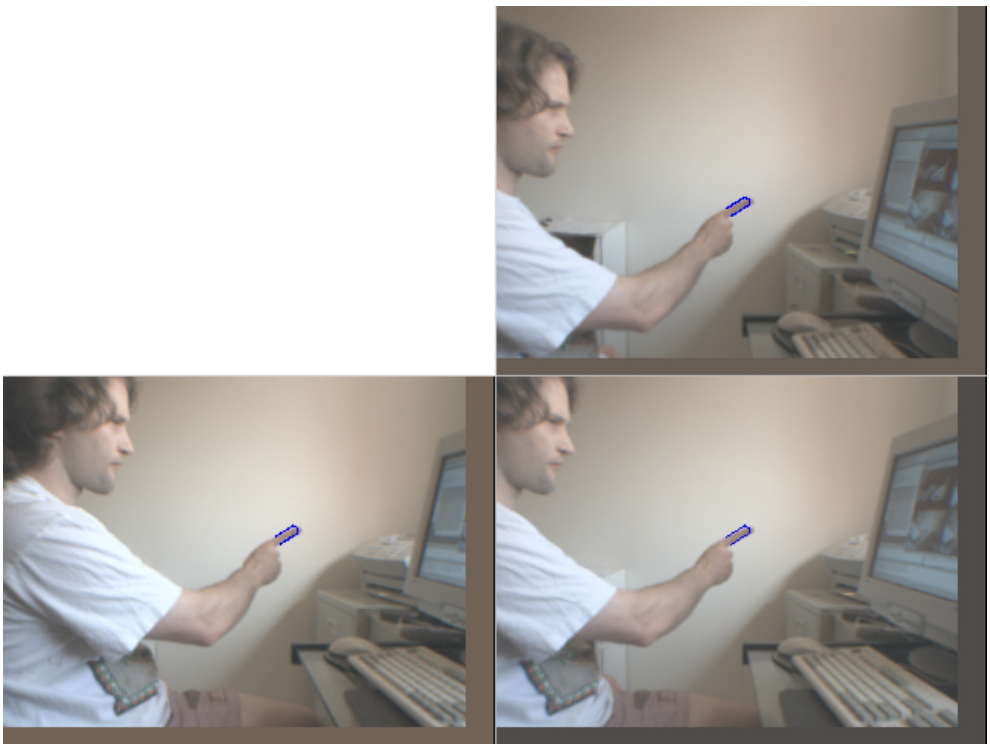

FIGURE 33. Final models

It is easy to see that the best model correctly overlays the finger when projected into all three views. The projection of the tip of the finger into the 3D work space is shown in Figure 34. In Figure 34 the work space is drawn in white. A gray cross hair that intersects the finger tip is drawn, and the projection of that

cross hair onto the faces of the work space is drawn in green. A red line indicating the orientation of the finger is drawn intercepting the finger tip.

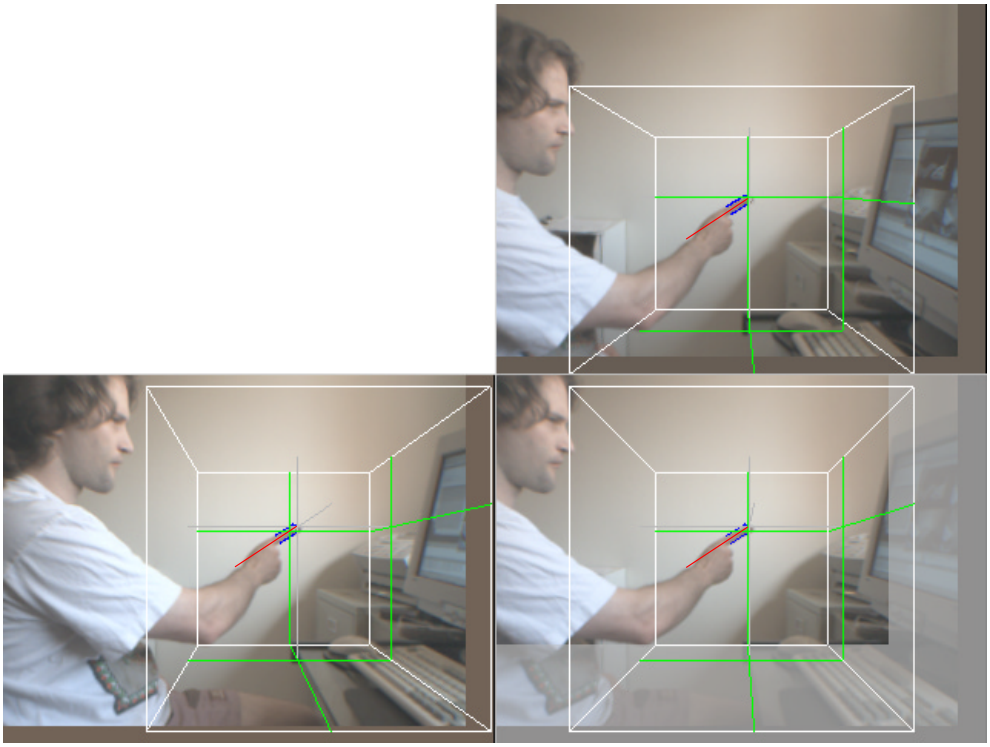

FIGURE 34. Projected model

5.2 Test Sequences

In this section several types of image sequences that typically cause problems for tracking systems are run to demonstrate the robustness achieved by this system.

5.2.1 Complex backgrounds with many edges

Figure 35 shows the results of tracking in a test sequence with a challenging background.

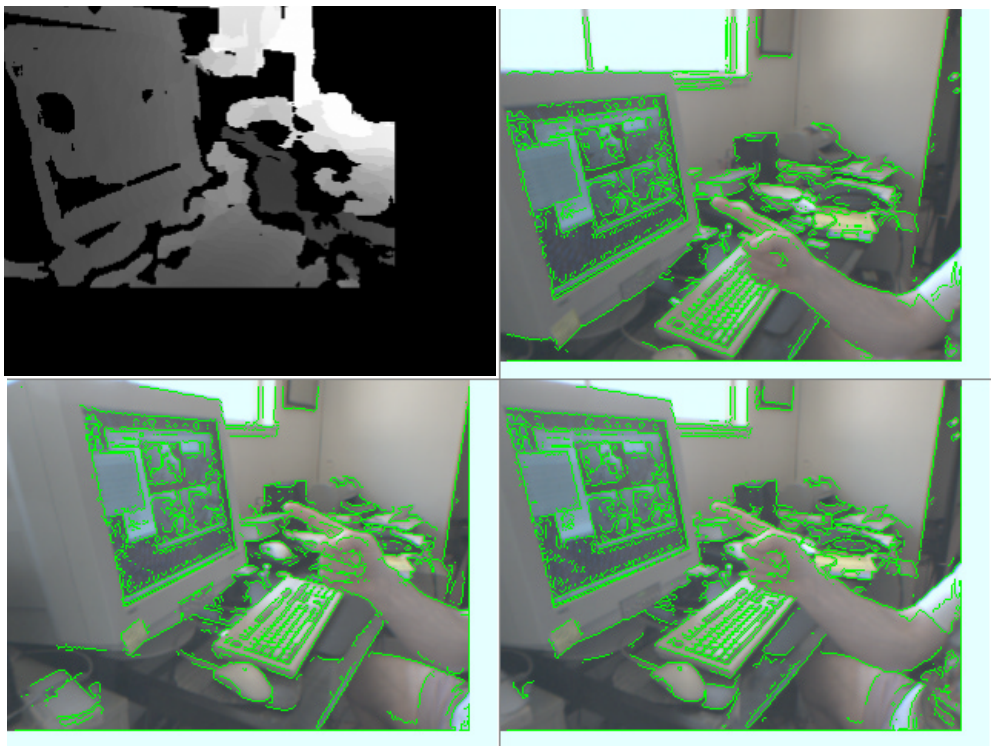

FIGURE 35. Edge and range images for a complex background

The background of this image has several parallel edges on the keyboard that are, by design, about the width of a finger. There is clutter on the screen, and there are strong edges behind the finger that, with an edge detector like that by Shen and Castan (used here), cause a break in the weaker edge of the finger where

it is suppressed near the stronger edge in the background. The Canny edge detector has the same issue.

The system correctly identifies the finger, which is shown in Figure 36.

FIGURE 36. Complex backgrounds with many edges[1]

1. The model lines in this image were hard to see so I drew over them in red by hand. This did not alter the location of the model in the image.

5.2.2 Moving backgrounds

In many situations there will be movement in the background: a shadow, another person or object, or just some moving object. The images in Figures 37 and 38 have such a background.Between the first and second frames, the background has significantly changed, which would cause problems for some schemes, particularly ones that depend heavily on background subtraction. Because the evidence channel images are purely a function of the current images and no previous ones, the system is quite immune to background motion issues.

