3D TRACKING OF NON-RIGID ARTICULATED OBJECTS

Djambazian Haïg

Department of Electrical and Computer Engineering
McGill University, Montreal

November 2001

A Thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering

Abstract

Articulated objects can be found in many living beings. Tracking is essential if we want to interpret the behavior of such objects. This thesis describes a framework for learning the relationship between the state and the appearance of the object. It will also show how to use this representation to track the state of the articulated object.

The learning phase of the method results in populations of models that describe the appearance of small regions of the object for small regions of the state space. To efficiently train off-line, it is necessary to model the appearance of the object as function of the state. The local models use Principal Component Analysis (PCA) on windowed regions of the projected object. Manifolds in PCA subspace represent the appearance of the small local regions as they undergo deformations.

The tracking algorithm recursively matches the link appearances while searching in the state space of the articulated object. To match the object appearance to the model, a coarse search finds the models that are active. The error of the projected object image is then minimized (at the new unknown state) in model subspace by fine-tuning the state.

Algorithm performance is evaluated on real and synthetic data of a 4 d.o.f. finger following arbitrary 3-D paths. The results show that the local PCA models capture the deformations successfully even after discarding some of the bases. These deformations account for key features that are essential to the matching process. Also, the way in which the appearance data is partitioned allows for a fast and efficient caching strategy, thus allowing the algorithm to meet real-time constraints. Finally the merging of predictions and observations makes the algorithm very robust to outliers.

Résumé

Un grand nombre de formes de vies comportent des articulations. Si l'on cherche à interpréter le comportement de ces êtres, il est essentiel que l'on puisse suivre leur position et leur pose (configuration de l'objet). Cette thèse présente une méthode pour apprendre la relation entre la pose d'un objet et l'apparence de ce même objet. Cette relation entre la pose et l'apparence nous sert ensuite dans le développement d'un algorithme pour suivre la pose de l'objet.

Grâce à la phase d'apprentissage de l'algorithme on obtient des populations de modèles qui décrivent l'apparence de petites régions sur la surface de l'objet. Chacun de ces modèles sont définis pour de petites portions de l'espace qui contient les configurations possibles de l'objet. Un modèle précis est initialement construit. Ce modèle nous premettra d'apprendre l'apparence de l'objet de manière automatisée. Chaque modèle est obtenu par une Analyse en Composantes Principales (ACP) des différentes régions de l'objet projeté. Des courbes dans l'espace défini par l'ACP nous permet de décrire les déformations de régions distinctes de l'objet projeté.

Pour déteterminer la pose de l'objet, l'algorithme trouve récursivement les angles associés avec chaque articulation. Pour faire ceci l'algorithme commence par trouver les modèles qui seront utilisés. Puis il s'agit de chercher dans l'espace même de ces modèles (en projetant l'image percue dans l'espace de l'ACP) pour trouver l'apparence apprise qui ressemble le plus à l'observation faite par le biais de la caméra.

L'algorithme est évalué sur des images réels et synthétiques d'un doigt avec 4 degrés de liberté éffectuant des mouvements alléatoires.

Acknowledgments

I must begin by acknowledging my colleagues in the Artificial Perception Lab at McGill University. Philippe and Louis Simard, Marcel Mitran, Stephen Benoit, Isabelle Bégin and Peter Savadjiev. They have always made time to discuss the sanity of new ideas. I must also express my sincere appreciation to my thesis supervisor, Frank P. Ferrie, for sharing his wisdom and his experience while giving me advice. I would also like to thank him for giving me the artistic freedom necessary to realize this task.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

Articulated objects can be found in many living beings. Tracking is essential in the interpretation of the behavior of such objects. This paper describes a framework for learning the relationship between the state and the appearance of the object. This representation will be used in the tracking of the articulated object state.

The articulated object chosen to illustrate the algorithm is a finger with 4 degrees of freedom (d.o.f.). Figure 1 shows an example of the two views of the user's finger that are input to the tracking algorithm.

