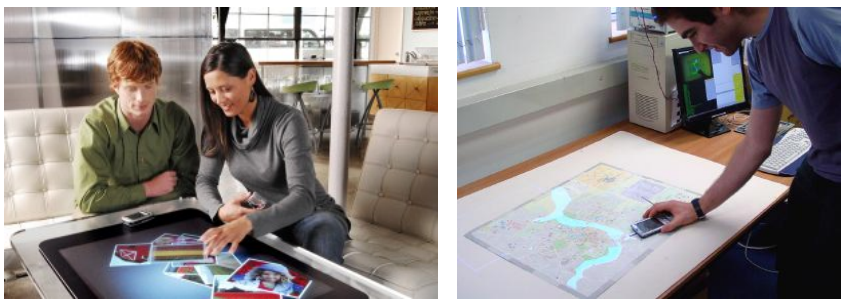
Contents

1 Introduction

Table-top environments are familiar as places of work, where they are often now augmented by a PC. This has proved effective for individual work, though has been plagued by issues of usability and is less effective for collaborative work. The paradigm of ubiquitous computing (Weiser, 1991) has been proposed as a means of tackling these problems, in which computing technology augments reality such that the computers fade into the background and become unnoticed.

Designing and building effective technology within this paradigm is still an active area of research (see Section 2.1); perhaps the two most common implementations have been the use of multi-touch screens as the table-top, and the combination of a computer vision system and projector to allow an ordinary table to become interactive. Examples of such systems are shown in Figure 1.

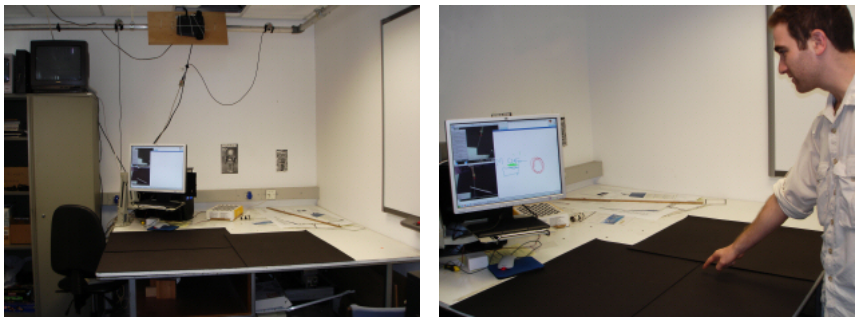
Recent advances in the field and the increased availability of computing hardware has led to the beginnings of commercialisation of such interfaces. Examples of this are Microsoft Surface and the various technologies of the companies Mindstorm, Mgestyk and GestureTek. These systems in general require specialist hardware systems and the cost of these has prevented widespread uptake thus far. The motivation for this project was to work towards systems that require only simple and affordable hardware available on the high street, as in the work of Reitmayr & Drummond (2005).



(a) Surface display (from Microsoft, 2009)

(b) Projection (from Reitmayr & Drummond, 2005)

Figure 1: Augmented table-top interaction systems



(a) System setup (b) System in use

Figure 2: Developed gesture-interaction system

In this project, a working interface was developed which used computer vision methods with cheap webcams to track multiple hands over a table-top. It identified hand-like objects in front of a background that was assumed stationary. The position and orientation of each hand was then extracted, with tracking using a Kalman filter allowing the hand velocity to be estimated. Characteristic patterns of motions pre-defined as canonical gestures were recognised by dynamic time warping. Finally, applications were built, allowing the system to be tested. The system can be seen in use in the photographs of Figure 2.

The remainder of this report details the development of this system and reports results. Section 2 describes previous work in the field and justifies the general approach taken. Section 3 then provides the technical framework for the system, which is referred to where it is applied later in the report. Section 4 provides a more detailed overview of the system, with a full description of the implementation given in Section 5. Finally, Section 6 reviews and evaluates the system and Section 7 gives possible extensions for the project and future directions for research in the area.

2 Previous work

2.1 Table-top environments

Tables can be defined, as by Mazalek et al. (2009), as physical products with smooth flat tops. Around these inauspicious objects, a wealth of human activities take place. They provide a surface on which arms can be rested and on which objects can be placed for interacting with. Traditionally, these have been fixed physical artefacts, but with modern technology, these artefacts can be changeable, augmented or even virtual, offering the potential for greater power and flexibility, as well as opening up table-top environments for direct interaction with computers.

Numerous approaches have previously been taken in developing new forms of table-top environment interfaces. One distinguishing factor is the sensing technology used in order to augment the table-top. One form is the placing of sensor technology within physical artefacts, for example pens. Song et al. (2009) use a sensing pen and projector to digitise the table-top, developing an application in architectural design. Similarly, Steimle (2009) also use a sensing pen 'Anoto', and recognise gestures from the motions of the pen to perform certain functions.

A more prevalent form is sensing table-top surfaces, often known as multi-touch surfaces. An overview of issues relating to these surfaces and designing interactions with them has been given by Buxton (2009), with recent examples involving both interaction with marked tangible objects on the surface (Kaltenbrunner & Bencina, 2007), (Mazalek et al., 2009) and by direct touch with hands (Wobbrock et al., 2009).

A final form is the use of vision for the sensing, with cameras suitably placed to view the table-top. Note that many of the sensing table-tops use the same technology but visually view the table from below, eliminating occlusion between the hands, at the cost of not allowing for interaction with the table covered. This form has also been used for interaction with tangibles (Reitmayr & Drummond, 2005), (Zufferey et al., 2009) and directly with hands (Taylor & Drummond, 2007), (Do-Lenh et al., 2009).

From these examples, it is clear that another distinguishing factor is the use of tangibles versus 'natural interaction' i.e. the use of the body directly. In many cases, systems that make use of tangibles require special tangible objects to be used, and while these may be as simple as visual markers on pieces of paper as in the work of Zufferey et al. (2009), in practice this may cause increased costs and setup time for the user.

With natural interaction, information is encoded by the motion of the person interacting, which with the more sophisticated means of sensing described above can go beyond treating this motion like the simple motion of a mouse and make use of gesture.

2.2 Gesture-based interaction

Gestures have been studied for human-computer interaction for well over twenty years now, as evidenced by early work such as that of Krueger et al. (1985). However, its use is still an active area of research due to difficulties in many aspects of the design of such systems. In using gestures for interface design, the two critical aspects of the design are the design of the gestures that the users themselves must perform (or at least the form these can take), and the design of a system to sense and recognise these gestures.

Considering the first aspect of the design, it is first necessary to clarify what is meant by a gesture. A simple definition is 'a motion of the body that contains information' (Kurtenbach & Hulteen, 1990). A rich set of body motions fits such a definition, and to aid understanding various gesture taxonomies have been developed. Billinghurst & Buxton (2008) develop a taxonomy in which six forms of gesture are described:

- Gestures that evoke a referent: symbolic (i.e. with an agreed meaning) and deictic (directing towards something in the environment)
- Gestures that depict a referent: iconic (describe an object) and pantomimic (describe how to use an object)
- Gestures that relate to the communication process: beat (communicate rhythm) and cohesive (communicate connections)

The types of gestures considered are important as they have different characteristics: symbolic and deictic gestures may often be stationary, whereas iconic and pantomimic

gestures often involve motion. Related are concerns of the usability of the system, whether it is comfortable to use and accessible to the users. This relationship comes about because gestures that are more comfortable to perform are often performed more accurately by the user. For example, Foehrenbach et al. (2008) choose an open hand point over a point with a finger extended due to cited literature providing evidence that this both requires less hand tension and can be used to point more accurately.

The second aspect of the design is how to sense and recognise these gestures. Using vision to sense gestures is well established for the reasons described in Section 2.1, despite problems including accuracy, ability to track fast movements and occlusion (Billinghurst & Buxton, 2008). An overview of hand gesture recognition with vision is given by Pavlovic et al. (1997), in which two key properties of such a system are described: time instant invariance and time scale invariance, as in general it does not matter when a gesture is performed or (perhaps within some limits) how long is taken in performing it.

Means of modelling gestures to be recognised visually are described by Konstantinos (2004), with the three main groups being model-based, view-based and low level feature-based approaches. Model-based approaches typically define an articulated model of the hand, and infer parameters for this model from the images. Gestures are then defined in terms of hand poses and their change over time. Examples include the work of Dorfmüller-Ulhaas & Schmalstieg (2001) which requires the user wear an exoskeleton glove but runs in real time and Wu et al. (2001) which takes a probabilistic approach and makes no mention of the speed of processing. As evidenced by these examples, the three key problems with this approach are the typically high number of parameters to be inferred, the difficulties of carrying out the inference in real time and the difficulty of determining appropriate visual features with largely textureless hands.

View-based approaches define gestures in terms of the data observations expected. A simple example is given by Ye, Corso & Hager (2004), with a more sophisticated approach for multiple simultaneous users given by Stenger et al. (2008). Using two cameras, a depth map can be used as the basis of a similar system as in Muñoz-Salinas et al. (2007). Key problems with these methods is ensuring invariance to affine transformations and

background, and providing adequate training data. One approach to achieving the desired invariance is to infer a canonical view as in the work of Grzeszcuk et al. (2000).

The final approach is the use of low level features. The aim of this is typically to increase speed and robustness, but there is also justification from the study of human gestures. Quek (1994) suggests that human observers also find precise hand pose movements difficult to detect as the hand moves, implying that is sufficient to determine gross hand motion when a hand is moving, and hand pose only when a hand is relatively stationary, to detect any form of gesture. Examples of work using low level features include the work of Maggioni & Kämmerer (1998), where the hand position is tracked while the user is wearing a glove and the work of Starner & Pentland (1998), where the centroid and principal axes of hands detected by colour segmentation are tracked. Stereo systems also take this approach: Malik & Laszlo (2004) use the stereo disparity to determine whether fingers are in contact with a specified surface on which gestures can be performed, and Leibe et al. (2000) determine 'fingertip' and 'shoulder' points to track where a single arm is pointing to. In all of these systems, the key difficulty is still hand detection in potentially changing background and lighting conditions.

Numerous approaches for building classifiers for gesture recognition have also been taken, including Hidden Markov Models, artificial neural networks and rule-based approaches, as described in Mitra & Acharya (2007) and Konstantinos (2004), with the approaches offering trade-offs between accuracy, speed and training time.

Overall a wide range of tools have been developed for visual gesture interaction systems. However, there are still challenging problems which have not been fully addressed, including hand detection and segmentation and gesture detection (i.e. differentiating a gestural from a non-gestural motion), which are often dealt with by making assumptions about environmental conditions (Konstantinos, 2004). There are also higher level issues that have not been addressed: most solutions do not make use of 3D information and do not allow for multiple users to interact simultaneously with the same system.