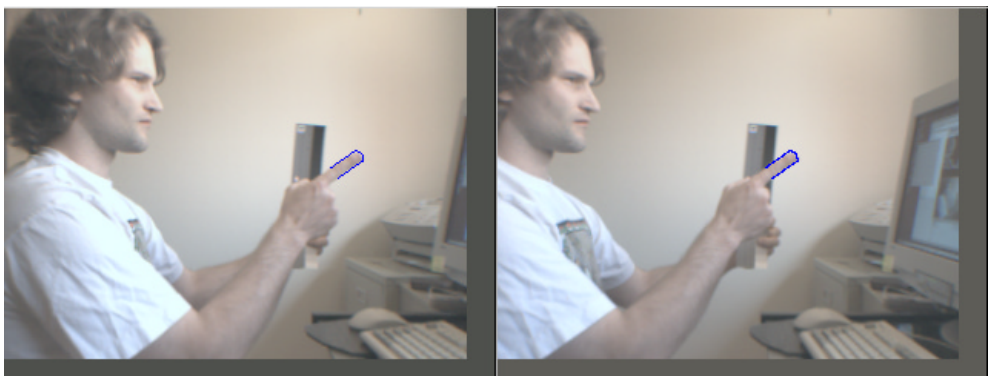

FIGURE 37. First image in moving background sequence

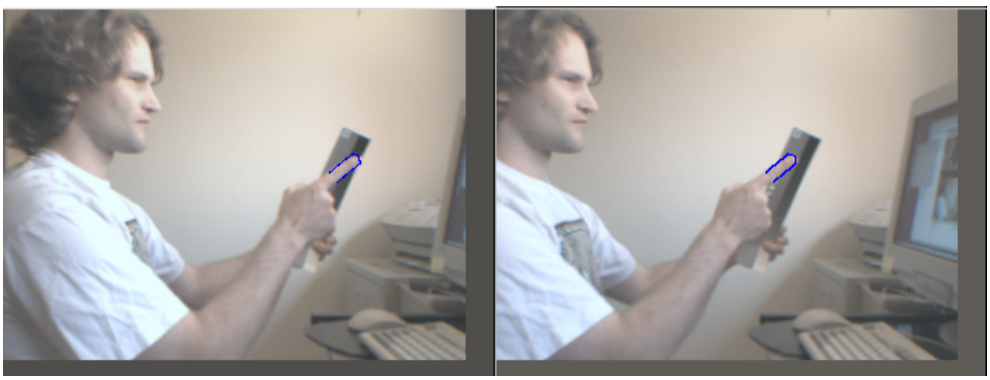

FIGURE 38. Second image in moving background sequence

5.2.3 Environments with skin colored backgrounds

Office environments often have inanimate objects with colors very similar to skin. These may be wood furniture and doors, cardboard boxes, or even off-white walls. Figures 39 and 40 have a skin-colored shirt.

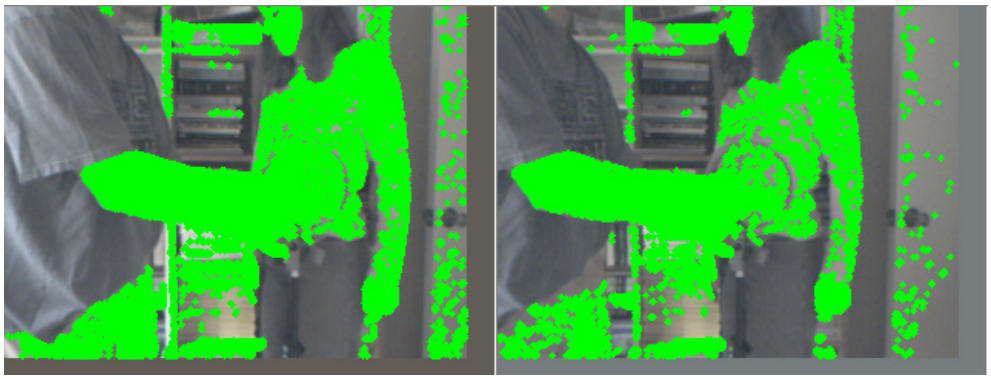

FIGURE 39. Skin detection result

FIGURE 40. Model detection with skin-colored background

In this example, the skin detection and convexity channels completely fail, but the system uses other cues to converge on the correct model. Color is a great cue for vision systems, but there are always cases where it fails, necessitating other ways for the system to keep working.

5.2.4 Changing lighting conditions

Changing light conditions can result from such things as fans that cast moving shadows, monitors that cast different colors of light as windows are iconized or reappear, or the sun going behind a cloud. All of these conditions have complicated experiments during this work. Figure 41 shows the left and right images for three frames in this sequence. Note that the lower images in the sequence are darker: the blinds of a nearby window were closed. This was done with the AGC in the camera turned off.

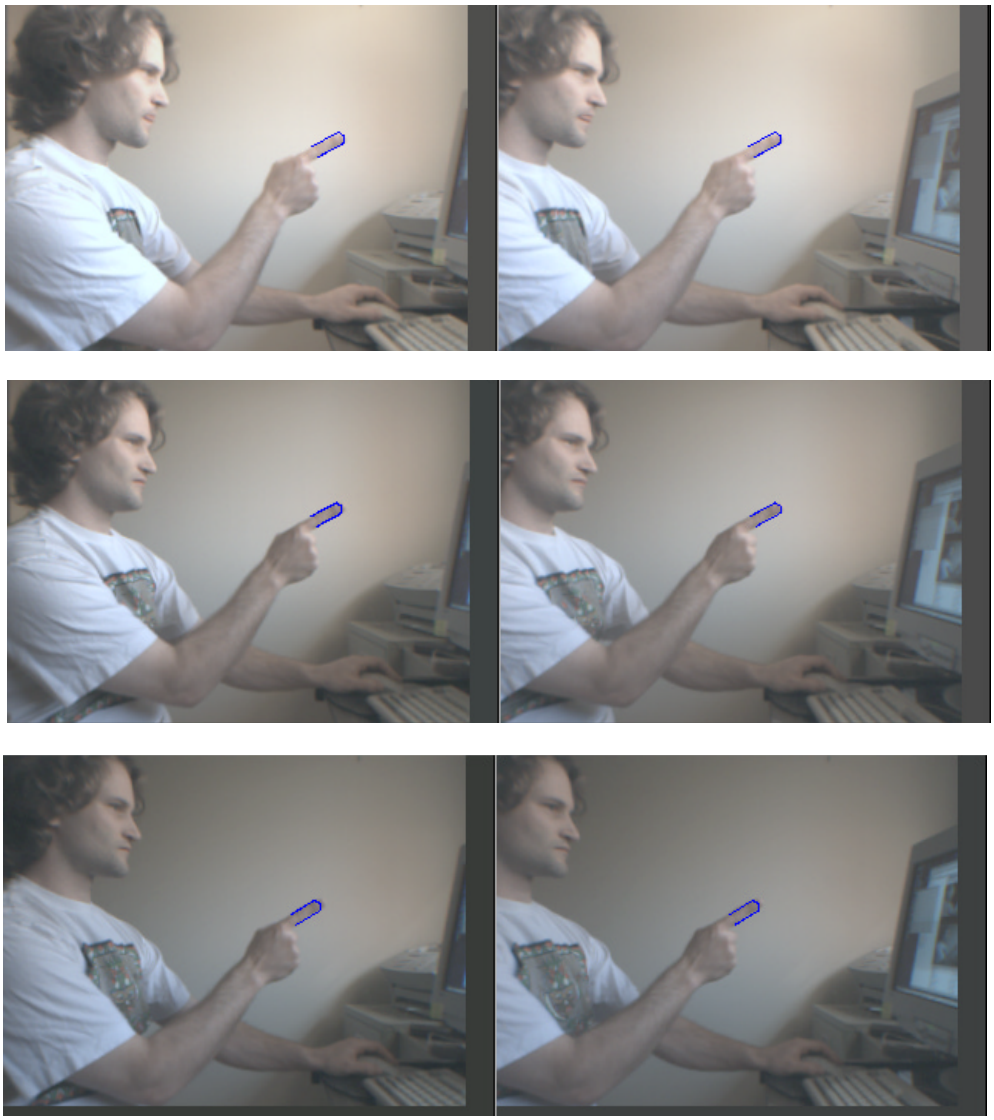

FIGURE 41. Frames from changing lighting test sequence

The evidence channels used in this work - color, edges, convexity, and range images - are all fairly robust to changes in lighting, so the resulting tracking works well across various lighting conditions.

5.2.5 Camera motion

Since the system does not use any sort of image subtraction, it is fairly robust to camera motion. A camera rotating even a small amount can cause a large apparent motion of the finger. Sometimes the finger leaves the tracked area, causing the system to search for it, but otherwise the system copes well with camera motion.

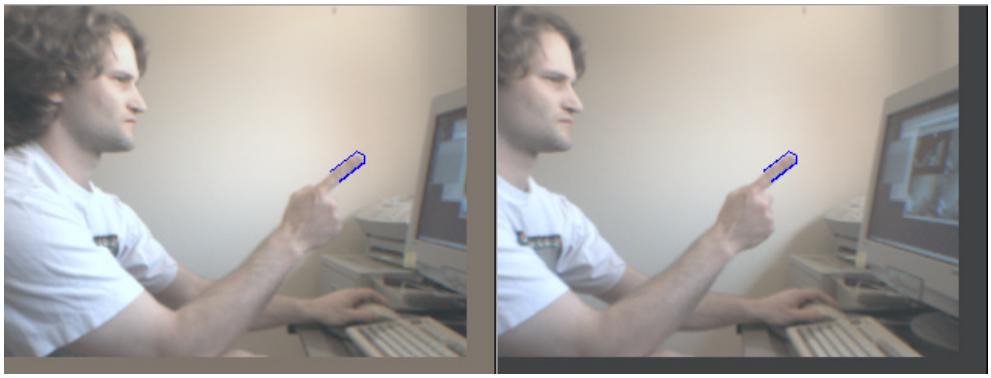

FIGURE 42. Images from first frame in camera motion sequence

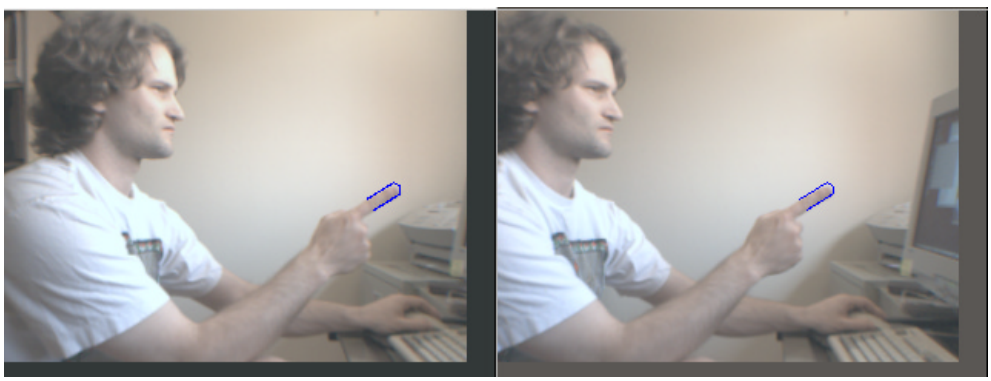

FIGURE 43. Left and right images from second frame

Figure 42 shows images from both cameras. The Digiclops was rotated about 10° between the two frames to get the images in Figure 43.