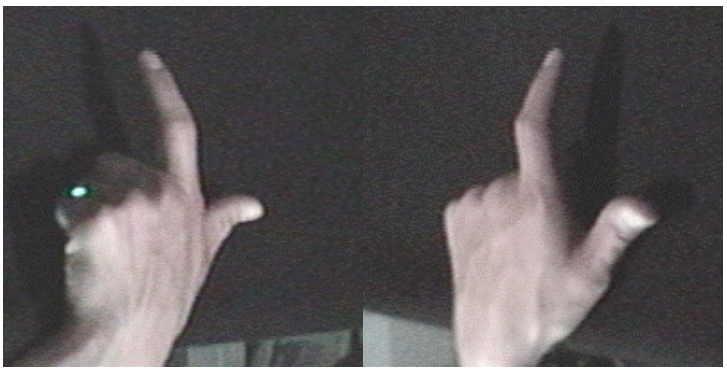

Figure 1 Input Stereo Images of Object

The solution we are seeking is the angle of the finger's joints. The four joint angles corresponding to the four d.o.f. of the finger define the state of the finger. Figure 2 shows the pose solution of the stereo pair in Figure 1. The stick figures superimposed on the images, corresponding to the skeleton of the finger, represent the state estimated by the algorithm.

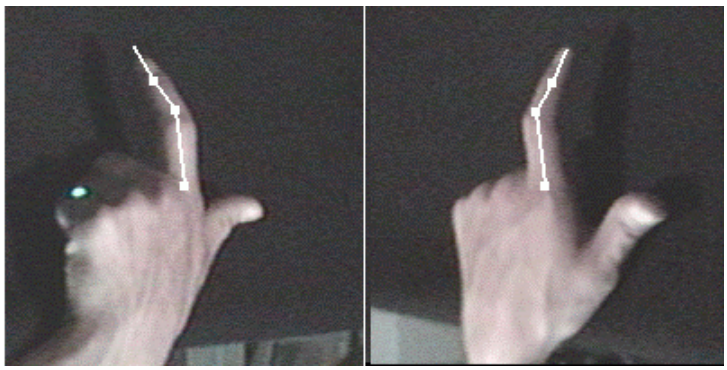

Figure 2 Input Images with Overlay of Solution

Although the kinematic properties of the object in Figure 1 are well defined, many variables remain and make the tracking task difficult. For example when solving for the first link, which has 2 d.o.f., a single view would not be enough to solve for the angles. This is because motions in the view direction look similar. Another difficulty is the non-rigid properties of the object. As the object is being tracked the appearance of the object is perpetually changing. This prevents simplifications such as approximating the fingers by fixed geometrical shapes. The scene lighting is also an important factor as it changes the object's appearance.

The generative model presented in this thesis simultaneously accounts for the deformations, the lighting, the object's texture and the perspective projection of the cameras. A realistic environment is first created, simulating the object as well as the lighting conditions. This environment is then used to train and construct the generative model of the object appearance. The generative model represents each link independently. Essentially the problem is to invert the generative model, i.e., to recover the model parameters from local measurements of its appearance in an image. This is accomplished by partitioning the state space of the model into regions. Each region is such that local changes in appearance can be unambiguously tied to local variations of the

model parameters. In other words, for each partition of the parameter space there exists a corresponding appearance model that provides a one to one mapping to a local region of the image. If the approximate state of the model is known, then it is straightforward to localize the image regions for each link and determine the corresponding model parameters. This scheme is completely general in that it can recover arbitrary states (poses) of the model. This is essential for our tracking purposes. Meeting real-time requirements is another important aspect for tracking. The generative model is very well suited for this since it requires a minimal amount of data to represent a given state.

In the following Section our approach will be compared to work previously done in the field. Some of the approaches presented are very common, however our approach distinguishes itself on a number of key points.

1. Previous Work

Tracking articulated objects using image sequences poses many challenging problems. Some of these problems are specific to tracking; others are common to all computer vision problems.

One inherent characteristic of articulated objects is the high dimensionality of their configuration space. This makes real-time requirements difficult to meet. However there are ways to reduce the amount of computation required. Very often knowledge about object structure is embedded in the algorithm. In [4] Deutscher and Blake perform tracking of a human body pose. They propose an iterative particle-filtering algorithm that uses a segmented image as well as an edge-detected image to fit a body-model to the observed data. In the method the human body shape is built with a kinematic chain

holding together different conic sections. This model is used to find which image pixels to test while fitting. De la Torre and Black [3] go a step further by constructing a generative model of the object's appearance. They use this model to track the state of an articulated body. This appearance model is constructed by approximating each limb by a cylinder; these are connected according to the kinematic structure of the human body. To take into account the appearance of the body, the appearance of each limb is mapped to its corresponding cylinder. Different views of the limb must be merged to have a complete representation of the limb appearance. In [15] Wu and Shah design a system that allows the user to input a 3-D motion to the computer. Their algorithm involves tracking the user's fingertip in 2-D. The 2-D motion is used to define a 3-D motion thanks to the spherical inverse kinematics of the user's arm. Delamarre and Faugeras [2] attempt to determine hand pose using stereo. Their technique first computes depth data from the stereo views, followed by the fitting of a 3-D hand model to the data. The 3-D hand model is composed of truncated cones (phalanges) and spheres (joints). In the proposed approach the exact structure of the articulated object is used. The appearance of the object is modeled in two parts: the texture of the object and the deformations of the object.