3 Technical background

This section provides an overview of the theoretical framework used in various aspects of the system and gives references to more thorough descriptions. The specific application details of these areas are given later in the report and refer back to this section.

3.1 Camera calibration

Camera calibration refers to the determination of the relationships between the geometry of real world objects and the geometry of these objects in camera images of them. These relationships are termed *projective* relationships, as they relate the geometry of 3D objects to 2D projections of those objects. The following describes the projection framework used in this project. Further details can be found in Cipolla (2006).

3.1.1 Homogeneous co-ordinates

A key concept widely used in projection theory is that of *homogeneous co-ordinates*. These are an alternative and equivalent way of writing co-ordinates which allow many equations in projection which are non-linear in Cartesian co-ordinates to be written linearly (Cipolla, 2006). Cartesian co-ordinates can be converted into homogeneous co-ordinates by adding an extra dimension and an arbitrary scale:

$$(X, Y, Z) \rightarrow (\lambda X, \lambda Y, \lambda Z, \lambda)$$

3.1.2 Extrinsic calibration

Extrinsic calibration determines the projective relationships which are independent of the camera used. This amounts to determining a transformation from real world co-ordinates to a set of camera-centric co-ordinates, as illustrated in Figure 3. This can be described very simply in homogeneous co-ordinates:

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.1}$$

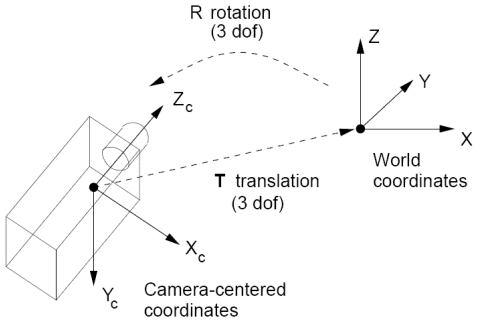


Figure 3: Extrinsic camera calibration (from Cipolla, 2006)

3.1.3 Intrinsic calibration

Intrinsic calibration determines the relationships between a camera-centric co-ordinate system and camera pixels. A pin-hole model for the camera is used, which defines the camera in terms of an *optical centre* (the 'pin-hole') and an upside-down image plane which the light which passes through the pin is projected onto. The upside-down image plane can be more conveniently represented by its mirror: an image plane between the optical centre and the real world, onto which the real world scene is projected. Using such a model, linear camera calibration amounts to positioning and scaling the image plane in camera-centric co-ordinates, as illustrated in Figure 4.

The first step is to project 3D positions onto the 2D image plane, taking perspective into account. Note that the inverse of this is that a pixel on the 2D image plane defines a *ray* through this point and the optical centre on which the real object colouring that

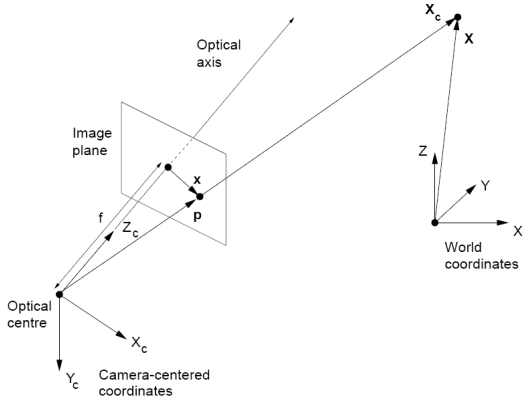


Figure 4: Intrinsic camera calibration (from Cipolla, 2006)

pixel must lie. This projection is described by the following equations (Cipolla, 2006):

$$x = f\frac{X_c}{Z_c} \qquad y = f\frac{Y_c}{Z_c}$$

Where f is the focal length of the camera and $\mathbf{x} = (x, y)^T$ for consistency with the figure.

The next step is to convert from the 2D image plane co-ordinates to pixel co-ordinates. This is described by the following equations:

$$u = u_0 + k_u x \qquad v = v_0 + k_v y$$

Where $\mathbf{w} = (u, v)^T$ are the pixel co-ordinates (from Cipolla, 2006).

In practice, cameras exhibit non-linear distortion. To account for this, a quintic camera model with greater scope for parametric tuning was used throughout this project, with the calibration described by the following equations (from TooN, Drummond et al.):

$$u = u_0 + f_u \frac{X_c}{Z_c}\left(1 + c\left|\begin{matrix}X_c/Z_c \\ Y_c/Z_c\end{matrix}\right|^2 + q\left|\begin{matrix}X_c/Z_c \\ Y_c/Z_c\end{matrix}\right|^4\right) \tag{3.2}$$

$$v = v_0 + f_v \frac{Y_c}{Z_c}\left(1 + c\left|\begin{matrix}X_c/Z_c \\ Y_c/Z_c\end{matrix}\right|^2 + q\left|\begin{matrix}X_c/Z_c \\ Y_c/Z_c\end{matrix}\right|^4\right) \tag{3.3}$$

Note that the inverse of this relationship, giving the transformation from image pixels to the image plane, cannot be found analytically. However, the solution can be found by iterative methods such as Gauss-Newton.

3.2 Epipolar geometry

In multiview systems, the relationships between what is seen in each camera can be understood through their epipolar geometry. A number of key concepts in epipolar geometry are shown in Figure 5:

- The *baseline* is the line joining the optical centres of the two cameras.
- An *epipole* is a point of intersection of the baseline with an image plane.
- The *epipolar plane* is a plane defined by the baseline and a point in the real world.
- An *epipolar line* is a line of intersection of the epipolar plane with an image plane.

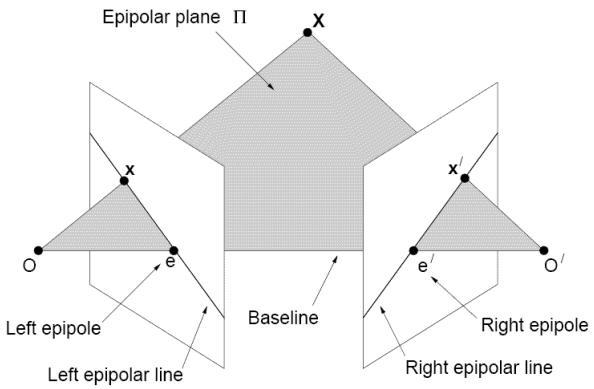


Figure 5: Epipolar geometry (from Cipolla, 2006)

Note that an epipole represents the projection of the optical centre of one camera onto the image plane of the other. Note also that any point on an epipolar plane, by definition, defines the same epipolar plane and thus the same epipolar lines. All other real world points define different epipolar planes, rotated by a certain angle about the baseline.

This last property can be used to define the *epipolar angle* as the angle the epipolar plane makes around the baseline. This angle can be used to uniquely define an epipolar plane and thus a pair of epipolar lines in an epipolar system. This angle thus parameterises the *epipolar constraint*, the constraint that any real world point must appear in each camera on epipolar lines corresponding to the same epipolar plane.

This epipolar constraint can be used to solve a key problem in multicamera systems, the problem of *correspondence* or *matching*: given multiple views of the same scene, which points in each view correspond to the same point in the real scene? The epipolar constraint constrains corresponding points to lie on corresponding epipolar lines.

In order to make use of this constraint for solid objects, it is necessary to identify points on each image of an object which correspond to a unique point on the real object. The epipolar constraint may then be used for matching. The global *epipolar tangencies*, the most extreme points on the real objects tangent to epipolar planes as illustrated in Figure 6, are suitable for this purpose as these are uniquely identifiable in each image as the points on the most extreme images of the object which are tangent to epipolar lines.

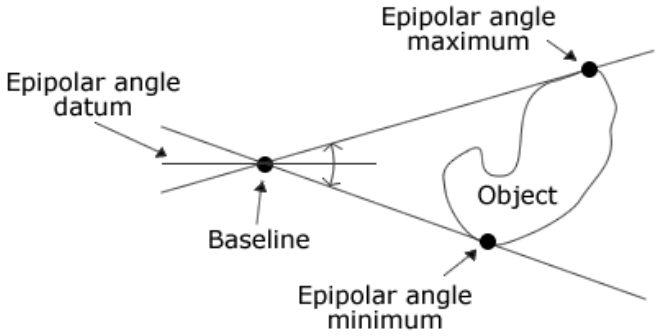


Figure 6: Global epipolar tangencies

Equivalently, by parameterising each epipolar line (and thus each pixel which lies on that line) by the corresponding epipolar angle, the points of epipolar tangency are those in the image of the object with maximum and minimum epipolar angles, and hence they may also be referred to as *global epipolar angle extrema*.

Global epipolar tangencies are especially useful because, although they may be occluded by other objects, they cannot be self-occluded. However, further information may also be gained by considering *local epipolar tangencies*, even though these can be self-occluded, which correspond to local maxima and minima in the epipolar angle along the edges of the images of the objects (hence *local epipolar angle extrema*). These are illustrated in Figure 7.

For further details, see Cipolla (2006).

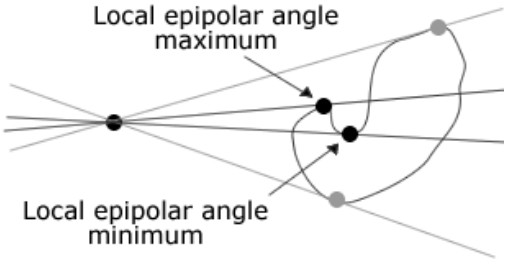


Figure 7: Local epipolar tangencies

3.3 The Kalman filter

The Kalman filter provides the minimum mean square error estimator for a dynamical process which can be represented as by the following stochastic difference equations, with internal state $\mathbf{x}_k$, control input $\mathbf{u}_k$, measurement $\mathbf{z}_k$, and additive process noise $\mathbf{w}_k$ and measurement noise $\mathbf{v}_k$:

$$\mathbf{x}_k = A\mathbf{x}_{k-1} + B\mathbf{u}_k + \mathbf{w}_k \tag{3.4}$$

$$\mathbf{z}_k = H\mathbf{x}_k + \mathbf{v}_k \tag{3.5}$$

Where it is assumed that:

$$\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k, P) \tag{3.6}$$

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, Q) \tag{3.7}$$

$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, R) \tag{3.8}$$

In practice, real systems do not meet the above conditions, but may be well approximated by this model.