5.2.6 Backgrounds without texture

Backgrounds without texture information cause failure of the stereo matching in the background, but they still allow for accurate stereo information around the finger.

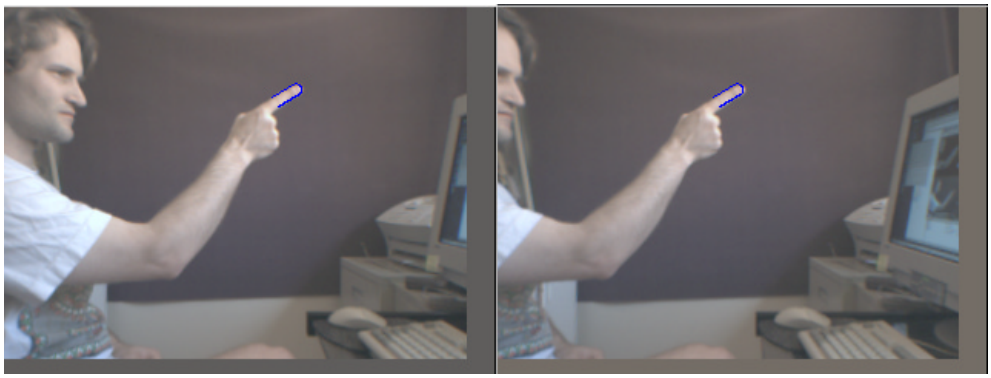

FIGURE 44. Background lacking texture

In the scenes in Figures 44 and 45, a velvet backdrop essentially eliminates any stereo matches in the background.

FIGURE 45. Background lacking texture - stereo range image

The stereo range data in Figure 45 shows that there is no range data in the background. In this case, systems relying on background subtraction would have difficulty using the range image data to segment out background from foreground. In this case there is still reasonable range information for the finger, but even if

the range channel is totally destroyed (e.g. by putting a lens cap over the reference camera for the stereo), the system still works using the other evidence channels available from the other two cameras.

5.2.7 Various rotations

A sequence was run with a wide range of finger orientations, and the finger was found almost every time (see section 5.4 for a discussion of errors). Figures 46 through 48 show some interesting images from this sequence.

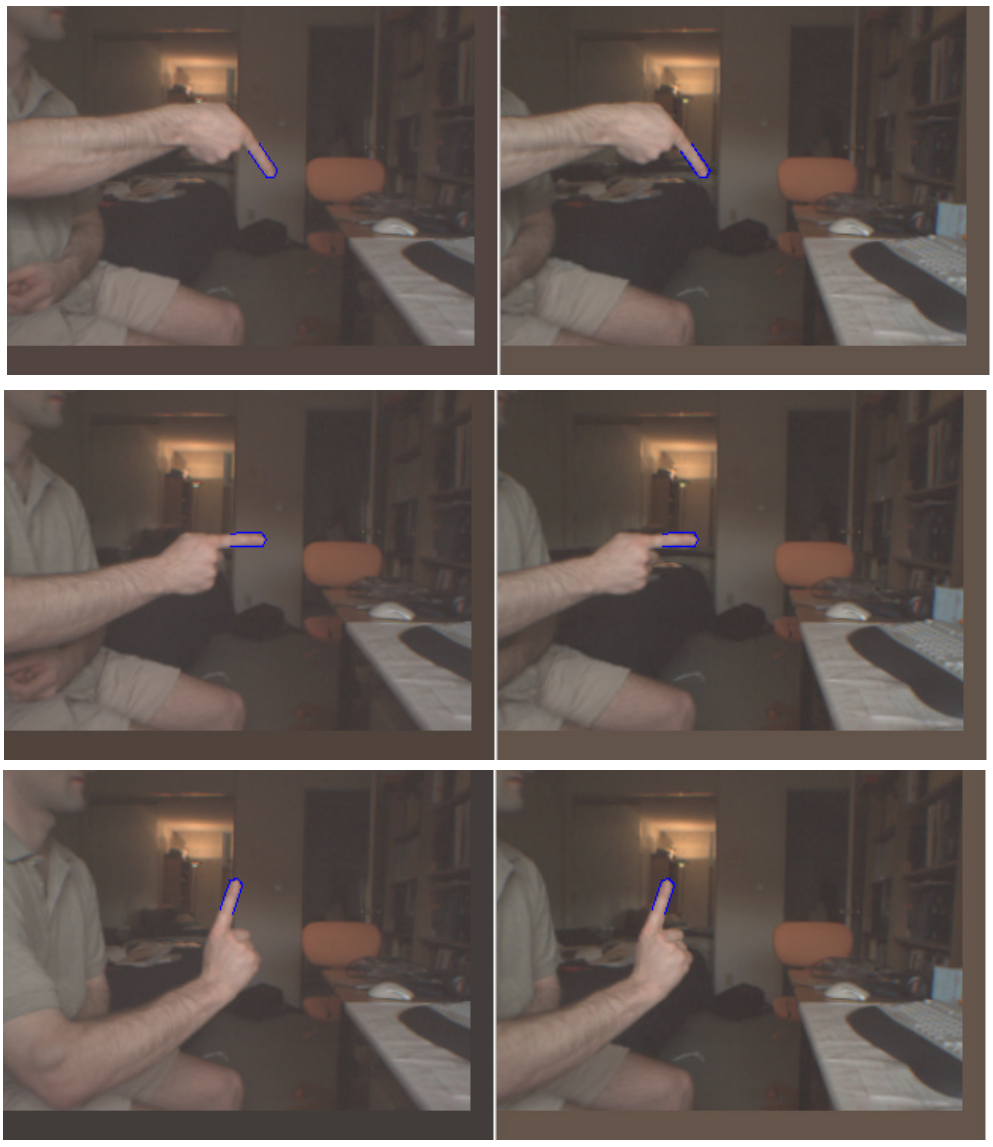

FIGURE 46. Finger rotations A

FIGURE 47. Finger rotations B

FIGURE 48. Finger rotations C

5.3 Multiple Fingers

All the results shown so far in this chapter have been generated by a program that chooses the most likely finger in the image and reports only that as the final result. To track multiple fingers, the program was modified to report results on all

fingers that had a likelihood above 10% after the non-maximal suppression of models that were within 5 mm and 5° in orientation of each other. An image with multiple fingers is shown in Figure 49.

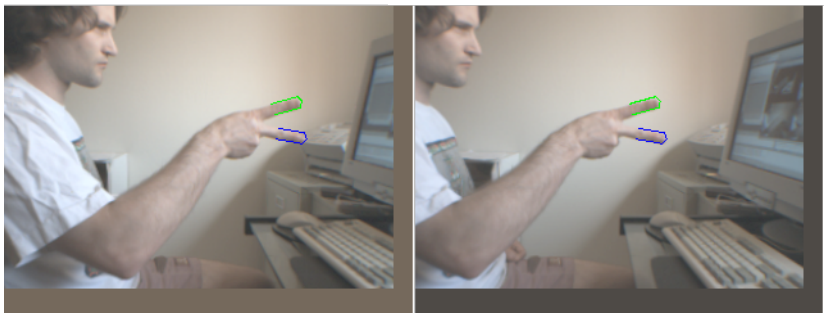

FIGURE 49. Sequence with two fingers

No advantage is taken here of the fact that the two detected fingers are connected to the same hand, which could in theory provide many constraints. This system can also track fingers from different hands, as shown in the sequence in Figure 50.