In [7] Hauck and Lanser propose to recognize rigid articulated objects. Their method consists of matching an object model to an edge-detected image. This matching is performed by hierarchically determining the position of the different parts of the object. For example in the simple case of a door, their algorithm would first find the door frame, followed by the door position with respect to the door frame and finally the door handle position with respect to the door. Kwon and Zhang in [10] exploit the hierarchical

structure of the human hand to track its state. Their method first determines the pose of the hand palm, which in turn allows them to find the joint angles of the fingers. The main appeal of recursive structures is that the problem becomes linear in the number of degrees of freedom of the object. To use this recursive formulation in the work presented in this thesis, the appearance of each model link is trained for separately.

Another commonly used method to reduce the search space is to learn or predefine possible or probable behaviors. Alberola and Juan's [1] goal in their gesture recognition system was to recognize gestures specific to the Spanish alphabet of the hearing impaired. They use a Finite State Machine to recognize key hand gestures. They also detect intermediate states between gestures. This allows them to efficiently track the hand between gestures. In [14] Sato and Kobayashi use the silhouette (obtained by an IR camera) of the hand to track the palm and the fingertips. Their system determines the 2-D position of the hand and the visible fingertips reducing the range of possible hand poses. Min and Yoon [13] use Hidden Markov Models to recognize similar static gestures. However instead of determining the absolute 2-D location of the users hand, they predefine a set of dynamic gestures from which solutions are drawn. The problem then becomes one of estimating how close the hand trajectory is from one of the predefined paths. In [6] Gonclaves and Di Bernardo propose a similar but more fine-grained approach. In their method they define a low-level motion that can be successively used to estimate the actual hand motion of the user. One drawback in this approach is that motion discontinuities could be observed when switching from one motion model to another. A continuous solution space allows for a broader class of applications. The

method presented in this thesis does not constrain the problem using any application specific information.

Visual markers worn by the person or tracked object can provide very robust information regarding different key positions on the body. In [6] the authors are able to track the complete pose of the user's body in 2-D by having fluorescent ping-pong balls placed at joint locations. In [1] color-coded rings are placed around the user's fingertips and hand joints to determine the pose of the hand in 3-D. Kwon and Zhang [10] are able to determine the 3-D position of the palm and the position of the fingers with a single camera. They achieve this by having the user wear a cardboard glove painted with circles. These markers can sometimes be a little invasive but they offer a reliable estimate of joint locations.

Implicit in all these approaches is the problem of recovering 3-D information from 2-D images. Some authors [6, 13, 14, 9] choose to determine and interpret the 2-D motion of the tracked objects. However we can regularize this ill-posed problem to extract 3-D motion. Information about the world constrains what we can possibly see. Kwon and Zhang [10] use a glove with a well-determined shape and features. Other authors [7, 15, 16, 3, 4] directly integrate kinematic constrains into their algorithm.

The use of more then one camera can also help regularize the problem. Delamarre and Faugeras [2] attempt to find the pose of a hand using stereo views. First they use the stereo pair to reconstruct a dense 3-D scene. In a second phase they fit a hand model to the 3-D data from which they can extract joint angles. The solution presented in this thesis is formulated for multiple arbitrary viewpoints.

Modeling non-rigidity can make tracking algorithms more robust. Heap and Hogg [8] address the non-rigidity aspect of the human hand for tracking purposes. Their method relies on a Point Distribution Model. This deformable model is trained with edge-detected images of the hand in different configurations. Tracking is performed by finding which pose and model parameters best align the model to the observed image. Same as in the training phase, the observed image is edge-detected and only model points that lie on the boundary of the projected model are used in the comparison. The appearance model presented in this thesis uses PCA to learn the non-rigid behavior of the object.

Occlusion is a problem common to all tracking algorithms. It occurs when the tracked object completely or partly disappears behind another object. Since feature matching is at the core of object tracking, most algorithms will not work if too many features are hidden. In the case of articulated objects we also have to deal with self-occlusions where the object itself obstructs the view to other parts of the object. Hauck and Lanser [7] handle occlusions by predicting which features are potentially occluded. If enough features are guaranteed to be visible they proceed with their estimation. In theory self-occlusions should be easier to handle since we are tracking not only the occluded object but also the object that is causing the occlusion. In other words we know when and where occlusions and disocclusions will occur. In [11] Lathilière and Hervé designed a system to track hand pose. They handle occlusions by defining a visibility table. This table ranks the fingertips by order of visibility to the camera. It also indicates which fingers might be occluded. Our appearance model inherently handles self-occlusions. It is also designed with occlusion handling in mind when tracking multiple fingers.