For such a system, given the parameters A, B, H, Q and R, the Kalman filter can be derived as a two-step recursive filter (Welch & Bishop, 2001). The first step, the *time update* or *prediction* step, estimates the distribution of the state at time step k based on prior knowledge i.e. knowledge up to time step $k-1$:

$$\hat{\mathbf{x}}_k^- = A\hat{\mathbf{x}}_{k-1} + B\mathbf{u}_k \tag{3.9}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{3.10}$$

The second step, the *measurement update* or *correction* step, incorporates the measurement at time step k to provide an *a posteriori* estimate of the distribution of the state at time step k:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{3.11}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + K_k(\mathbf{z}_k - H\hat{\mathbf{x}}_k^-) \tag{3.12}$$

$$P_k = (I - K_k H) P_k^- \tag{3.13}$$

Where K_k is termed the *Kalman gain*, which with the above form minimises the trace of the *a posteriori* state covariance matrix, i.e. the mean square error of the estimate.

The recursive nature of this filter, and the simple linear algebra calculations required at each time step, allow it to be implemented simply and efficiently.

For a reasonable set of parameters for a system, the Kalman filter can thus be used to estimate the state of the system over time, despite process and measurement noise. This can also be useful for estimating state components that cannot be directly measured.

For further details, see Welch & Bishop (2001).

3.4 Dynamic time warping

Dynamic time warping is a technique for aligning time series which are stretched or warped relative to one another along the time axis, as illustrated in Figure 8. It determines the time warp over the whole time series with minimum cost, for some defined cost function, and calculates this minimum warp cost. In the context of this project, this minimum cost is the useful output as it gives a measure of similarity between the two time series. Dynamic time warping provides a very efficient means of calculating such a measure as it can be calculated using dynamic programming (whence the name).

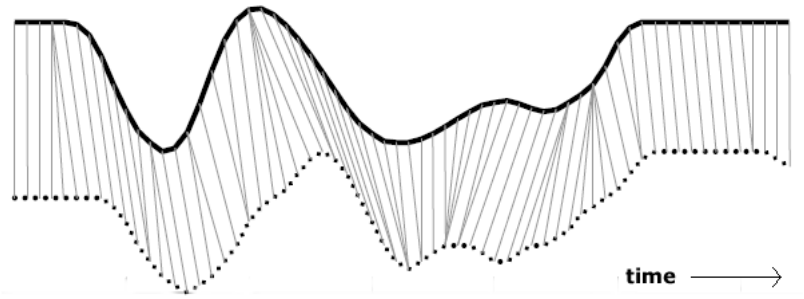


Figure 8: Finding a time warp (from Salvador & Chan, 2004)

The problem it seeks to solve can be clearly formulated as in Salvador & Chan (2004). Given two time series X and Y,

$$X = x_1, x_2, ..., x_N$$

$$Y = y_1, y_2, ..., y_M$$

construct a time warp W,

$$W = w_1, w_2, ..., w_K$$

where each element of the warp w_k represents a mapping between x_n and y_m, which can be shown specifically in the notation as w_{nm}.

The warp is constrained to start at the beginning of both time series and end at the end of both time series and to pass through each intervening element of each time series, giving a full mapping. The warp is also constrained to proceed along each time series monotonically.

The optimal warp is that which minimises a specified cost function, which must be dependent only on the mapping defined by each w_k independently for efficient calculation with dynamic programming. This cost can be considered a distance between the mapped elements of each time series:

$$\text{Dist}(W) = \sum_{k=1}^{K} \text{Dist}(w_k) \tag{3.14}$$

Having defined the above, dynamic programming can be used to find the minimum warp cost. This is simple to define for the application of this project as a means to determine the similarity between a pre-defined time series (a canonical gesture) and a series of data from the vision system (see Section 5.7). Hence between each time step it is only necessary to store the distance associated with the optimal warp that terminates with a mapping to each element of the canonical gesture. When the data at the next time step t is available, these distances (stored in vector D) can then be updated to give the new optimal warp costs by the following update rule for element n of the canonical gesture:

$$D^{(t)}(n) = \text{Dist}(w_{nt}) + min\left(D^{(t-1)}(n), D^{(t-1)}(n-1), D^{(t)}(n-1)\right) \tag{3.15}$$

This update rule allows for the three possible monotonic mappings between the canonical gesture and the data, as also illustrated in Figure 9. The first maps the current data point to the same state of the canonical gesture as the last data point, the second maps to the next state of the canonical gesture, and the last maps the current data point to more than one state of the canonical gesture (i.e. skips states in the canonical gesture).

At each time step, the similarity between the time series, or between the data from the vision system and the canonical gesture in this case, is given by the cost of the mapping being completed at the time step. As this is dependent on how many data points have been used in the mapping, the actual cost can be taken to be averaged over the number of data points:

$$\frac{1}{t}D^{(t)}(N) \tag{3.16}$$

For further details, see Salvador & Chan (2004).

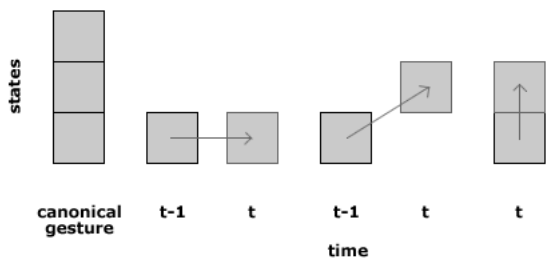


Figure 9: Possible warps

4 Overview

4.1 Key features

From the previous work, described in Section 2, a number of high level problems were identified as having not previously been tackled. These related to the use of vision-based gesture interaction systems for multiple simultaneous users while at the same time taking advantage of the potential with stereo vision systems of using full 3D information. The major contribution of this project was to identify and work towards solving these problems.

Another key feature was that the gestures considered were hand gestures. This was not uncommon in the literature, but in the context of table-top environments was an obvious choice as hand gestures performed over the table-top could be viewed by viewing the table-top itself, allowing interaction from any position around the table, which would not be possible if the cameras angled away from the table-top towards the user.

As originally stated in Section 1, it was also desired for the solution not to require any specialist or expensive equipment, and hence the cameras used were cheap webcams.

Finally, a key simplification used by the system was to focus only on determining *gross hand motion* i.e. the overall movements of the hand. Based on the work of Quek (1994), this low level feature extraction should be more robust than full hand pose determination due to the reduced complexity of what has to be inferred by the vision system, and also faster and able to run on affordable hardware, but not greatly reduced in potential to recognise gestures. Such a system was also expected to increase usability by not placing specific demands on users' hand configuration. The system should also generalise to allowing gesture using a pointing device, to give more accuracy as with the use of a stylus in many touch-screen devices.

These features led to the following project problem statement.

> To develop a real-time system to visually detect and track the gross motion of hands in 3D over a table-top, and recognise specific motion gestures, using only cheaply available cameras. The system should be easy to use, support multiple simultaneous users and be robust.

Note that it is not suggested that such a system is the answer to all the problems of human-computer interaction: it is not. There are many drawbacks of the system proposed, for example that requirement of line of sight for the vision system to function, and the usability issues of requiring specified motions to be carried out by the user. However, the hope is that it would be a more effective interface for certain applications.

4.2 Use cases

The potential for the outlined system can be demonstrated by considering use cases in which it can outperform existing methods. Firstly, a design environment can be imagined in which a team of designers may work together on a single augmented surface. The 3D hand tracking could be used to assist 3D design, with gestures used to choose different functions of the design system, such as to add, remove and locate parts and to choose which part of the design to work on. Many people in the team could be working on designing the same element or parts of the same element, and would all be able to the work of the others as well as if they were hand-drawing the design. With no special devices required for interaction, people could leave and join the process easily, without the potential for essential interactive devices to become lost or to break.

A second use case can be taken from the example of Reitmayr & Drummond (2005): one can imagine augmenting the map table interactions for co-ordinating flood control in the city of Cambridge by use of hand gestures. For example, thumbnails of photos taken at sites around Cambridge could be projected onto the map at the location they were taken. A selection gesture (such an encircling) could select a photo to be viewed in greater detail elsewhere on the the table (at a location selected by pointing). While viewing, rotation and vertical motion gestures may be used for rotating and resizing the image to allow a clearer view, or to show the image to another user. This may have some benefits over the system described by Reitmayr & Drummond by reducing the need for

tangible interfaces which may be mislaid or in short supply, allowing the system to aid the decision making process and responsiveness of a flood control team and allow them to carry out their operations more effectively.

4.3 Implementation

The physical setup for the system was flexible but was essentially as shown in Figure 10. Two cheap web cameras were used, located above the table-top to allow the vision system to clearly see the the table-top.

The cameras were connected to a computer with a dual-core processor, each processor with clock speed 2.40GHz, running openSUSE Linux. The cameras could easily be accessed through the C++ library LibCVD (Drummond et al.), developed within the lab. The software developed for the system was written in C++ so as to make use of this library and also due to its advantages in speed and power over other programming languages. This was necessary as it was desired for the system to run in real time, resulting in high requirements of processing and memory usage, especially for parts of the system requiring operations to be carried out on every pixel of each image from the two cameras.

The processes required by the software were as shown in Figure 11:

Background subtraction: Determine pixels of interest in the images i.e. those that differ from the (assumed fixed) background (see Section 5.2).

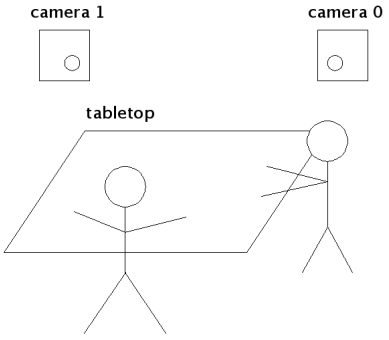


Figure 10: Table-top system setup

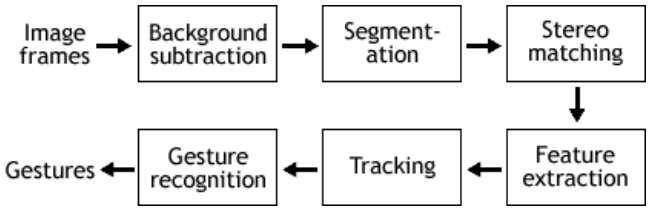


Figure 11: Process diagram of system

Segmentation: Separate the groups of pixels into lists of pixels for each hand in each image (see Section 5.3).

Stereo matching: Determine which hand images from each camera correspond to the same real hand (see Section 5.4).

Feature extraction: Extract the required motion features for gesture detection from the tracked hands (see Section 5.5).

Tracking: Track real hands moving in the images over time (see Section 5.6).

Gesture recognition: Recognise gestures from the time series of motion features (see Section 5.7).

These detected gestures could then be used by applications to carry out actions (see Section 5.9).