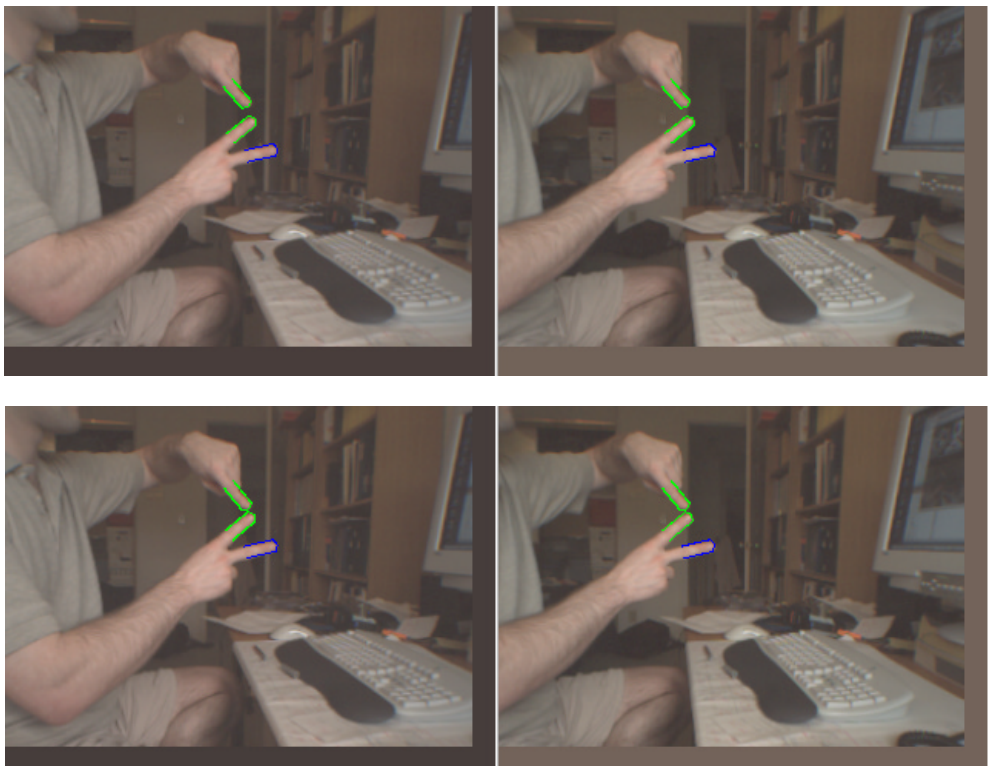

FIGURE 50. Sequence with multiple fingers

5.4 Errors

The model finding for the initialization sequence takes between two and 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 whose posterior probability of being a finger was less than 5%.

The system performs poorly in two situations. The first is when the finger is occluded by the arm or hand (see Figure 47). 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 (the system did detect the finger in this case in Figure 48, but the orientation is wrong). 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.

The system also assumes the cameras are calibrated and performs very poorly if they are not.

5.4.1 Example failure

The sequence shown in Figures 57 to 61 (in chapter 6) is very challenging. The system fails to track correctly in only one frame, shown in Figure 51.This frame

has several failing channels. The color channels fails because the shirt in the background got classified as skin. This also caused the convexity channel to fail. The low contrast between the finger and background eliminated much of the edge information for the finger. The combined result of all these failures was that the system did not find a likely position for a finger.

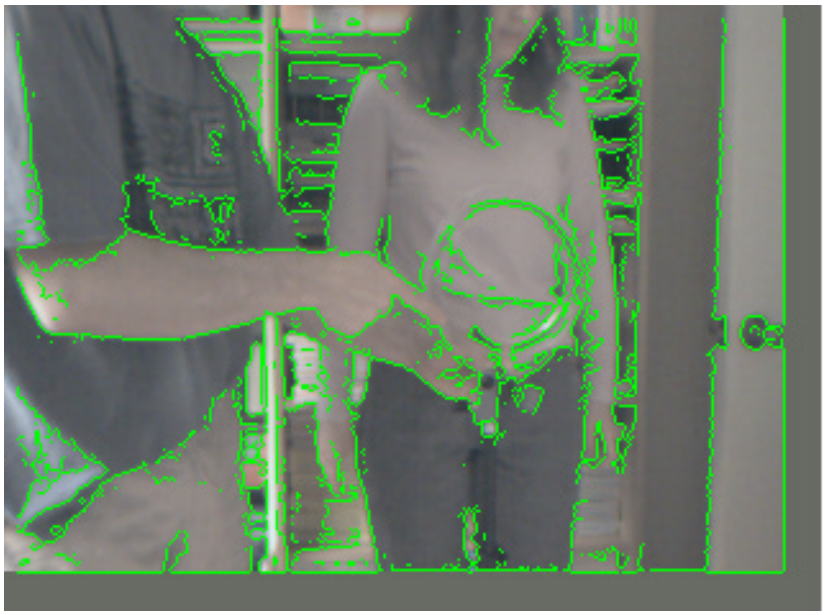

FIGURE 51. Edge image from failed frame

In the frame immediately after this one, the system found the correct result and continued tracking.

5.5 Accuracy

Two accuracy experiments were done. The first involved collecting a grid of points in a known coordinate system and comparing those to the values produced by the system. The second experiment involved collecting a set of data on a circle, fitting a circle to the data, and measuring the deviation of the data from the circle.

5.5.1 Grid experiment

The setup for the first experiment is shown in Figure 52. There is a Digiclops camera in the far right, and a coordinate grid has been drawn on the paper on the table. The position of the camera was measured, and the camera was rotated until the principal optical axis aligned with the coordinate system. This alignment was done by looking at the output images and rotating the camera until a block at the origin of the coordinate system was in the center pixel of the resulting image. The accuracy of the rotation lined up the optical axis such that it had an error of less than one pixel, which corresponds to about 0.16°.

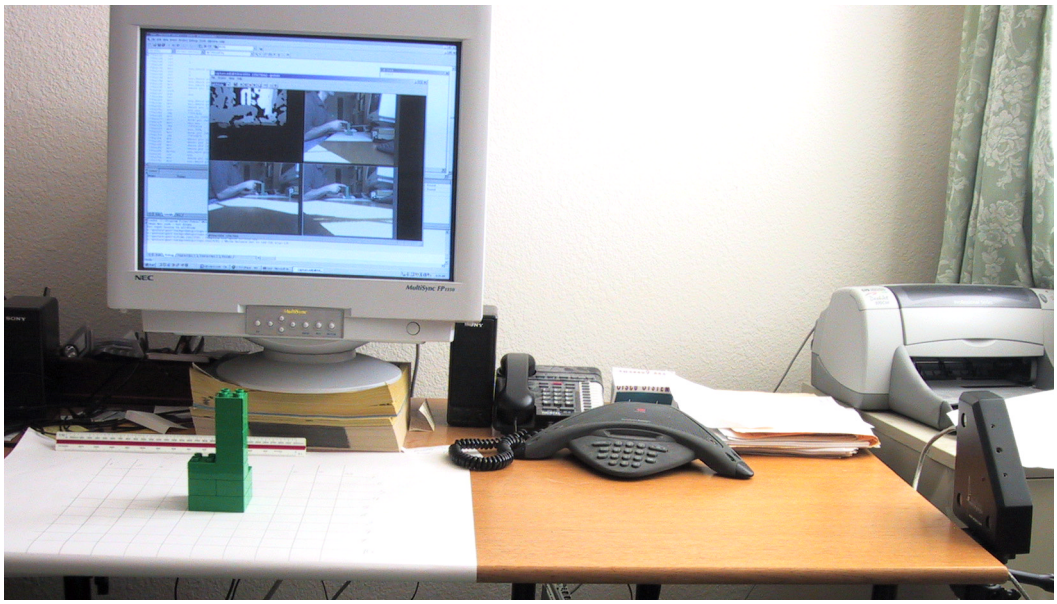

FIGURE 52. Accuracy experiment setup

In this experiment, the table lies in the x-z plane, and the y axis corresponds to a vertical direction. The z axis aligns with the optical axis of the Digiclops and corresponds to the left to right axis in Figure 52. The x axis goes from the front to the back of the table. The uncertainty in the measurements of the ground truth locations is substantial. In the x and y plane, there is up to 5 mm of uncertainty regarding the positioning the finger and up to 5 mm of uncertainty in estimating the center of the camera. The z direction has the same 5 mm uncer-

tainty in the positioning of the finger, but because of the difficulty in estimating the location of the virtual pinhole of the camera, the uncertainty in the camera position is greater, up to 8 mm. This results in ground truth measurements that are ± 10 mm along the x and y coordinates and ± 13 in z.

Once the Digiclops was aligned, a 10 by 10 grid of points was sampled at two different heights above the table. The grid spacing in the x-z plane was 30 mm, and the grid was sampled at heights of 0 mm and 95 mm. The three images for one of the samples is shown in Figure 53.

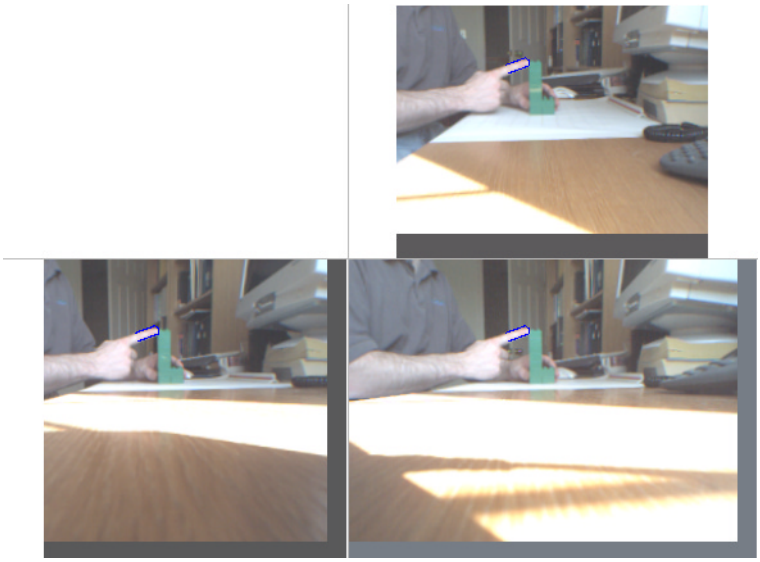

FIGURE 53. Image from accuracy sequence

The sample points of all the data from the taller height are shown in Figure 54 projected into the x and z planes. Errors in the y plane are very similar to the x plane, but the graph is too confusing in 3D. The measured points are shown as red dots. The ground truth location, including the error bars for the ground truth, are shown as gray crosses.

The plot shown in Figure 54 illustrates the difficulties with this experiment. The accuracy of the ground truth is so poor that it makes it difficult to tell how well the data fits.