Arbitrary backgrounds behind the tracked object being can sometimes give us false positives in the feature matching process. Very often authors constrain the background color so that it is easily ignored. Sato And Kobayashi [14] allow for an arbitrary background in their hand tracking application through the use of an infrared camera. This camera can only see the hand since the background does not have the same temperature. This allowed them to project images on top on the background in their Augmented Desk Interface.

Most methods presented here assume detailed *a priori* information regarding the object. Incorporating too much information can sometimes significantly slow down the algorithm. In our approach the non-rigidity and the appearance are integrated while at the same time upholding real-time constraints.

2. Outline of Thesis

The remainder of the thesis is organized as follows. Chapter 2 shows how a generative model of the articulated object's appearance can be designed. This generative model is trained using a precise finger model rendered using parametric surfaces, texture mapping and perspective projection. Chapter 3 explains how this appearance model is incorporated in the tracking framework. The mechanisms involved in the tracking scheme, such as the predictive model, and the Kalman filtering, are also presented. In Chapter 4 the tracking results are presented and discussed. The results include image sequences obtained by simulation and from real camera images. Here the strong points and the limitations of the algorithm are discussed. Chapter 5 concludes by reiterating the contributions of this method and proposes future research directions.

3. Contributions

The contributions claimed in this thesis are the following:

- A novel framework for learning a generative model of non-rigid articulated object's appearance. The model is formulated for any number of arbitrary viewpoints and any type or articulated object.
- The appearance model is designed to have a continuous response. Appearances can be generated for any state chosen from the real numbered state space. This allows for tracking general motions.
- A caching strategy adapted for the generative model. This allows the model to be drawn very quickly for a given object state while tracking.

CHAPTER 2

Modeling Object Appearance

An important component of the tracking algorithm presented here is the observation model. Modeling the appearance of the object is essential to the observation model. The following sections will show how a realistic non-rigid finger was modeled. This realistic model is used in to generate another appearance model that is more suited for tracking purposes. The intuition behind the appearance model will also be presented.

1. Non-Rigid Articulated Objects

The underlying structure of the object is simple. It consists in a 4 degree of freedom (d.o.f.) 3 link kinematic chain. Figure 3 illustrates the degrees of freedom of the links. The first link has 2 d.o.f. and the two others have 1 d.o.f. The 4 parameters α, β, γ and omega ω completely define the state of the finger.

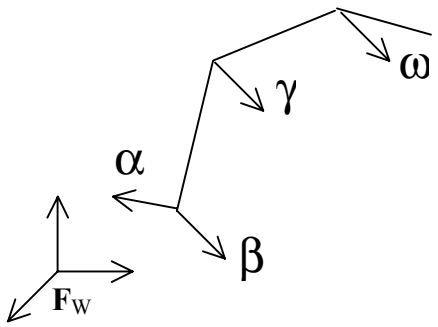

Figure 3 Kinematic Chain

Figure 4 shows the locations of the reference frames associated with each link.

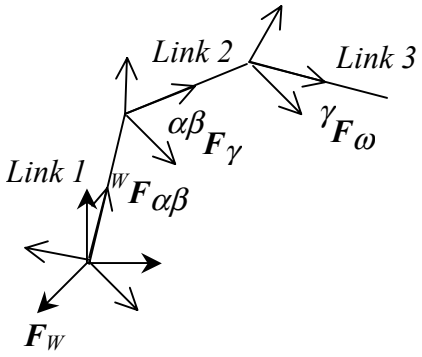

Figure 4 Link Reference Frames

Explicit form of the 3 frames is given in Equations 1 to 3 using homogeneous transformation matrices. Equation 1 shows the *Link 1* frame to the world frame transformation, Equation 2 shows *Link 2* frame to *Link 1* frame transformation and Equation 3 *Link 3* frame to *Link 2* frame transformation.

$$ {}^{W}F_{\alpha\beta} = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} \cos(\beta) & -\sin(\beta) & 0 & 0 \\ \sin(\beta) & \cos(\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

$$ {}^{\alpha\beta}F_{\gamma} = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

$$ {}^{\gamma}F_{\omega} = \begin{bmatrix} \cos(\omega) & -\sin(\omega) & 0 & 0 \\ \sin(\omega) & \cos(\omega) & 0 & 0 \\ 0 & 0 & 1 & L_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