In the design of these processes, the main constraint was that the processes were fast enough to be able to run on the system in real time. Further desired properties were accuracy, robustness to illumination, hand movement location and speed, usability and the ability for multiple users to interact simultaneously. Note that robustness to a generally varying background was not aimed for, allowing background subtraction to be used to identify pixels of interest. This was to reduce the scope of the project to more manageable limits, although it is clearly a desirable property which could be worked towards if time allowed. In practice, simple and fast approaches were tested first, with more complex procedures used if the simple approach did not provide adequate performance.

5 Implementation

5.1 Camera calibration

In order to use a stereo vision system to locate real-world objects in 3D, it is necessary for the cameras themselves to both be calibrated to a 3D reference frame, and for the mappings from 3D locations to camera pixels to be defined. This is termed calibration of the cameras, and determines the parameters of the projective relationships defined in Sections 3.1.2 and 3.1.3. Note that this need only be carried out once before the gesture interaction system could be used.

The calibration was performed using existing software that was developed in a previous fourth year project (Taylor & Drummond, 2007) and software included with LibCVD (Drummond et al.). These systems output these parameters to a file which could be read in by the software.

Note that, due to the non-linear camera model used (see Section 3.1.3), the full projection from 3D location to camera pixel was non-linear. Hence much of the geometric processing was carried out on the linear image plane found by inverting the non-linear relationship of Equations 3.2 and 3.3.

5.2 Background subtraction

The first step of the vision system was to detect pixels in the camera images belonging to hands. This was implemented by simple background subtraction, under the assumption that the background changed less frequently than pixels of interest. Background subtraction builds a model of the background pixels by some form of adaptive filtering. Pixels from each camera image were classified as foreground if they differed from the background model by above a certain threshold in colour, otherwise they were classified as background. The fast and simple approximated median filtering algorithm (McFarlane & Schofield, 1995) was used to adapt the background model, as described in Algorithm 1.

Note that the algorithm only updated the background model for pixels which the existing model classified as background. This required a good initial estimate of the background

Algorithm 1 Approximate median filtering background subtraction

```
for all pixel positions p do
  if p is a background pixel then
    if Input(p).Red > BackgroundModel(p).Red then
      Increment BackgroundModel(p).Red
    else if Input(p).Red < BackgroundModel(p).Red then
      Decrement BackgroundModel(p).Red
    end if
    {Same for Blue and Green colour components}
  end if
end for
```

image (i.e. a 'hand-free' image of the scene). However the background model adaptation was still useful to allow some robustness to changes in lighting conditions and camera parameters such as contrast and gain, which by default varied automatically.

Another initial problem, due to the use of only background subtraction to detect hand pixels, was that this method was incapable of distinguishing the actual hand from the shadow of the hand on the table-top surface. To remove this problem, the table was covered in black card. This was an undesirable constraint as it places unreasonable constraints on the table-top which violate the motivation for the system, however it was necessary in this case due to limited time; with more time available, more sophisticated techniques could be employed which may allow this constraint to be removed.

5.3 Segmentation

5.3.1 Flood-fill edge segmentation

The hand pixels detected by background subtraction were then grouped into separate 'hands' by segmentation i.e. the grouping together of connected foreground pixels. This was first carried out in order to extract the segment edges by the flood-fill algorithm as outlined in Algorithm 2.

At the end of the algorithm's run, the list *Segs* contained the segments, each itself a list of pixels. Segments were removed if they only contained a small number of pixels, as many segments were simply groupings of noise. It was found that a lower limit of 200 pixels was suitable for this.

Algorithm 2 Flood-fill algorithm for edge segmentation

```
create list Segs
for all pixels p in image I do
  if Foreground(p) is true and Scanned(p) is false then
    create new list S
    push p to list T
    while T not empty do
      remove item from T and put in t
      if Scanned(t) is false then
        Scanned(t) is true
        if Foreground(t) is true and Foreground(t') is false for some t' in
        Neighbours(t) then
          push t to list S
          push all t' in Neighbours(t) to list T
        end if
      end if
    end while
    push list S to list Segs
  end if
end for
```

A key problem with this method was that if hands touched or occluded each other from the viewpoint of the camera, they were placed in the same segment (even if a human could easily detect that there must be two hands involved due to the number of fingers, etc). However, this was not problematic provided the hands were not always in this state, as later processes tracking the hands moving over time were able to determine when hands have moved into such a position and still track the position of the hands separately.

5.3.2 Branch-free edge segmentation

The edge segmentation described in Section 5.4.2 also had the problem that the extracted edge contained branches and loops due to noise and single pixel wide sections in the output of the background subtraction. This was problematic in that it led the extracted edge to not be the true, most outer edge of the segment and because the list of pixels in the edge was not in a meaningful order. To allow later processing to use this information, a better method to determine the correct edge segmentation needed to be implemented.

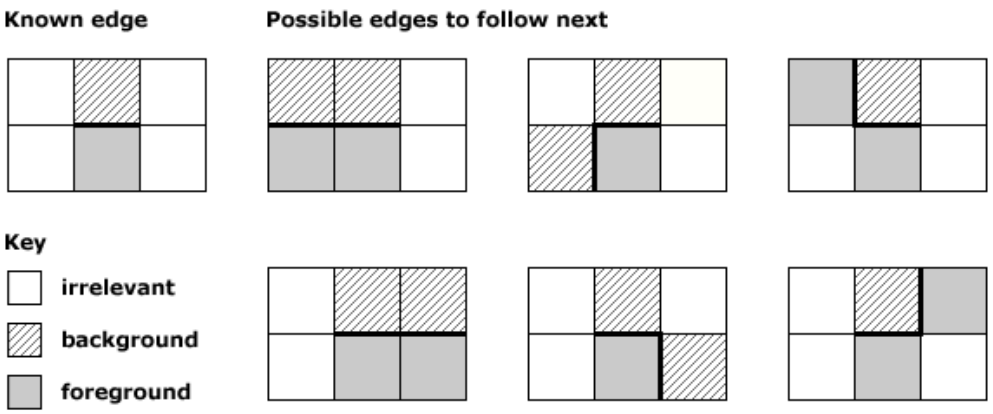


Figure 12: Possible edges to follow for branch-free edge detection

The key idea in this implementation was that the very first edge pixel found must lie on the true edge, as otherwise one of the pixels checked earlier in the scan through the image would have initiated the segmentation of that segment. The true edge could then be determined by following the pairs of foreground and background pixels where the two types of pixel border the respective type of pixel in the previous pair. Hence all the background and foreground pixels segmented were connected, a more strict condition than that used in Section 5.3.1. This was equivalent to considering the actual edge to be the border between the foreground and background pixels rather than a list of either.

From a starting point, there are a number of possible paths which this edge could take, as illustrated in Figure 12. Which are valid depended on whether the neighbouring pixels were classified as foreground or background as shown. By following at each step a valid next edge and preventing an edge being followed more than once, the branch-free edge segmentation could be made.

This method was implemented and correctly segmented the true edge, giving the list of edge pixels in order of bordering pixels. Despite requiring more conditions to be checked throughout the segmentation, it was found not to be noticeably slower.

5.4 Stereo matching

Once the pixels were segmented into hypothetical images of hands, it was necessary to determine which hand images corresponded to the same real hand. Note that, with a

very high frame rate, this could be possible simply by time-domain tracking of the hands from the moment they entered the images. However, with the relatively low frame rate available and the potential for people to move their hands very fast during an interaction, such methods could not be relied on. Instead, other methods were used in which the epipolar constraint (as described in Section 3.2) was key.

First, the global epipolar angle maximum and minimum were determined for each hand image. The epipoles for each camera image were determined during the camera calibration in the 2D image plane co-ordinate system. One of these, say from camera 1, was used to determine the epipolar angle for each pixel of each camera, a property of each pixel which was constant provided the physical setup of the cameras did not change, and hence one that could be calculated as part of the calibration stage.

In order to calculate this for camera 1, firstly a pixel in the image of camera 1 was chosen and its image plane position determined. This required inverse solving Equations 3.2 and 3.3 (from Section 3.1.3), which was done using the standard LibCVD function which uses three iterations of Gauss-Newton (Drummond et al.). The epipolar angle, with epipole position $\mathbf{e} = (e_x, e_y)^T$, was then given by:

$$\theta = \arctan\left(\frac{X_c - e_x}{Y_c - e_y}\right) \tag{5.1}$$

Note that this set the epipolar angle datum ($\theta = 0$) in the direction of the X_c axis for camera 1, passing through of the camera 1 epipole (i.e. the projection of the origin for camera 2 onto the image plane of camera 1). Hence to calculate epipolar angles in the same parameterisation for camera 2, it was also necessary to project pixels in the image of camera 2 to the 2D image plane of camera 1. This was possible by inverse solving the intrinsic calibration equations as above, setting $Z_c = 1$ (for camera-centric coordinates for camera 2) so as to complete the specification of the 3D position on the image plane. The transformation into 2D image plane co-ordinates for camera 1 was then completed by transforming by extrinsic calibration Equation 3.1. Finally the above Equation 5.1 was again used to find the epipolar angle.

5.4.1 Use of global epipolar angle extrema

Using the epipolar angle calibration, it was simple to determine the global epipolar angle maximum and minimum for each hand image by searching through the pixels of each hand image. The hand images could then be characterised in a 2D space by these values. Matches were then made at first by greedily matching hand images which were closest in terms of Euclidean distance in this space.

This method of matching was found not to be very accurate. A key reason for this was that an implicit assumption of using the maximum and minimum epipolar angles for each hand image was that each hand was a separate object, entirely located within the image of each camera. In fact both of these assumptions were false, with at least one edge of each hand image would be expected to be the edge of the camera image. Often one of the epipolar angle extrema would occur along this edge, so that the extremum did not provide information about the hand and would not match to the extremum for the same hand in the other camera image. Hence the 2D space identified above could be viewed as actually being of a much lower dimensionality and thus providing less discriminatory information for the hand matching than might be expected.

5.4.2 Use of local epipolar angle extrema

In order to match more accurately, it was necessary to extract more information from the segmented hand images. As well as the epipolar constraint, another common method of determining information for matching is to use appearance, for example making use of colour and texture to determine matching image patches from multiple cameras. However, in this case, only hands were to be tracked, with the poor quality images provided by cheap web cams. Hence it was thought unlikely that methods of matching based on appearance would be able to provide any reliable information. Instead, further geometric information was required.