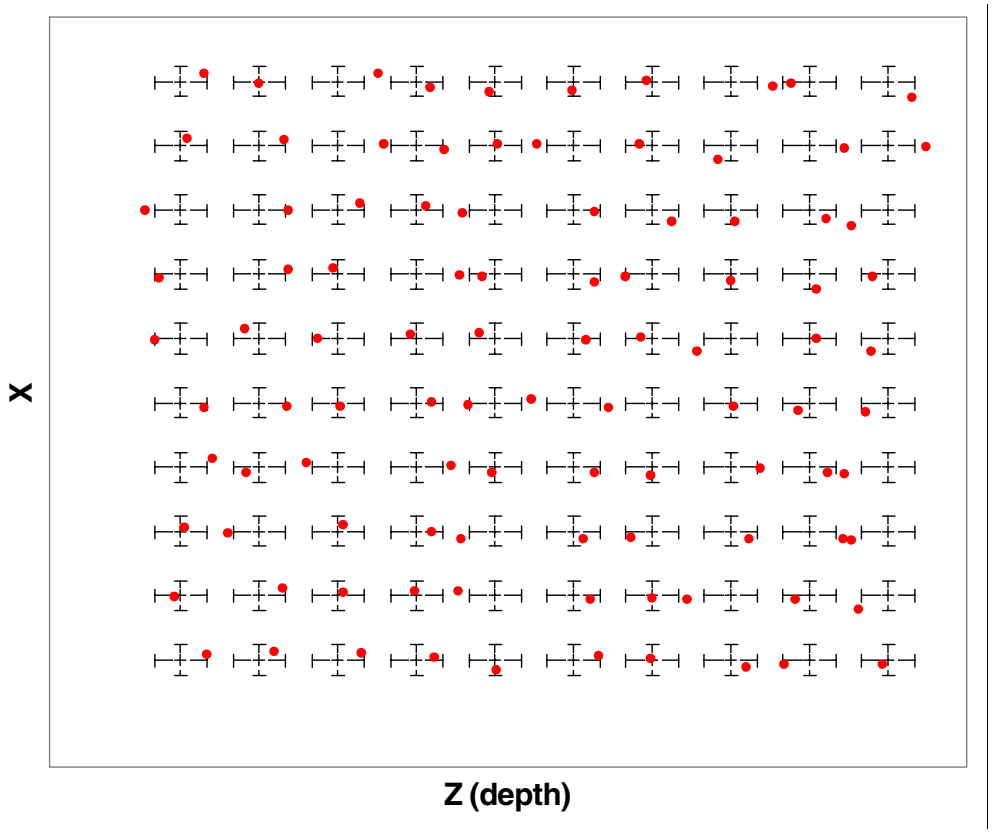

FIGURE 54. 2D plot of finger tip locations

From this data, the average errors along the x, y, and z coordinates were found to be 2.4, 2.3 and 8.2 mm respectively. The average errors were less than the uncertainty of the ground truth. Most of the uncertainty in the ground truth came from estimating where the true center of the camera was.

5.5.2 Circle experiment

The second accuracy experiment tries to address the problem of knowing where the camera is by using all the data collected to estimate its location and orientation, and then, given this information, measuring the accuracy of the result.

The accuracy of the system was measured by mounting the lid of a glass bowl in front of the camera. The edge of the lid is a circle and has a ridge that is very convenient for tracing the tip of a finger. I ran the tip of my finger around the lid and gathered the results. This data was fitted to a circle of the correct radius, and then the distance from the sample points to the fitted circle was computed to measure the error. This approach to accuracy measurement was motivated in part by the work of Kakadiaris and Metaxas [36]. A 3D plot of the results is shown in Figure 55.

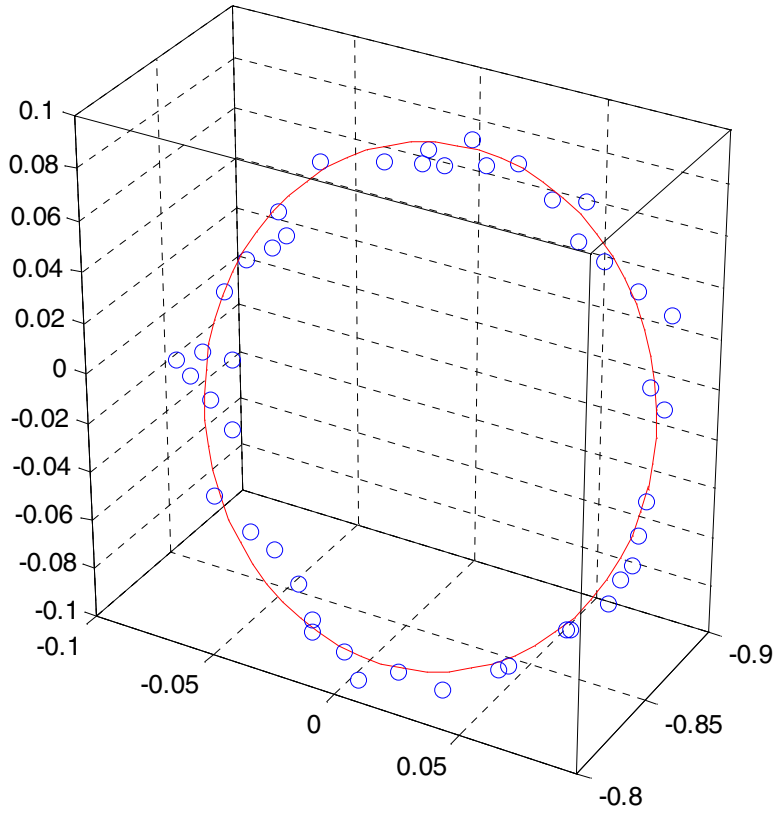

FIGURE 55. Plot of finger tip location and circle fit to it

Projecting this data into 2D front and side views (Figure 56) makes it easier to see the results. Clearly the system is less accurate in its depth measurements.

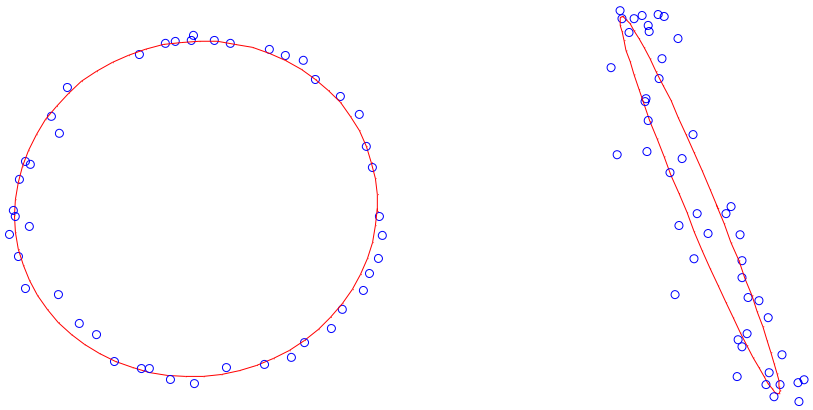

FIGURE 56. Front and side view of finger tip plot

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. The average error moving in the plane of the camera was 2.7mm. On the depth axis (moving away from the cameras) the accuracy is worse, with an average error of 9mm. Given the narrow baseline and wide field of view of the cameras, the 9mm error represents an error below 0.4 pixels in the image. This subpixel accuracy is possible because the whole finger model ends up fitting to many edge pixels that contribute to the result. It will be hard to improve on these results; 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°. The accuracy data was collected from a very hard sequence that shows a finger being tracked over a skin colored region and against a background with motion and many edges. The sequence is shown in chapter 6.

Given the size of the work space, the system's accuracy is about 99%. If the depth component is ignored, the average accuracy is around 99.5%.

5.6 Speed

The system was run on a 1400 MHz Pentium 4 processor running Linux. The time varies up to 10% for a given frame, but the average time per frame is shown in Table 1. Frames where the track is lost and the system must reinitialize the tracking take about 590 ms.

TABLE 1. Average computation time per frame

Task	Time (ms)
Capture, scaling, unwarping, and image correction	77
Stereo processing	48
Edge detection	17
Convexity features	1
Skin detection	12
Model fitting and searching	32
Miscellaneous	26
Display	43
Total	**257**

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 eight. 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.

The size of the tracking regions of interest was set so that the finger can move at about 40 cm/s and not get lost. If it moves faster than this, the system

loses the tracking and slows down to a bit under 2 fps as it searches the whole work space for the finger.

The system can also be configured such that the number of models tried or the size of the search space is dynamically adjusted to achieve the desired frame rate. A program like this is excellent for some real time applications in which such an update is needed - every second, say. The system will give the best result possible within this computation time constraint. Conveniently, if exactly the same software is later run on a newer, faster computer, the system will produce better results and still meet the real time constraint.

For multiprocessor systems, it is easy to pipeline the processing so that some of the early stages happen on one processor and other stages on different processors. All of the steps, including the model fitting, can easily be made parallel to run on multiple processors. The general architecture of tracking systems based on this work will be well suited for real time implementation.

(This page intentionally left blank.)

6. Conclusion & Future Work

This thesis has shown useful techniques for combining very different measurements to form a robust system. It combines this information using a Bayesian approach but also incorporates specific considerations for instances when an input is failing to improve robustness. A characteristic of this system that is critical to its success is that it does not ignore the nature or extent of the uncertainty in various measurements. It combines a least squares local optimization approach with a particle filter type approach to find the globally optimal solution. It uses inputs of stereo range images, color, and edges that would apply to many tracking problems, as well as a convexity feature that is useful in the particular case of tracking fingers.

To illustrate the use of the theoretical approach, a practical vision system was constructed. This work has resulted in a highly robust system that can accurately track a finger in three dimensions. It can deal with the non-rigidity of fingers, with motion in the camera, and with complex backgrounds containing motion, dense edges, skin colored objects, and varying lighting conditions. The system provides sub-pixel accuracy at speeds that are not far off real time.