Using these matrices point coordinates can be projected from one frame to another. For example if ${}^{\omega}\boldsymbol{p}=[p_x\ p_y\ p_z\ 1]^T$ is defined in the ${}^{\gamma}\boldsymbol{F}_{\omega}$ frame it can also be expressed it in the $\boldsymbol{F}_W$ frame. The expression of ${}^{W}\boldsymbol{p}$ in the $\boldsymbol{F}_W$ frame is shown in Equation 4.

$$ {}^{W}p = {}^{W}F_{\alpha\beta}\cdot{}^{\alpha\beta}F_{\gamma}\cdot{}^{\gamma}F_{\omega}\cdot{}^{\omega}p \tag{4}$$

These matrices can be used to project non-rigid object control points from link frames to the world frame $\boldsymbol{F}_W$.

1.1 Non-Rigid Object Model

Parametric bicubic surfaces are chosen to model the deformable parts of the object. Hermite curves will be used to illustrate the properties of bicubic surfaces. To draw a Hermite curve two control points and two control vectors must be specified in 3-D. Figure 5 shows different curves that are drawn in 2-D ignoring the z component of the curve and the control vectors and points. To generate these different curves only the magnitude of one of the control vectors (R_1) was changed.

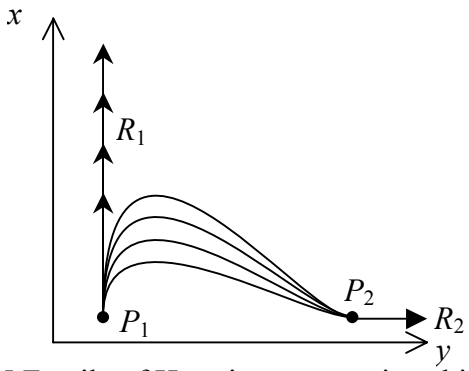

Figure 5 Family of Hermite parametric cubic curves

This illustrates the appeal of this type of mathematical construct where changing a small number of parameter affects the overall shape of the object. Bicubic surfaces can be

deformed in a similar way as cubic curves. However they require a greater number of control points, as surfaces are more complex then curves.

The Hermite cubic curve Equation will now be derived. This will help in understanding the bicubic surface formulation.

1.1.1 Parametric Cubic Curves

In the following Equations only the x component of the 3-D curve will be considered. The expression for the x component of the curve is shown in Equation 5,

$$\begin{aligned} x(t) &= [t^3 \quad t^2 \quad t^1 \quad t^0] \bullet [a \quad b \quad c \quad d]^T \quad 0 \le t \le 1, \\ x(t) &= T \bullet C_x . \end{aligned} \tag{5}$$

C_x contains the polynomial coefficients that define the curve. The 3-D curves must verify certain boundary conditions given by the Hermite geometry vector. These conditions allow us to find the polynomial coefficients. These conditions permit determination of the polynomial coefficients and are shown in Equation 6. The vector in Equation 6 contains the x component of the start and end points of the curve (P_1 and P_2) as well as the start and end tangent vectors (R_1 and R_2).

$$G_{Hx} = \begin{bmatrix} P_1 \\ P_2 \\ R_1 \\ R_2 \end{bmatrix}_x \tag{6}$$

The expression for the time derivative of Equation 5 is given in Equation 7. The boundary condition equations are shown in Equation 8.

$$\frac{d}{dt} x(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \bullet C_x \tag{7}$$

$$\begin{bmatrix} x(0) \\ x(1) \\ \left.\frac{d}{dt}x(t)\right|_{t=0} \\ \left.\frac{d}{dt}x(t)\right|_{t=1} \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ R_1 \\ R_2 \end{bmatrix}_x \tag{8}$$

Using Equations 5, 7 and 8 C_x can be solved for. This is shown in Equation 9.

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \bullet C_x = \begin{bmatrix} P_1 \\ P_2 \\ R_1 \\ R_2 \end{bmatrix}_x \tag{9}$$

Solving for the polynomial coefficients the Hermite Basis matrix M_H is found. As shown in Equation 10 the polynomial coefficients can be obtained from the geometry vector G_H. Using Equation 5 the x component of the Hermite cubic curve can be plotted.

$$\begin{aligned} C_x &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \bullet \begin{bmatrix} P_1 \\ P_2 \\ R_1 \\ R_2 \end{bmatrix}_x \\ C_x &= \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \bullet \begin{bmatrix} P_1 \\ P_2 \\ R_1 \\ R_2 \end{bmatrix}_x \\ C_x &= M_H \bullet G_{H_x} \end{aligned} \tag{10}$$

The same expression holds for the y and z components of the curve given the y and z counterparts of the geometry vector.