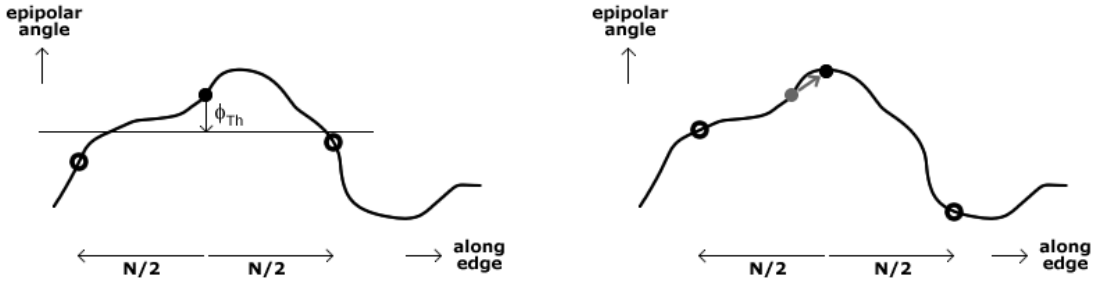
One potential source of this information was *local* epipolar angle extrema, as described in Section 3.2. These can self-occlude and thus they may not always be visible from both cameras (if occluded) or the epipolar angles at which they occur may be different for each camera (if partially occluded), as illustrated by three possible viewpoints of a hand

(a) No occlusion (b) Partial occlusion (c) Full occlusion

Figure 13: Self-occlusion of a local epipolar extremum (marked by dot)

in Figure 13. However, as many may be present on each hand (obvious examples include fingertips and the gaps between fingers, with the hand oriented appropriately), it was thought likely that enough of these extrema may match to provide enough evidence to match hands. Also, these local extrema would be more dependent on the position and pose of the hand than the global extrema, and so could provide greater discrimination.

Local epipolar angle extrema were extracted simultaneously with the edge segmentation by a simple approximation. Given an epipolar angle threshold θ_{Th} and a minimum extremum scale in terms of distance along the edge N pixels, a maximum was detected if a pixel was greater in epipolar angle, by more than θ_{Th}, than the pixel $\frac{N}{2}$ before it and the pixel $\frac{N}{2}$ after it. This is illustrated in Figure 14(a). A minimum was similarly detected if the epipolar angle was smaller. If the extremum was stronger than another within the length scale N, it was deemed a more accurate positioning of the same extremum and replaced it. This is illustrated in Figure 14(b).



(a) Maximum detected (b) Better maximum detected

Figure 14: Detecting local epipolar angle extrema

Several methods were then tried for obtaining hand matches from these local epipolar angle extrema. These included cost functions between each hand segment e.g. based on the average sum of squared distances between each of the maxima and each of the minima, and based on the number of extrema which were close (i.e. within a specified threshold). It was also attempted to match each extremum individually, again by Euclidean distance, with each match acting as a vote for the possible hand segment matches.

None of these method proved very effective at finding suitable matches. One obvious reason for this, looking at the extrema detected, was the number of 'false' extrema detected in the noisy segment edges where segmentation failed, often if the arm was in shadow, making it difficult to determine the correct edge against the black background. In order to try and improve this, a number of means of filtering the extrema were tested. These included increasing the threshold on epipolar angle difference such that only very strong extrema were detected. The edge pixels around the extrema were also tested for finger-like characteristics e.g. the pixels on each side of the extrema lying roughly on parallel lines with opposing senses, measured with the covariance matrices of the pixels on each side. However, it proved difficult to achieve any improvement in performance.

This testing had made it clear that there were a number of fundamental issues with the use of these features that it was possible no amount of processing would overcome. In general, due to issues with the segmentation in areas such as the shadowed area of the hand, the extrema detected would always be noisy. They would also always be subject to self-occlusion. For features that were inherently low dimensional to begin with, with maxima and minima each independent and one-dimensional, these problems would always lead to difficulties in extracting useful information from the extrema.

An artificial visual example of the task can be seen in Figure 15. While some similarities can be seen between the two images of the hand in the structure of the extrema, many of the extrema are only present in one image or are shifted due to partial occlusion. With hands located in a similar position in the visual scene, it would not be expected that the extrema would be located so differently, and hence distinguishing between them would not be expected to be reliable.

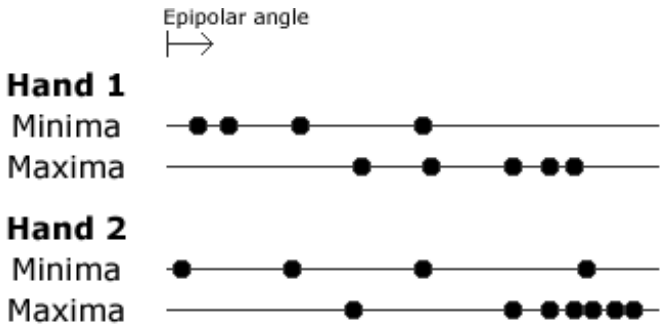


Figure 15: The difficulty of matching with local epipolar angle extrema

5.4.3 Requiring a user constraint

A large amount of time had been spent working on the use of local epipolar angle extrema for hand matching, without much success. In order to ensure that at least a simplified version of the system could be developed in the time remaining, it was necessary to impose some further constraints on the system to allow the matching stage to be made accurate enough. A simple way to do this was to ensure that matching with the global epipolar extrema would work well.

This was achieved by requiring that the user have their hand not pointed away from the cameras and pointed down (towards the table), ensuring that a global epipolar angle minimum could be used for matching. These were quite strong constraints, but would still allow plenty of flexibility in the interaction to allow the system to be tested.

In order to deal with situations when the hands were located in close to the same epipolar plane, the local epipolar angle extrema were still extracted. Only the minima were used, as with the constraints added above these were less likely to be occluded, and these were filtered to remove the weakest minima in each segment by location in the image. The global epipolar angle minima were found by further filtering of this set to find the minimum with the lowest associated epipolar angle within a large region of the image. Note that this allowed for multiple global minima in each segment, to allow some tolerance to issues with the segmentation.

Algorithm 3 Unique matching

```
while boolean c is true do
  c ← false
  for all hand images i from camera 0 do
    if i not already matched then
      for all hand images j from camera 1 do
        if j not already matched then
          if |MinEA(i)−MinEA(j)| < EA_Thres then
            j is a potential match for i
          end if
        end if
      end for
      if i only has one potential match j_0 then
        Match i and j_0
        c ← true
      end if
    end if
  end for
  for all hand images j from camera 1 do
    {As above with potential matches i from camera 0 searched for}
  end for
end while
```

The matching algorithm then made matches between hands with global minima uniquely located in close to the same epipolar plane (specified by a threshold). The algorithm for achieving this is described in Algorithm 3. Matches from each hand image from each camera were checked as a match may be unique even if one of the hand images has several potential matches within the epipolar angle threshold, provided the other hand image has only the one potential match. Note that the continuation boolean c remained false at the end of an iteration once all unique matches have been made, ending the loop.

If all the hands could not be matched uniquely, a cost function was calculated between each hand based on the average error in good epipolar angle matches (i.e. within a threshold) between local epipolar angle extrema. The match with the lowest cost was made, and then algorithm then attempted again to make unique matches and reverted back to using the cost function when this was not possible.

This proved much more reliable than the previous matching algorithms, providing usable performance.

5.4.4 Fingertip matching

An alternative mode of the system was also developed in which it was required for the user to point with a finger. The system was otherwise as described in Section 5.4.3. This mode had the advantages of offering greater precision to the user in terms of control of the position and orientation measured by the system. However, it was possible that this would come at the cost of a decrease in usability as described in previous work in Section 2.2, due to the physical difficulty of pointing for a length of time. With the inherent usability issues of gesture-based interaction, the further requirement of a pointing gesture may not seem such a strong additional constraint, however.

5.5 Feature extraction

The means of interaction of the user with the system was the gross hand motion, based on visual measurement of the position and orientation of the hand. Once hands had been segmented onto of the camera images and matched, it was possible to do extract this information.

In order to determine a 3D position for the hand, it was first necessary to define a 'position' of the hand in each camera image. In the hand tracking mode and fingertip tracking mode respectively, the pixels making up the hand and finger were extracted by proximity to the global epipolar angle minimum. The centroid of these pixels was then chosen for the position measure, given by:

$$\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} = \frac{1}{N} \sum_{n=1}^{N} \begin{pmatrix} x_n \\ y_n \end{pmatrix} \tag{5.2}$$

As in Section 5.4, this position was then mapped into 3D co-ordinates by inverse solving Equations 3.2 and 3.3, setting the z co-ordinate to unity and solving Equation 3.1. With no error in any of the measurements, the 3D position of the hand would then be constrained to lie on the ray between this 3D point and the optical centre of the camera. With such rays from two cameras, the 3D position would lie at their intersection. In practice, it was necessary to solve for the least squares position, given by the half-way point on the line joining the two rays at their closest point, as illustrated in Figure 16.

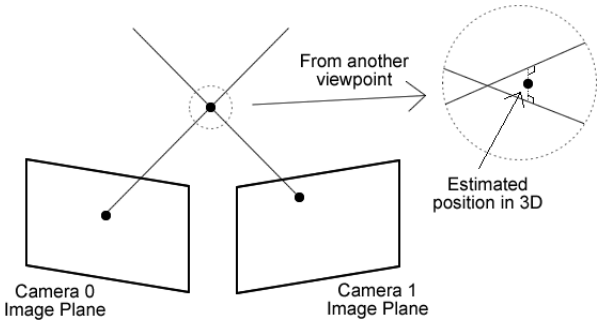


Figure 16: Rays from each camera constrain a 3D position

The rays were parameterised in terms of the x co-ordinate, giving 3D line equations:

$$y = m_1 x + c_1 \tag{5.3}$$

$$z = m_2 x + c_2 \tag{5.4}$$

For both cameras, these equations can be written in a matrix form:

$$\begin{pmatrix} -m_{11} & 1 & 0 \\ -m_{21} & 0 & 1 \\ -m_{12} & 1 & 0 \\ -m_{22} & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} c_{11} \\ c_{21} \\ c_{12} \\ c_{22} \end{pmatrix} \tag{5.5}$$

The least squares solution for the position was then calculated using the singular value decomposition functions of TooN (Drummond et al.).

It was then desired to determine a 3D orientation for the hand. In order to do this, it was first necessary to determine the orientation of the hand in each camera image. The measure chosen for this was the axis through the centroid about which the second moment of area of the hand pixels (given as before) was minimised. This can be shown to be given by the eigenvector of the covariance matrix of hand pixels (about the centroid as found by Equation 5.2) corresponding to the larger of the two eigenvalues (Pearson, 1901).