The sequence shown in Figures 57 to 61 is very challenging. It reasonably demonstrates the considerable capabilities of this system. It demonstrates robust tracking in a real time environment against a background with complex noise, parallel edges about a finger-width apart, motion, skin colored objects, changing lighting - in short, the sorts of things encountered in real world vision problems. The results are robust and provide 3D position information that is very accurate - in fact the system provides sub pixel level accuracy. Some of the error measurements discussed in the previous chapter come from this sequence.

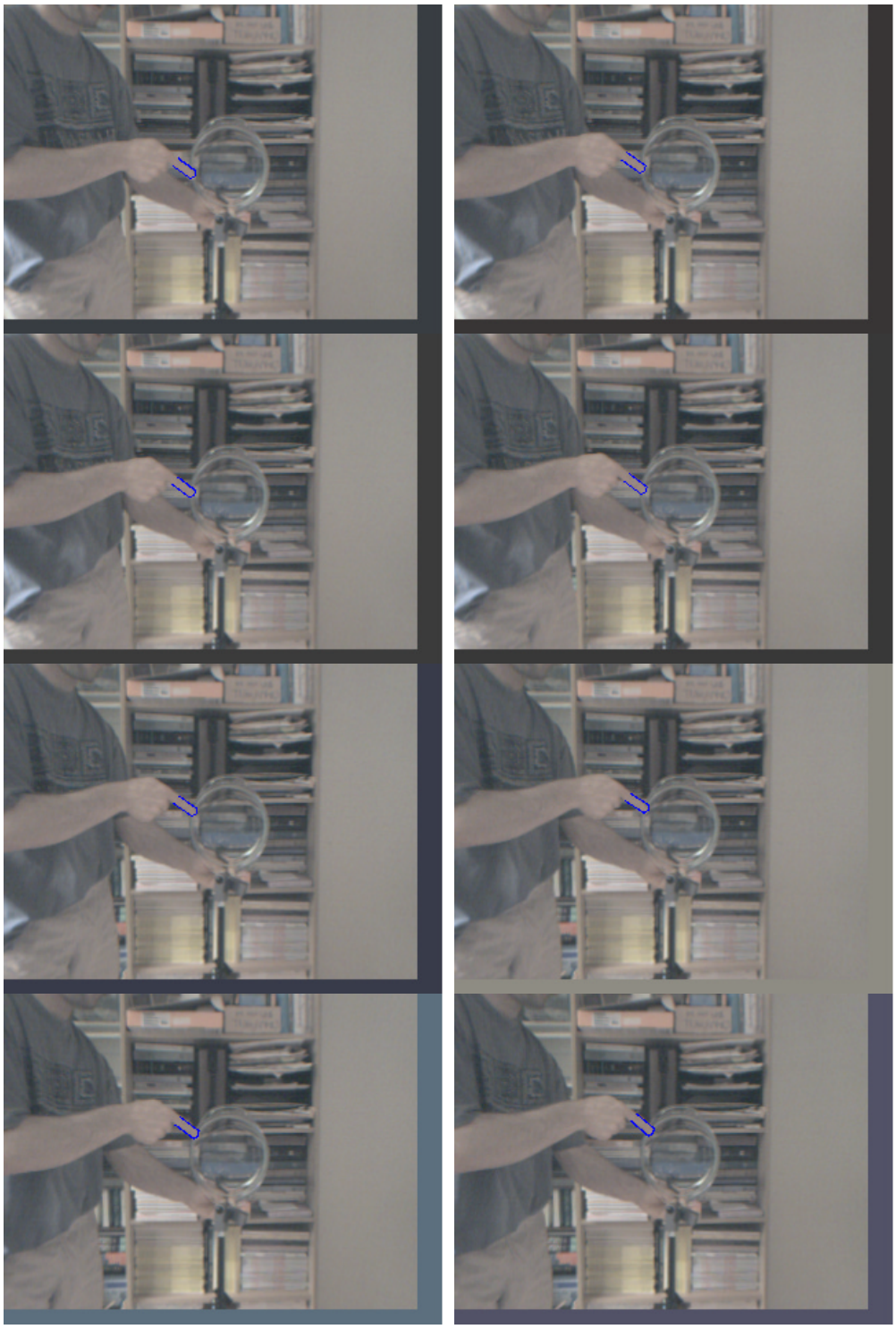

FIGURE 57. Image sequence - part A

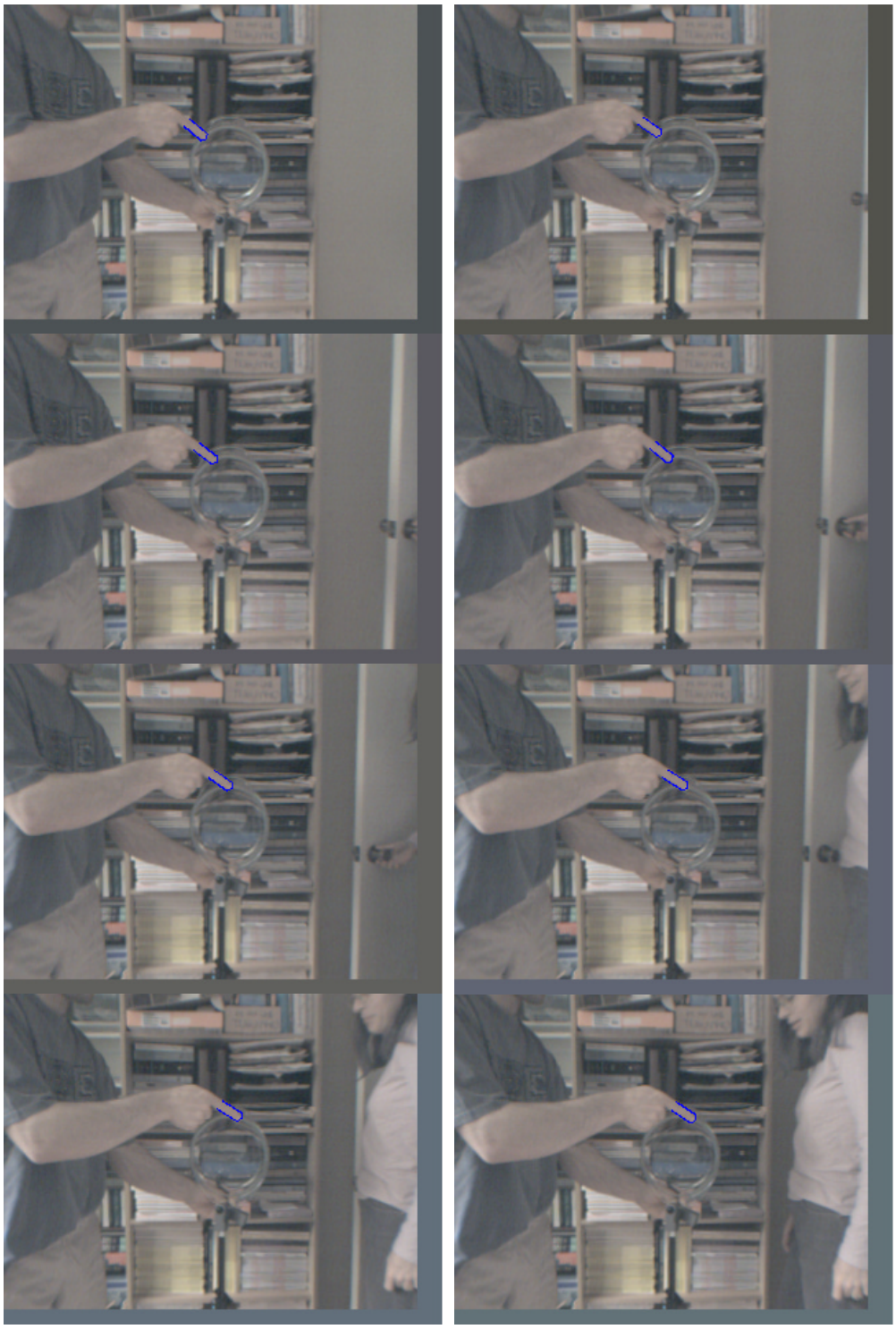

FIGURE 58. Image sequence - part B

FIGURE 59. Image sequence - part C

FIGURE 60. Image sequence - part D

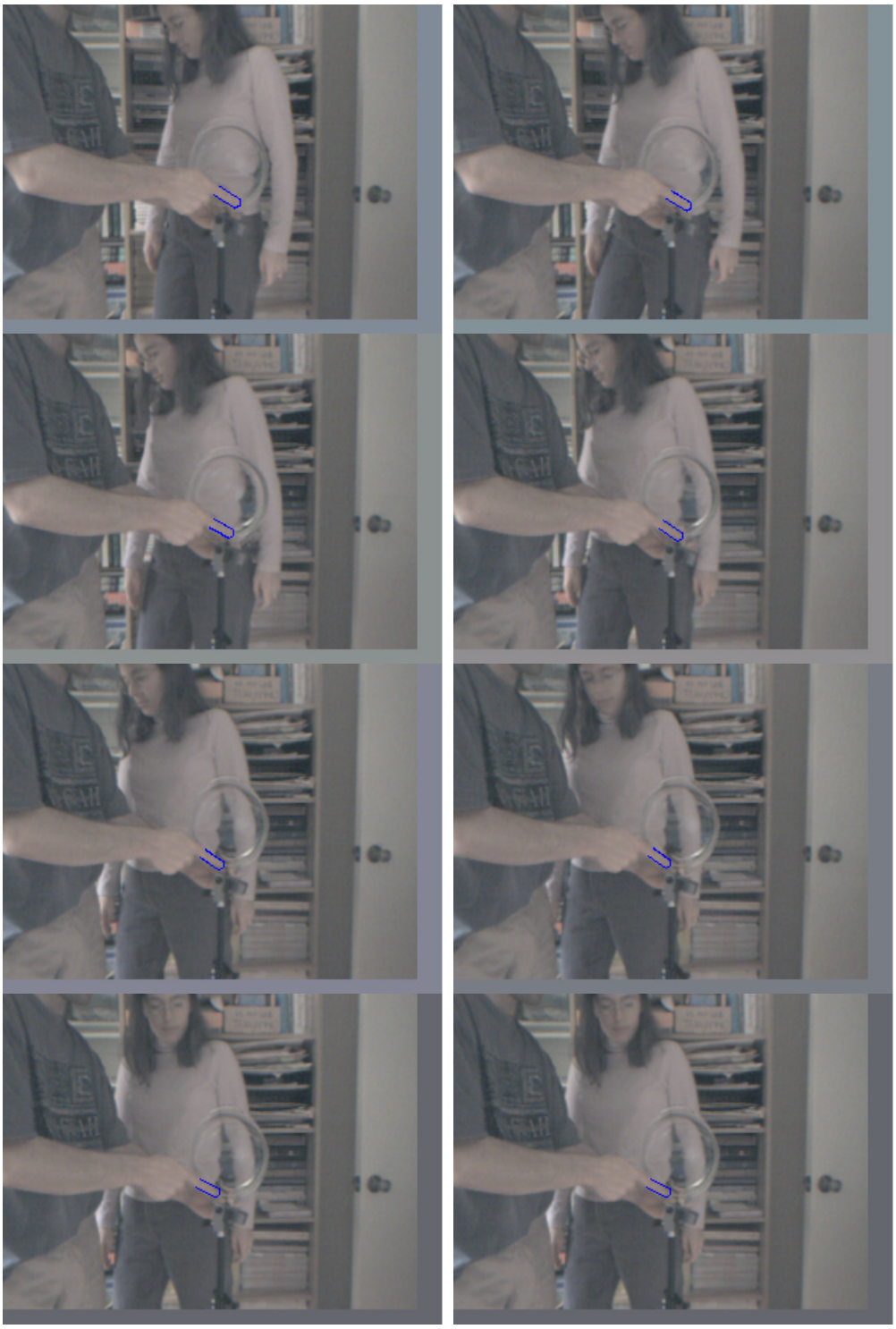

FIGURE 61. Image sequence - part E

Some of the key ideas coming out of this work that can be applied elsewhere include:

- simultaneously combining information sources;
- paying attention to errors;
- acknowledging failures of input information in the sensor model;
- using stereo range images for recognition problems;
- using a gradient approach for local optimization;
- performing a randomized search for global optimization;
- using sequential Monte Carlo to propagate priors forward;
- combining the local optimization into the SMC step to accelerate SMC in finding the nodes of the distribution; and
- the overall engineering framework for building a complex vision system.

These techniques can be applied to a wide range of complex tracking problems.

Future work falls into three major categories: improving the technique, using this work for other problem domains, and improving the finger tracking system described here.

In the theoretical dimension, future work could look at using factorization approaches to particle filtering to propagate information from one frame to the next. It might also be profitable to look at better ways to detect when a channel is failing and how to feed that information into the system.

In developing practical systems based on this work, many applications could be explored. The system would work for head tracking, for finding stop signs in images taken from moving vehicles, and for non-vision based tasks like finding boundary layers between different types of rock in seismic processing. Many tracking problems in such domains as games, engineering and medicine could be approached this way.

Future work on this particular finger tracking system could 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 has been found and could then be used for finding similarly colored regions in subsequent frames. Looking at adaptively choosing the best size for the region of interest in tracking is another area.

7. References

[1] E. Angelopoulou, R. Molana, K. Daniilidis, "Multispectral Skin Color Modeling," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 635-642, 2001.

[2] A. Azarbayejani, C. Wren, and A. Pentland, "Real-Time 3-D Tracking of the Human Body," Technical Report 374, Media Lab Perceptual Computing Section, Massachusetts Inst.of Technology, 1996.

[3] N.I. Badler and S.W. Smoliar, "Digital Representations of Human Movement," *ACM Computing Surveys*, vol. 11, no. 1, pp. 19-38, 1979.

[4] M.J. Black and A.D. Jepson, "EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation," *Proc. European Conf. Computer Vision,* pp. 1-14, 1996.

[5] A. Blake, and M. Isard, "3D Position, Attitude and Shape Input Using Video Tracking of Hands and Lips," *Computer Graphics Proc. (SIGGRAPH),* pp. 185-192, 1994.

[6] T.-J. Cham and J. M. Rehg, "A Multiple Hypothesis Approach to Figure Tracking," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 239-245, 1999.

[7] K. Choo and D.J. Fleet, "People Tracking Using Hybrid Monte Carlo Filtering, *Proc. Eighth IEEE Int'l Conf. Computer Vision,* vol. 2, pp. 321-328, 2001.