1.1.2 Parametric Bicubic Surfaces

As previously mentioned, bicubic surfaces can be used to model deformable objects. The main appeal is that it allows a small numbers of parameters to modify an entire surface.

Bézier surfaces can be used to render a surface using 16 control points. For the derivation of the formulas we refer the reader to [5]. Equation 11 states the Bézier bicubic formulation, t and s are both defined over the [0, 1] interval,

$$\begin{aligned} x(s,t) &= S \cdot M_B \cdot G_{B_x} \cdot M_B^T \cdot T^T, \\ y(s,t) &= S \cdot M_B \cdot G_{B_y} \cdot M_B^T \cdot T^T, \\ z(s,t) &= S \cdot M_B \cdot G_{B_z} \cdot M_B^T \cdot T^T. \end{aligned} \tag{11}$$

The matrix G_{Bx} contains the x component of the control points. An example Bésier bicubic surface is shown in Figure 6. The 16 control points defining this surface are also labeled.

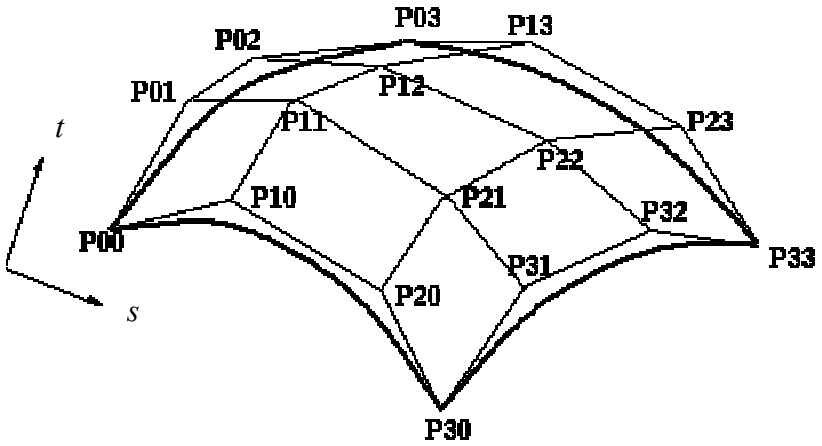

Figure 6 Bicubic Surface Control Points

In the previous section the *Hermite Basis matrix* M_H was defined. In Equation 11 M_B plays a similar role as M_H, however it is called the *Bésier Basis matrix*. M_B maps the

control points to the polynomial coefficients. The polynomial in the bicubic case is a little more complex then in the cubic case. In fact it is a linear combination of the terms in matrix A of Equation 12. Again such a polynomial would have to be defined for each coordinate.

$$T = \begin{bmatrix} t^3 & t^2 & t^1 & t^0 \end{bmatrix}$$
$$S = \begin{bmatrix} s^3 & s^2 & s^1 & s^0 \end{bmatrix} \tag{12}$$
$$A = S^T \cdot T = \begin{bmatrix} s^3t^3 & s^2t^3 & s^1t^3 & s^0t^3 \\ s^3t^2 & s^2t^2 & s^1t^2 & s^0t^2 \\ s^3t^1 & s^2t^1 & s^1t^1 & s^0t^1 \\ s^3t^0 & s^2t^0 & s^1t^0 & s^0t^0 \end{bmatrix}$$

In the following section bicubic surfaces will be used to construct the non-rigid finger.

1.1.3 Control Point Computation

As mentioned previously changing the location of the control points will deform the surfaces and hence the object shape. To model the finger six bicubic surfaces are used, two per link. Each control point location in the link frame depends on the pose of the kinematic chain.

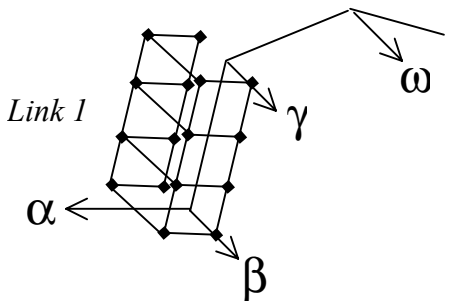

Figure 7 Bicubic Surface Control Points

For example Figure 7 shows a possible control point configuration of one of the two surfaces belonging to *Link 1*. In the case of *Link 1* some of the control points will change location as α, β and γ vary, *Link 3* control points would not be modified.