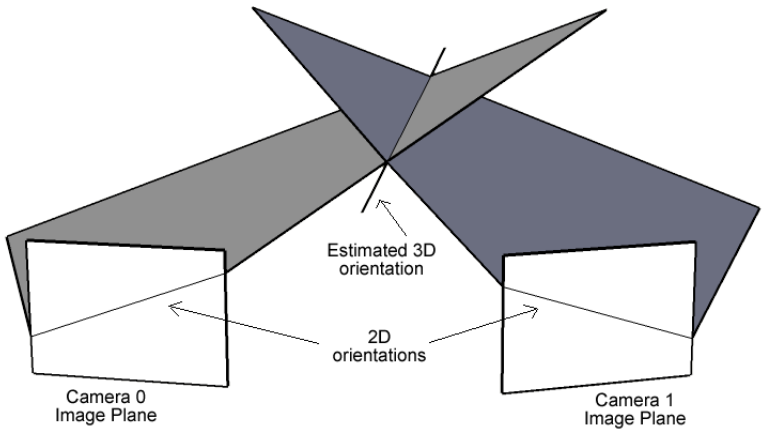


Figure 17: Planes from each camera constrain a 3D orientation

This covariance matrix is given by (ignoring normalisation):

$$\sum_{n=1}^{N} \begin{pmatrix} (x_n - \mu_x)^2 & (x_n - \mu_x)(y_n - \mu_y) \\ (x_n - \mu_x)(y_n - \mu_y) & (y_n - \mu_y)^2 \end{pmatrix}$$

The eigenvector was found using the eigenvector functions of TooN (Drummond et al.).

The same geometry as used in finding the 3D hand position was then used to find the 3D ray between the optical centre and a point in the image along the direction of this orientation from the centroid. This ray and the ray passing through the centroid defined a 3D plane. The 3D orientation was given by the intersection of the two planes from the two cameras, as illustrated in Figure 17.

In order to calculate this orientation, each plane was parameterised in terms of its normal vector $\mathbf{n}$:

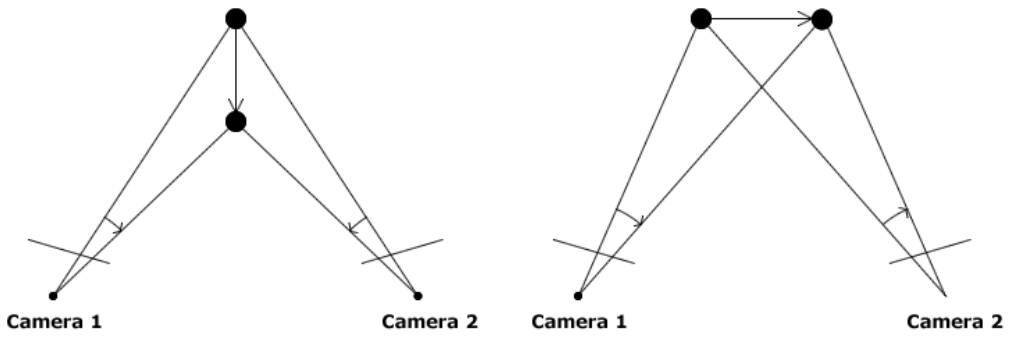
$$\mathbf{n}^T \mathbf{r} = d \tag{5.6}$$

This normal vector could be determined as the vector cross product between vectors parallel to the two rays defining the plane. The orientation could then be found by the vector cross product of these two plane normals. This was normalised and multiplied by ± 1 to constrain the orientation by assumption to always be pointing downwards (towards the table-top), as given by the constraint introduced for matching in Section 5.4.3.

5.6 Tracking

The correspondence problem was concerned with matching images of the same real hand from two cameras. Tracking was then the problem of matching the matched images of the same real hand over time, a necessary step in order to determine the motion of the hands. This problem was solved in three stages; firstly, a predictive motion model was developed, which predicted the 3D location of the real hands in the scene based on past data. Secondly, these models were matched to the data from the preceding parts of the vision system to complete the matching. The third stage was necessary for dealing with the 'edge effects' of hands leaving the visual space, or new hands entering it.

The predictive model used was the Kalman filter as described in Section 3.3. A predictive motion model was needed in order to deal with occlusions with multiple hands in the frame: if one hand occluded another, the predictive model would be able to predict this and allow tracking to continue when the hands were visible again. A constant acceleration model was used (i.e. only position and velocity were included in the state vector) and the parameters of the filter model were chosen based on the recommendations for initialisation of Kohler (1997) and by trial and error. Appropriate setting of the parameters allowed the model to predict the hand motions well, but this was dependent on the setup of the system. For example, while the calculations described in Section 5.5 obtained an estimated 3D position, the error in this estimate was greater in the directions towards the cameras as a small change in angle could lead to a large change in estimated 3D position along these directions. This is illustrated in Figure 18.



(a) Small angle subtended (b) Larger angle subtended

Figure 18: Angles subtended by a moving object depend on the direction of motion

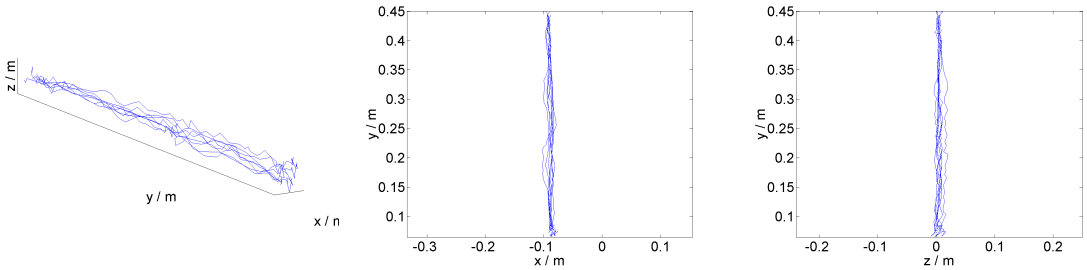
The second part of the tracking algorithm matched detected hands in the latest image frames ('current hands') to the predicted positions for previously detected hands ('past hands'). Hypothetical matches were proposed if the Euclidean distance between a past hand and current hand was less than a threshold. If a number of past hand positions were within this threshold for the same current hand, the past hand with the highest position (i.e. on the z axis) was chosen for the match, by assumption that this hand was occluding the others. If no past hand match could be made for a current hand, it was assumed that the current hand was new to the visual scene.

The third part of the tracking dealt with a number of 'edge effects'. In each case, a most common explanation was inferred and action taken accordingly.

The first effect was the case of past hands for which there was no matched current hand. It was assumed that the hand had temporarily left the visual scene. A counter was incremented to count the number of consecutive time frames the past hand had not been matched to a current hand, and if this counter reached a certain threshold, tracking of the past hand ceased.

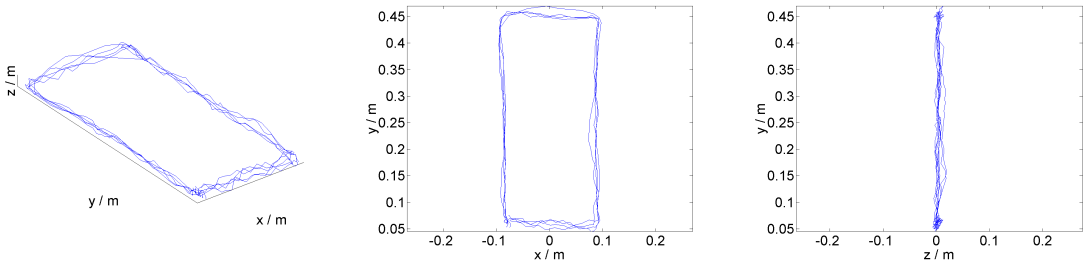
The second effect was the case of past hands which were matched to the same current hand as other past hands. It was assumed that this was due to occlusion. For each past hand, counters were incremented if the two past hands were matched to the same current hand. If any of these counters reached a certain threshold, the past hands were deemed to have converged to tracking the same real hand and the tracking of extraneous past hands was stopped, leaving only one to track the one hand.

This tracking system was tested and was found, along with the feature extraction, to perform well. This can be seen from Figures 19 and 20 which show plots of the positions of a tracked fingertip as it followed a straight line and then rectangular guide on the surface of the table a number of times. Notice that the results are very repeatable and that there is little variation in the height reading along each guide, as would be expected with both lying on the surface of the table.



(a) 3D plot (b) y against x (c) y against z

Figure 19: Tracked position when following a straight line guide on the table surface



(a) 3D plot (b) y against x (c) y against z

Figure 20: Tracked position when following a rectangular guide on the table surface

However, while the accuracy was good, the system was unable to track hands that moved very quickly, as the frame rate was not high enough to gain sufficient data. This was due to limitations in the cameras and the time taken to carry out the required processing. Hence, use of the system required sudden movements to be avoided, which felt somewhat unnatural. Provided movements were relatively slow however, the tracking worked well.

There were also a number of unavoidable failure modes of the implemented tracker; for example, if two hands were brought together and then moved apart, the motion model would assume that instead the two hands had crossed over and so the hand trackers would swap the real hands they were tracking. However, such motions would not necessarily be normal and the ability to correctly interpret the action would not be worth the greatly more complicated processing required.

5.7 Gesture recognition

The previous sections of this report have detailed the system processes required to estimate tracked motion of hands based on images from two cameras over time. This was the input required for the motion-based gesture recognition system. In the previous work described in Section 2.2, probabilistic frameworks such as Hidden Markov Models were often used for gesture recognition to account for the uncertainty in the measurements and in the performance of gestures. However, in the interests of speed, a more deterministic method was tested first, based on dynamic time warping.

Dynamic time warping was introduced in Section 3.4. In general it offers an efficient way to calculate an optimal warp between two time series in terms of a given cost function. This was applied to gesture recognition by considering gestures as a time series of motion states. In these terms, canonical gestures could be pre-defined and compared to the tracked hand motions. Dynamic time warping could then be used to find the minimum cost of the tracked hand motions matching to any of the canonical gestures, and hence allow detection of the gestures by setting a threshold on this cost.

Firstly, a motion state was defined. It was desired that the state should be invariant to the position and orientation of the user. This was achieved by using as the state a normalised 3D velocity vector in a co-ordinate system relative to the orientation of the hand. For orientation unit vector $\mathbf{o}$, the x direction was chosen to be the same as $\mathbf{o}$, the y direction was chosen to be perpendicular to x, horizontal and pointing to the right

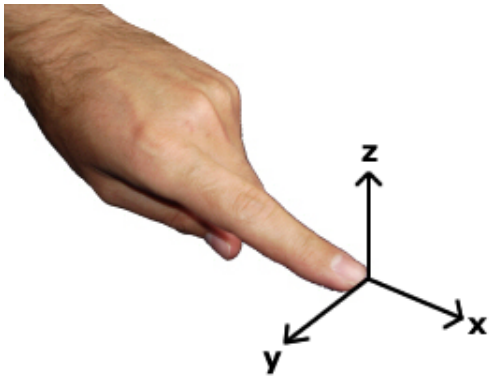


Figure 21: User-centric co-ordinate system

(found by the cross product of $\mathbf{o}$ with a vector in the z direction) and the z direction was left as the vertical direction. The origin of the co-ordinate system was the centroid or fingertip of the hand, depending on the mode being used. This is illustrated in Figure 21.