[8] R. Cipolla, and N.J. Hollinghurst, "A Human-Robot Interface Using Pointing with Uncalibrated Stereo Vision," *Computer Vision for Human-*

Machine Interaction, R. Cipolla and A. Pentland, eds., pp. 97-110, Cambridge University Press, 1988.

[9] D. Crisan, "Particle Filters - A Theoretical Perspective," *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, eds., pp. 17-41, Springer, 2001.

[10] J. L. Crowley, J. Coutax, and F. Bérard, "Things that See," *Comm. ACM*, vol. 43, no. 3, pp. 54-64, Mar. 2000.

[11] T. Darrell, G. Gordon, M. Harville, and J. Woodfill, "Integrated Person Tracking Using Stereo, Color, and Pattern Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 601-608, 1998.

[12] T.J. Darrell and A.P. Pentland, "Space-Time Gestures," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 335-340, 1993.

[13] T. Darrell, G. Gordon. J. Woodfill, and M. Harville, "A Virtual Mirror Interface Using Real-time Robust Face Tracking," *Proc. Third Int'l Conf. Automatic Face and Gesture Recognition,* pp. 612-621, 1998.

[14] J. Davis and M. Shah, "Visual Gesture Recognition," *IEEE Proc. Vision, Image and Signal Processing,* vol. 141, pp. 101-106, 1994.

[15] J. Deutscher, A. Blake, and I. Reid, "Articulated Body Motion Capture by Annealed Particle Filtering," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 126-133, 2000.

[16] A. Doucet, N. de Freitas, and N. Gordon, "An Introduction to Sequential Monte Carlo Methods," *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, eds., pp. 3-14, Springer, 2001.

[17] M.M. Fleck, D.A. Forsyth, and C. Bregler, "Finding Naked People," *Proc. European Conf. Computer Vision*, pp. 593-602, 1996.

[18] W. T. Freeman and C. D. Weissman, “Television Control by Hand Gestures,” *Proc. First Int’l Workshop Automatic Face and Gesture Recognition,* 1995.

[19] D.M. Gravila, “The Visual Analysis of Human Movement: A Survey,” *Computer Vision and Image Understanding*, vol. 73, pp. 82-98, 1999.

[20] C. Harris, “Geometry from Visual Motion,” *Active Vision*, A. Blake and A. Yuille, eds., pp. 263-284, MIT Press, 1992.

[21] C. Harris, “Tracking with Rigid Models,” *Active Vision*, A. Blake and A. Yuille, eds., pp. 59-73, MIT Press, 1992.

[22] T. Heap, and D. Hogg, “Towards 3D Hand Tracking Using a Deformable Model,” *Proc. Second Int’l Conf. Automatic Face and Gesture Recognition,* pp. 140-145, 1996.

[23] D. Hogg, “Model-Based Vision: A Program to See a Walking Person,” *Image and Vision Computing*, vol. 1, no. 1, pp. 5-20, 1983.

[24] D.P. Huttenlocher, J.J. Noa, and W.J. Rucklidge, “Tracking Non-Rigid Objects in Complex Scenes,” Technical Report CUCS TR-92-1320, Computer Science Dept., Cornell University, 1992.

[25] Y. Iba, “Population Monte Carlo Algorithms,” *Trans. Japanese Society Artificial Intelligence,* vol. 16, no. 2, pp. 279-286, 2001.

[26] M. Isard and A. Blake, “Contour Tracking by Stochastic Propagation of Conditional Density,” *Proc. European Conf. Computer Vision,* pp. 343-356, 1996.

[27] M. Isard and A. Blake, “Icondensation: Unifying Low-Level and High-Level Tracking in a Stochastic Framework,” *Proc. European Conf. Computer Vision,* pp. 891-908, 1998.

[28] M. Isard and J. MacCormick, “BraMBLe: A Bayesian Multiple-Blob Tracker,” *Proc. Eighth IEEE Int’l Conf. Computer Vision,* vol. 2, pp. 34-41, 2001.

[29] K. Ishibuchi, H. Takemura, and F. Kishino, “Real Time Hand Gesture Recognition Using 3D Prediction Model,” *Int’l Conf. Systems, Man, and Cybernetics*, pp. 324-328, 1993.

[30] C. Jennings, “Robust Finger Tracking with Multiple Cameras,” *Proc. Int’l Workshop Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pp. 152-160, 1999.

[31] C. Jennings. “Video Switch,” *Circuit Cellar Ink,* pp. 32-38, April 1999.

[32] N. Jojic, B. Brummit, B. Meyers, S. Harris, and T. Huang, “Detection and Estimation of Pointing Gestures in Dense Disparity Maps,” *Proc. Fourth Int’l Conf. Automatic Face and Gesture Recognition,* pp. 468-475, 2000.