Control points require special attention to avoid gaps and discontinuities between surfaces. To achieve this G^1 geometric continuity was respected. G^1 continuity is maintained if the unit tangent vectors are identical at surface junctions

The resulting finger shape is shown in Figure 8. The bicubic surfaces have been rendered using the OpenGL graphic library. OpenGL's routine simply requires 16 control points to render each of the six surfaces.

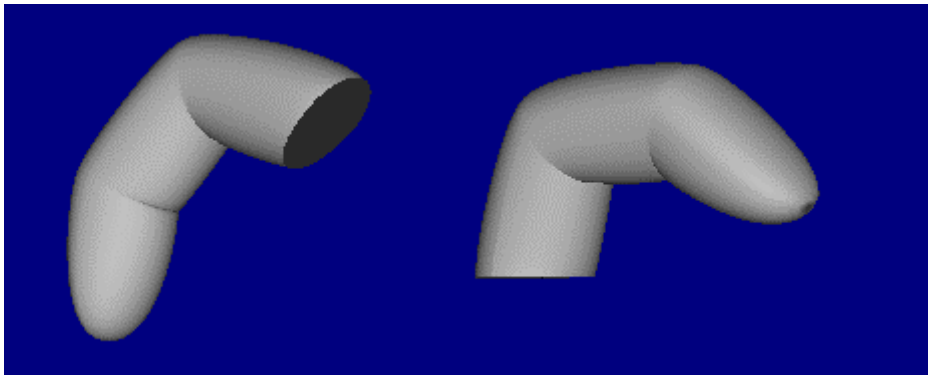

Figure 8 Finger Model Using 6 Bézier Bicubic Surfaces

Now that the shape of the object is defined with respect to the state, the appearance of the finger must also be taken into account.

1.2 Texture Mapping

Texture mapping will give a realistic appearance to the bicubic surfaces (*S* and *T*). Additionally to providing an image, texture mapping requires a set of image coordinates that will be used in the mapping.

Figure 9 shows two pictures of a real finger used as textures, a front view and a back view.

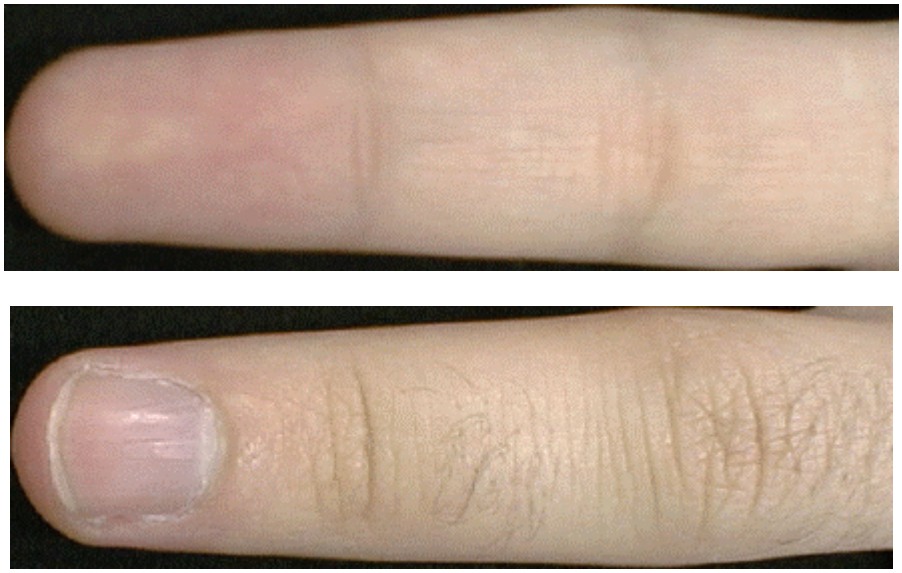

Figure 9 Different Views of Real Finger

The texture mapped model should have the same appearance as the Figure 9 pictures for identical views (top and bottom). To achieve this the previously defined x and y Bézier surface coordinates are used. The x and y texture coordinates for the top of *Link 1* can be seen in Figure 10 (line intersections) where they have been overlaid on the texture image for the top of *Link 1*. These coordinates will achieve the desired warping effect during the mapping.

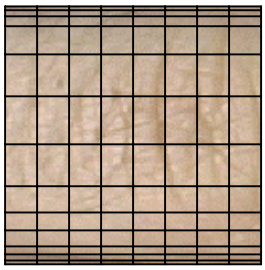

Figure 10 Texture Coordinates for Texture Mapping

The final results of the finger with the textured Bézier surfaces are shown in Figure 11.