A vector $\mathbf{x}$ could be transformed to this co-ordinate system, given the 3D origin point $\mathbf{p}$ in the original co-ordinate system, by the transformation:

$$\mathbf{x}' = \begin{pmatrix} \leftarrow \mathbf{e}_x^T \rightarrow \\ \leftarrow \mathbf{e}_y^T \rightarrow \\ \leftarrow \mathbf{e}_z^T \rightarrow \end{pmatrix} (\mathbf{x} - \mathbf{p}) \tag{5.7}$$

where

$$\mathbf{e}_x = \mathbf{o},\ \mathbf{e}_y = \mathbf{o} \times [0\ 0\ 1]^T,\ \mathbf{e}_z = [0\ 0\ 1]^T$$

The same transformation, without the translation correction for the origin, was used to find velocities in this co-ordinate system. Note that velocities with a magnitude lower than a threshold were mapped to the zero vector to reduce noise when a hand was stationary or near-stationary.

Secondly, it was necessary to define a cost function between these states. The Euclidean distance was chosen for this. It was also made possible to specify a fixed cost for skipping a state in the canonical gesture i.e. matching a state from a tracked hand to more than one state of the canonical gesture. For user-defined gestures which may have few states, it would be expected that many states from the tracked hand would match to each state in the canonical gesture. Hence the cost of skipping one state would actually be much larger than just the Euclidean cost function of one poor match. The state skipping cost could account for this.

Finally it was necessary to define the threshold for gesture detection. This was chosen to be a constant allowed cost per time frame, with at least as many time frames as states in the canonical gesture required. Recognition hypotheses were ruled out if, of all of the possible warps of the tracked hand states ending at each of the canonical gesture states, the minimum cost warp had a cost greater than the allowed cost.

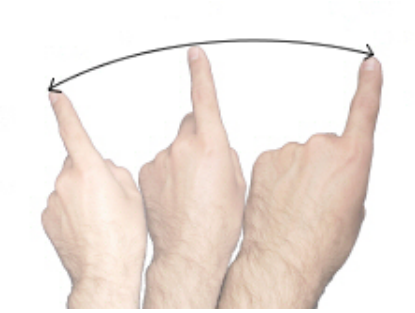


Figure 22: A natural left-right hand motion

Hence a state skipping cost and an allowed cost per tracked hand state were required parameters of a canonical gesture. These costs, along with a series of states, were defined for a few simple gestures such as a repeated up-down or left-right motion, and detection of these gestures was tested. With appropriate parameters, the system worked well, detecting most gestures performed and correctly distinguishing the different gestures.

However, a key issue with the system as so far described was clear when performing a 'left-right' gesture: even in the user-centric co-ordinate system defined, moving the hand in a direction perpendicular to the orientation was unnatural. A more natural left-right hand motion would be a clockwise-anticlockwise rotation of the arm about the shoulder, involving movement forward and backward and possibly up and down, as illustrated by Figure 22. While it was possible to carefully define canonical gestures of more accurate representations such as this, it would be easier to be able to determine the motions by recording the motions carried out in a training example. This was implemented in the system. To ensure the lists of states required were not too long and to allow generalisability for the gesture if performed more quickly than the training example, the states recorded were discretized to 26 directions and duplicate consecutive states removed.

This improved detection of more realistic gestures, making the system easier to use, and also made user control of the system much more flexible, with the option to record or write canonical gestures, with all gestures also fully editable at a later time.

5.8 Code optimisation

Throughout the coding, care was taken in the programming of the algorithms in order to ensure that the system could run as fast as possible, and at least in real time.

Such code optimisation was built into the processes by ensuring that they were written in such a way as to minimise the required number of loops and branch statements, and also to minimise the amount of data duplication. This allowed the full system tracking one hand to be able to run at a frame rate of 14.3fps. That is, it was able to fully process on average 14.3 frames of data (i.e. pairs of image frames) per second.

Further speed was gained by making full use of both processor cores with parallel processing. The system was well suited to this as a number of the processing steps (background subtraction, segmentation and finding of the epipolar angle extrema) could be carried out for the image frame from each camera independently. This increased the frame rate to 20.0fps on average.

5.9 Applications

A number of applications were implemented to allow the tracking and gesture recognition system developed to be applied to carrying out real tasks. The first two applications required some precision and so used the fingertip tracking mode. For both these applications, the interaction space above the table-top was split in two by a horizontal plane. Below this plane, gesture detection was not carried out, while above the plane it was. This was necessary to overcome a key difficulty for some applications of using motion gestures: it is difficult to maintain a hand location while performing the gesture.

The first application allowed control of the mouse pointer on the computer screen, with an 'up-down' motion gesture used to perform a click. This was implemented using XWindow test methods (see X.org) which allow control of the mouse pointer and click events. This application worked well but proved hard to use; in order to locate the mouse pointer to interact with an object on the desktop, it was necessary to locate the corresponding position over the table-top which moved the mouse pointer to that loca-

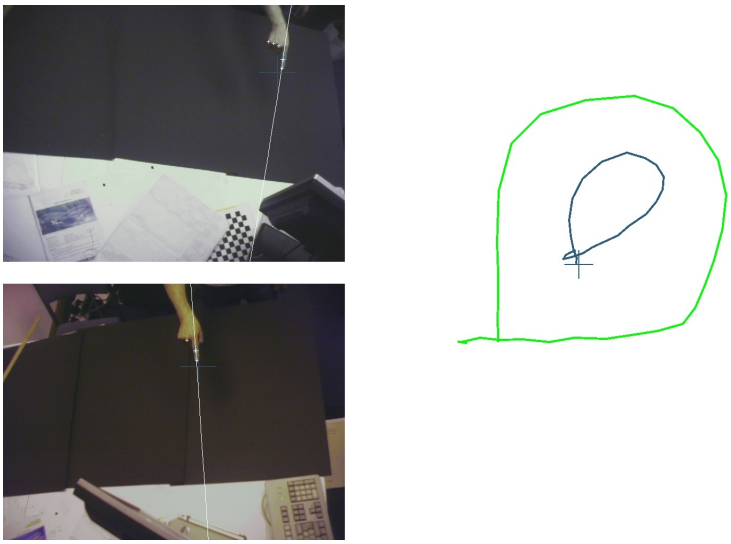


Figure 23: Painting application

tion (not necessarily intuitive in itself), and then move into the gesture recognition space in order to perform a click. However, it was hard to move up into this space without disturbing the position of the mouse pointer. Frequently this would move away from the desired object and be the source of much frustration.

The second application allowed the user to paint with their finger while in the tracking region, with an 'up-down' gesture set to randomly change the colour of the paint, and a 'left-right' gesture used to clear all previous paint from the screen. The working application can be seen in Figure 23, showing the images from the two cameras on the left with the automatic annotation of the detected finger and orientation, and the painting on the right. OpenGL (see OpenGL.org) commands were used to render the painting.

This application again worked well and provided a flexible and spatially meaningful interface for producing computer drawings. For example, it could easily be used to draw the outlines of real objects by moving the finger around the outline of the object. A noticeable drawback of the system with this application was that small errors in the matching could lead to erratic lines being painted, and hence the application did not work well with multiple simultaneous users. The appeal of the application also encouraged users to use it for a reasonable length of time; before long it was found to be physically tiring to use.

The third application simply required gestures to be performed and so made use of the hand tracking mode. Hand gestures were used to control a presentation, with 'right' and 'left' motion gestures allowing movement between slides and 'down' and 'up' gestures allowing the presentation to be stopped and started. The main difficulty with using this application was ensuring that only the required gesture was carried out while interacting, as it was easy to accidentally perform one of the other simple motion gestures while moving into and out of the interaction space.

Using these applications allowed the two modes to be compared. As expected, the fingertip tracking mode allowed for greater precision by allowing the user to more be aware of the position that was being tracked. However, it proved much harder to use physically. This presented a trade-off, with the appropriate mode depending on the intended application.

The applications also allowed the system in general to be evaluated. They showed that the system worked and could be used in a number of applications, but suffered from problems of specifying a precise location context for a gesture, defining and recognising suitably distinguishable but intuitive gestures, physical usability issues and occasional mis-matches. While the first three of these are inherent to this type of interface and not unexpected, the last could be improved upon by better matching. Overall, the system thus demonstrated the concept of a simple gesture-based interaction system, but showed that further work is needed to make such a system practical.

6 Conclusions and evaluation

In this project, a gesture-based interaction system for table-tops was successfully developed, requiring only a pair of cheap webcams and the software developed in the project. The resulting system visually detected and tracked the motions of hands over a table-top in real time, and recognised when pre-defined gestures were performed by the user, with applications built to demonstrate and test the system. The approach taken allowed for simultaneous interaction with multiple users and was reasonably robust to fast hand motion and differences in illumination.

Several constraints were placed on the system to achieve this functionality. The table-top was covered in black card, simplifying the problem of detecting and segmenting hands against a noisy background and even reducing the difficulty of dealing with shadows. The user was also required to point their arm roughly towards the camera in order for their hand to be detected, and to avoid sudden movement. In a fingertip tracking mode, the user was further required to point with a finger. These constraints reduced the applicability of the work in everyday environments and reduced the usability of the system.

Even with these constraints, there were a number of problems with the system implementation. It was not robust to changes in background and illumination while running, it could slow down heavily with noise in the images and it frequently made errors in matching hands between the two camera image frames. Further work is required to solve these problems (see Section 7).

Overall the system was able to prove the concept of a cheap and fast gesture-based interaction system. Only simple and fast techniques were used, and working applications were demonstrated using only the gestures that could be defined by the tracking of gross hand motion. The motivation for this project was to see whether building such a system was possible. This project has been successful in showing this. With further improvements and hardware availability in the future, gesture-based table-top interfaces should be available to all, and with them the hope of more flexible empowerment of people through human-computer interaction.

7 Extensions and future directions

Given more time, many possible extensions of the work in this project could be carried out. These include the following:

Improved segmentation: Use of more sophisticated background subtraction and segmentation algorithms such as that of Pilet, Strecha & Fua (2008) would allow the removal of the black card cover and increase the applicability of the system. However, such systems require more processing and hence would need to be chosen carefully to ensure the system would still run in real time.