[33] M.J. Jones and J.M. Rehg, “Statistical Color Models with Application to Skin Detection,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 274-280, 1999.

[34] R.E. Kahn, M.J. Swain, P.N. Prokopowicz, and R.J. Firby, “Gesture Recognition Using the Perseus Architecture,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 734-741, 1996.

[35] I.A. Kakadiaris and D. Metaxas, “Model-Based Estimation of 3D Human Motion with Occlusion Based on Active Multi-Viewpoint Selection,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 81-87, 1996.

[36] I. Kakadiaris and D. Metaxas, “Model-Based Estimation of 3D Human Motion,” *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 12, pp.1453-1459, Dec. 2000.

[37] 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, Univ. of California at San Diego, 1994.

[38] I. J. Ko and H. I. Choi, “Extracting the Hand Region with the Aid of a Tracking Facility,” *Electronic Letters,* vol. 32, no. 17, pp. 1561-1563, Aug. 1996.

[39] C. Le Gal, J. Martin, A. Lux, and J.L. Crowley, “SmartOffice: Design of an Intelligent Environment,” *IEEE Intelligent Systems*, vol. 16, no. 4, pp. 60-66, Jul./Aug. 2001.

[40] M.H. Lin, “Tracking Articulated Objects in Real-Time Range Image Sequences,” *Proc. Seventh IEEE Int’l Conf. Computer Vision,* vol. 1, pp. 648-653, 1999.

[41] D. Lowe, “Fitting Parameterized 3-D Models to Images,” *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 5, pp. 441-450, May 1991.

[42] D. Lowe, “Robust Model-Based Motion Tracking Through the Integration of Search and Estimation,” *Int’l J. Computer Vision,* vol. 8, no. 2, pp. 113-122, 1992.

[43] J. MacCormick, “Probabilistic Modelling and Stochastic Algorithms for Visual Localization and Tracking,” Ph.D. thesis, Oxford University, 2000.

[44] J. MacCormick and A. Blake, “A Probabilistic Exclusion Principle for Tracking Multiple Objects,” *Proc. Seventh IEEE Int’l Conf. Computer Vision,* vol. 1, pp. 572-578, 1999.

[45] J.P. MacCormick and M. Isard, “Partitioned Sampling, Articulated Objects, and Interface-Quality Hand Tracking,” *Proc. European Conf. Computer Vision,* vol. 2, pp. 3-19, 2000.

[46] C. Maggioni and B. Kämmerer, “GestureComputer - History, Design and Applications,” *Computer Vision for Human-Machine Interaction*, R. Cipolla and A. Pentland, eds., pp. 23-51, Cambridge University Press, 1998.

[47] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan, “The Unscented Particle Filter,” *Advances in Neural Information Processing Systems*, T.K. Leen, T.G. Dietterich, and V. Tresp, eds., MIT Press, pp. 584-590, 2000.

[48] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan, “The Unscented Particle Filter,” Technical Report CUED/F-INFENG/TR 380, Engineering Department, Cambridge University, 2000.

[49] T.B. Moeslund and E. Granum, “A Survey of Computer Vision-Based Human Motion Capture,” *Computer Vision and Image Understanding*, vol. 81, pp. 231-268, 2001.

[50] P. Moral and J. Jacod, “Interacting Particle Filtering With Discrete Observations,” *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, eds., Springer, pp. 43-75, 2001.

[51] M. Okutomi and T. Kanade, “A Multiple-Baseline Stereo,” *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 4, pp. 355-363, April 1993.

[52] V.I. Pavlovic, R. Sharma, and T.S. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 7, pp. 677-695, Jul. 1997.

[53] A.R. Pope and D.G. Lowe, "Probabilistic Models of Appearance for 3-D Object Recognition," *Int'l J. Computer Vision*, vol. 40, no. 2, pp. 149-167, 2000.

[54] Y. Raja, S.J. McKenna, and S. Gong, "Tracking and Segmenting People in Varying Lighting Conditions Using Colour," *Proc. Third Int'l Conf. Automatic Face and Gesture Recognition*, pp. 228-233, 1998.

[55] C. Rasmussen and G.D. Hager, "Probabilistic Data Association Methods for Tracking Complex Visual Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 6, pp. 560-576, June 2001.

[56] J.M. Rehg and T. Kanade, "DigitEyes: Vision-Based Human Hand Tracking," Technical Report CMU-CS-93-220, School of Computer Science, Carnegie Mellon Univ., 1993.

[57] Y. Rui and Y. Chen, "Better Proposal Distributions: Object Tracking Using Unscented Particle Filter," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 786-793, 2001.

[58] R. Rosales, V. Athitsos, L. Sigal, and S. Sclaroff, "3D Hand Pose Reconstruction Using Specialized Mappings," *Proc. Eighth IEEE Int'l Conf. Computer Vision,* vol. 1, pp. 378-385, 2001.

[59] Y. Sato, Y. Kobayashi, and H. Koike, "Fast Tracking of Hands and Fingertips in Infrared Images for Augmented Desk Interface," *Proc. Fourth Int'l Conf. Automatic Face and Gesture Recognition,* pp. 462-467, 2000.

[60] J. Shen and S. Castan, “An Optimal Linear Operator for Step Edge Detection,” *CVGIP: Graphical Models and Image Processing,* vol. 54, no. 2, pp. 112-133, Mar. 1992.

[61] J. Sherrah and S. Gong, “Tracking Discontinuous Motion Using Bayesian Inference,” *Proc. European Conf. Computer Vision,* vol. 2, pp. 150-166, 2000.

[62] H. Sidenbladh, “Probabilistic Tracking and Reconstruction of 3D Human Motion in Monocular Video Sequences,” Ph.D. thesis, Royal Institute of Technology, Stockholm, 2001.

[63] H. Sidenbladh and M.J. Black, “Learning Image Statistics for Bayesian Tracking,” *Proc. Eighth IEEE Int’l Conf. Computer Vision*, vol 2, pp. 709-716, 2001.

[64] H. Sidenbladh, M.J. Black, and D.J. Fleet, “Stochastic Tracking of 3D Human Figures Using 2D Image Motion,” *Proc. European Conf. Computer Vision*, vol. 2, pp. 702-718, 2000.

[65] H. Sidenbladh, F. de la Torre, and M. J. Black, “A Framework for Modeling the Appearance of 3D Articulated Figures,” *Proc. Fourth Int’l Conf. Automatic Face and Gesture Recognition*, pp. 368-375, 2000.

[66] L. Sigal, S. Sclaroff, and V. Athitsos, “Estimation and Prediction of Evolving Color Distributions for Skin Segmentation under Varying Illumination,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 152-159, 2000.

[67] C.V. Stewart, “Robust Parameter Estimation in Computer Vision,” *Siam Rev.*, vol. 41, no. 3, pp. 513-537, 1999.

[68] G. Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, 1986.

[69] J. Sullivan and J. Rittscher, "Guiding Random Particles by Deterministic Search, *Proc. Eighth IEEE Int'l Conf. Computer Vision,* vol. 1, pp. 323-330, 2001.

[70] S. Thrun, "Bayesian Landmark Learning for Mobile Robot Localization," *Machine Learning*, vol. 33, no. 1, pp. 41-76, 1998.

[71] K. Toyama and G.D. Hager, "Tracker Fusion for Robustness in Visual Feature Tracking," *SPIE Int'l Sym. Intel. Sys. and Adv. Manufacturing*, vol. 2589, 1995.

[72] J. Triesch and C. von der Malsburg, "Robust Classification of Hand Postures against Complex Backgrounds," *Proc. Second Int'l Conf. Automatic Face and Gesture Recognition,* pp. 170-175, 1996.

[73] J. Triesch and C. von der Malsburg, "A System for Person-Independent Hand Posture Recognition against Complex Backgrounds," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 12, pp. 1449-1453, Dec. 2001.

[74] K. Turkowski, "Filters for Common Resampling Tasks," *Graphics Gems*, A.S. Glassner, ed., pp. 147-165, Academic Press Professional, 1990.

[75] P. Wellner, "Interacting with Paper on the DigitalDesk," *Comm. ACM*, vol. 36, no. 7, pp. 86-96, 1993.

[76] M. Werman and D. Keren, "A Bayesian Method for Fitting Parametric and Nonparametric Models to Noisy Data," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 5, pp. 528-534, May 2001.

[77] G. Wolberg, *Digital Image Warping*, IEEE Computer Society Press, 1990.

[78] A. Wu, M. Shah, and N. da Vitoria Lobo, “A Virtual 3D Blackboard: 3D Finger Tracking Using a Single Camera,” *Proc. Fourth Int’l Conf. Automatic Face and Gesture Recognition*, pp. 536-543, 2000.

[79] Y. Wu and T.S. Huang, “Capturing Articulated Hand Motion: A Divide-and-Conquer Approach,” *Proc. Seventh IEEE Int’l Conf. Computer Vision,* vol. 1, pp. 606-611, 1999.

[80] Y. Wu and T.S. Huang, “A Co-Inference Approach to Robust Visual Tracking,” *Proc. Eighth IEEE Int’l Conf. Computer Vision,* vol. 2, pp. 26-33, 2001.

[81] Y. Wu, J.Y. Lin, and T.S. Huang, “Capturing Natural Hand Articulation,” *Proc. Eighth IEEE Int’l Conf. Computer Vision,* vol. 2, pp. 426-432, 2001.

[82] M.-H. Yang, D.J. Kriegman, and N. Ahuja, “Detecting Faces in Images: A Survey,” *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 1, pp. 34-58, Jan. 2002.