Use of motion for matching: It was noted in Section 5.7 that natural human motions often involve moving in all planes. This implies that, when moving, a hand is likely be seen to be moving from any possible viewpoint. This may provide further information for hand matching.

Use of a probabilistic gesture recognition framework: In many of the gesture-based interaction systems described in Section 2, probabilistic frameworks for gesture recognition were used to take into account the uncertainty in how gestures are performed and in their sensing. For example, Hidden Markov Models could be used, perhaps through the Georgia Tech Gesture Toolkit specifically built for gesture recognition (Westeyn et al., 2003). These frameworks typically allow more accurate gesture recognition, although as with the improved segmentation would also require more processing, as well as requiring training of the models.

Building of test applications: Further test applications could be built to demonstrate other aspects of the system such as its potential for multi-user interaction. A general system for allowing external programs access to the system could also be built e.g. through CORBA as in the work of Taylor & Drummond (2007).

Performing user tests: In order to determine whether the use of gross hand motion gestures really did provide advantages over existing interfaces in some applications, and to determine how the appropriate the two modes of the system (hand or fingertip track-

ing) were to different applications, user tests could be performed.

Determining usability indices: The usability of the interface could also be compared to other systems by developing a usability index such as through the development of a three-dimensional extension of the steering law (see Accot & Zhai, 1997).

The work in this project has also suggested a need for future work in a number of directions:

Detection and segmentation: These are problems that occur frequently in many applications of computer vision – how can objects of interest be detected against a general background, and how can they be isolated in the image? It is likely that new ways of representing objects and their visual appearance are needed to solve these problems in general. For interface design, it is further important that such systems can be run in real time.

Hand matching: This was the most challenging section of this project, as it concerned identifying matching pairs of textureless, similarly coloured and shaped objects from the view of each camera. It is possible that further non-appearance based constraints, such as the motion constraint suggested above, could be found and provide an adequate solution to this problem. A hardware-based alternative would be the use of three calibrated cameras, such that three epipolar planes would be defined (between each pair of cameras) which would allow unique positioning in real space (except where these three planes were coplanar, though this could be avoided by appropriate positioning of the cameras). This would lead to an increased cost of the system, however, and make camera positioning more challenging.

8 Acknowledgements

I would like to thank my supervisor Dr Tom Drummond for giving me the opportunity to work on this project and for his ideas and enthusiasm. I would also like to thank the other members of the vision group for providing an enjoyable and stimulating atmosphere to work in, particularly Dr Gerhard Reitmayr and Dr Ed Rosten for their thoughts on the project throughout the year, and Simon Taylor for helping me get started in Michaelmas term.

9 Risk assessment retrospective

The only health and safety risks identified in the risk assessment were those associated with general computer-based office work. This proved to be sufficient with no aspect of the project straying from this environment, and sufficient equipment was provided for comfortable and safe working.

References

Accot, J. and Zhai, S. (1997), 'Beyond Fitts' law: models for trajectory-based HCI tasks', *Proc. ACM Conference on Human Factors in Computing Systems (SIGCHI)*, 295-302

Billinghurst, M. and Buxton, B. (2008), 'Gesture based interaction', In: *Human Input to Computer Systems: Theories, Techniques and Technology*, book in progress, available at http://www.billbuxton.com/inputManuscript.html

Buxton, B. (2009), 'Multi-Touch Systems that I Have Known and Loved', website available at http://www.billbuxton.com/multitouchOverview.html

Cipolla, R. (2006), '4F12 - Computer Vision and Robotics', Lecture notes, Cambridge University Engineering Department

Do-Lenh, S., Kaplan, F., Sharma, A. and Dillenbourg, P. (2009), 'Multi-finger interactions with papers on augmented tabletops', *Proc. Tangible and Embedded Interaction (TEI)*, 267-274

Dorfmüller-Ulhaas, K. and Schmalstieg, D. (2001), 'Finger Tracking for Interaction in Augmented Environments', *Proc. IEEE and ACM Int. Symposium on Augmented Reality (ISMAR)*, 55

Drummond, T. et al., 'Cambridge Video Dynamics (LibCVD)', C++ software library available at http://mi.eng.cam.ac.uk/~twd20/libcvdhtml/index.html

Drummond, T. et al., 'Tom's Object Oriented Numerics (TooN)', C++ software library available at http://mi.eng.cam.ac.uk/~twd20/TooNhtml/index.html

Foehrenbach, S., König, W., Gerken, J. and Reiterer, H. (2008), 'Natural Interaction with Hand Gestures and Tactile Feedback for large, high-res Displays', *Multimodal Interaction Through Haptic Feedback Workshop*

Grzeszcuk, R., Bradski, G., Chu, M.H. and Bouguet, J.-Y. (2000), 'Stereo based gesture recognition invariant to 3D pose and lighting', *Proc. Computer Vision and Pattern Recognition (CVPR)*, 826-833

Kaltenbrunner, M. and Bencina, R. (2007), 'reacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction', *Proc. Tangible and Embedded Interaction (TEI)*

Kohler, M. (1997), 'Using the Kalman Filter to Track Human Interactive Motion - Modeling an Initialization of the Kalman Filter for Translational Motion', Technical Report 629, Informatik VII, University of Dortmund

Konstantinos, D. (2004), 'A Review of Vision-Based Hand Gestures', Internal report from York University, Canada

Krueger, M., Gionfriddo, T. and Hinrichsen, K. (1985), 'VIDEOPLACE - An Artificial Reality', *Proc. ACM Conference on Human Factors in Computing Systems (SIGCHI)*, 35-40

Kurtenbach, G. and Hulteen, E. (1990), 'Gestures in Human-Computer Communications', In: Laurel, B. (Ed.) *The Art of Human Computer Interface Design*, Addison-Wesley, 309-317

Leibe, B., Starner, T., Ribarsky, W., Wartell, Z., Krum, D., Singletary, B. and Hodges, L. (2000), 'The Perceptive Workbench: Toward Spontaneous and Natural Interaction in Semi-Immersive Virtual Environments', *Proc. IEEE Virtual Reality*, 13-20

Maggioni, C. and Kämmerer, B. (1998), 'GestureComputer - History, Design and Applications', In: Cipolla, R. and Pentland, A. (Eds.) *Computer Vision for Human-Machine Interaction*, Cambridge University Press

Malik, S. and Laszlo, J. (2004), 'Visual Touchpad: A Two-handed Gestural Input Device', *Proc. International Conference on Multimodal Interfaces (ICMI)*, 289-296

Mazalek, A., Winegarden, C., Al-Haddad, T., Robinson, S. and Wu, C.-S. (2009), 'Architales: physical/digital co-design of an interactive story table', *Proc. Tangible and Embedded Interaction (TEI)*, 241-248

McFarlane, N.J.B. and Schofield, C.P. (1995), 'Segmentation and tracking of piglets in images', *Machine Vision and Applications*, 8:187-193

Microsoft Surface Virtual Pressroom (2009),
http://www.microsoft.com/presspass/presskits/surfacecomputing/default.mspx

Mitra, S. and Acharya, T. (2007), 'Gesture recognition: A survey', *IEEE Transactions on Systems, Man and Cybernetics - Part C*

Muñoz-Salinas, R., Aguirre, E., García-Silvente, M. and Gómez, M. (2007), 'Continuous Stereo Gesture Recognition with Multi-layered Silhouette Templates and Support Vector Machines', *Proc. Mexican International Conference on Artificial Intelligence*

Pavlovic, V., Sharma, R. and Huang, T. (1997), 'Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677-695

Pearson, K. (1901), 'On the lines and planes of closest fit to systems of points in space', *Philosophical Magazine*, 2:559-572

Pilet, J., Strecha, C. and Fua, P. (2008), 'Making Background Subtraction Robust to Sudden Illumination Changes', *Proc. European Conference on Computer Vision (ECCV)*

Quek, F. (1994), 'Toward a Vision-Based Hand Gesture Interface', *Virtual Reality Software and Technology Conference*, 17-31

Reitmayr, G. and Drummond, T. (2005), 'Localisation and interaction for augmented maps', *Proc. IEEE and ACM Int. Symposium on Mixed and Augmented Reality (ISMAR)*, 120-129

Salvador, S. and Chan, P. (2004), 'FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space', *KDD Workshop on Mining Temporal and Sequential Data*, 70-80

Song, H., Grossman, T., Fitzmaurice, G., Guimbretière, F., Khan, A., Attar, R. and Kurtenbach, G. (2009), 'PenLight: combining a mobile projector and a digital pen for dynamic visual overlay', *Proc. ACM Conference on Human Factors in Computing Systems (SIGCHI)*, 143-152

Starner, T. and Pentland, A. (1998), 'Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371-1375

Steimle, J. (2009), 'Designing pen-and-paper user interfaces for interaction with documents', *Proc. Tangible and Embedded Interaction (TEI)*, 197-204

Stenger, B., Woodley, T., Kim, T.-K., Hernández, C. and Cipolla, R. (2008), 'AIDIA - Adaptive Interface for Display InterAction', *Proc. British Machine Vision Conference (BMVC)*

Taylor, S. and Drummond, T. (2007), 'Natural Interaction for Table-Top Environments', Fourth year project report, Cambridge University Engineering Department

Weiser, M. (1991), 'The computer for the 21st Century', *Scientific American* 265(3):94-104

Welch, G. and Bishop, G. (2001), 'An Introduction to the Kalman Filter', *ACM Conference on Human Factors in Computing Systems (SIGCHI) Course Notes*, Course 8

Westeyn, T., Brashear, H., Atrash, A. and Starner, T. (2003), 'Georgia Tech Gesture Toolkit: Supporting Experiments in Gesture Recognition', *Proc. International Conference on Multimodal Interfaces*

Wobbrock, J., Morris, M. and Wilson, A. (2009), 'User-Defined Gestures for Surface Computing', *Proc. ACM Conference on Human Factors in Computing Systems (SIGCHI)*

Wu, Y., Lin, J. and Huang, T. (2001), 'Capturing Natural Hand Articulation', *Proceedings of the International Conference on Computer Vision (ICCV)*

Ye, G., Corso, J.J. and Hager, G.D. (2004), 'Gesture Recognition Using 3D Appearance and Motion Features', *Computer Vision and Pattern Recognition (CVPR) Workshop*, 160

Zufferey, G., Jermann, P., Auréllen, L. and Dillenbourg, P. (2009), 'TinkerSheets: using paper forms to control and visualize tangible simulations ', *Proc. Tangible and Embedded Interaction (TEI)*, 377-284