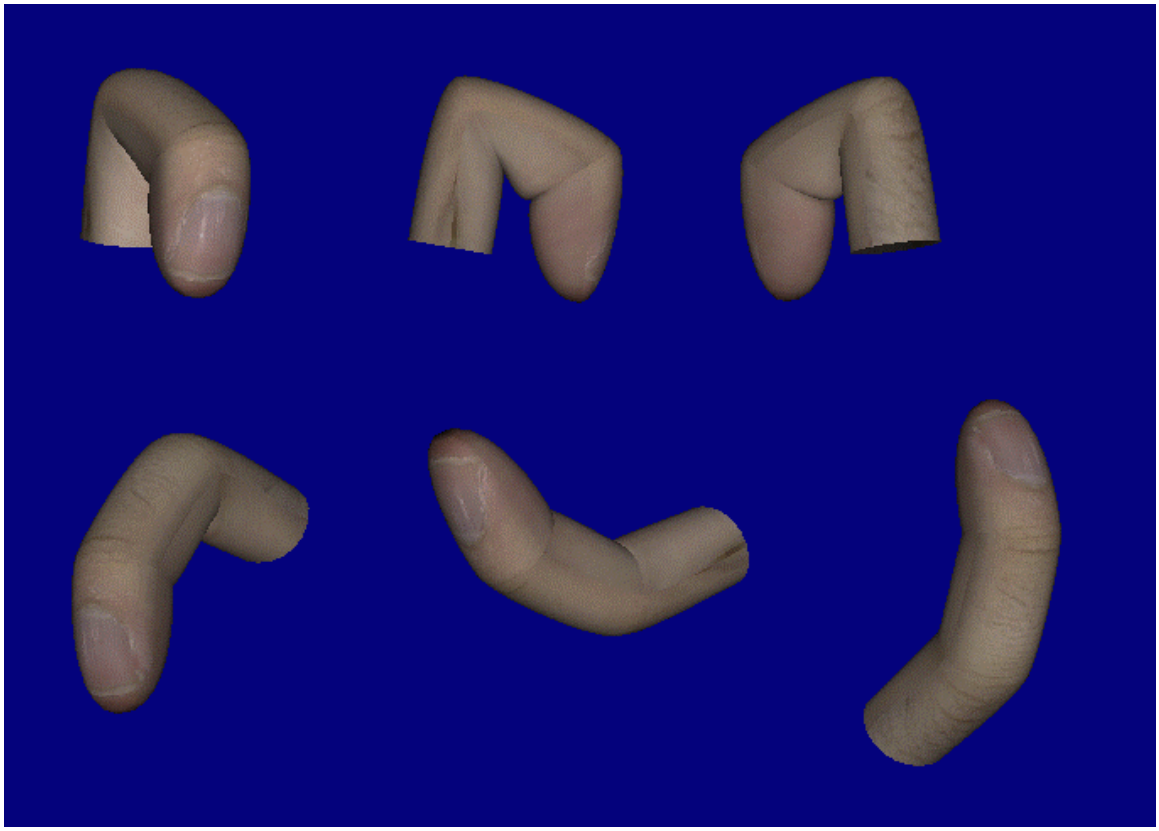

Figure 11 Different Views of OpenGL generated Finger

Figure 11 clearly illustrates the non-rigid behavior of the finger. Note that the top finger cusps are more visible then the bottom finger cusps. The texture mapping was generated using the OpenGL library.

1.3 Perspective Projection

To render the above model in 3-D, the camera is modeled as a pinhole camera with focal length *f*. Assuming that the camera and the object reference frame are as shown in Figure 12, the perspective projection consists of applying the transformation in Equation 13.

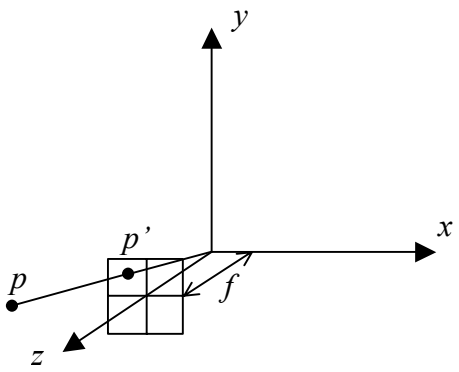

Figure 12 Perspective Projection

$$p'_x = \frac{f}{p_z} p_x$$
$$p'_y = \frac{f}{p_z} p_y \qquad (13)$$

The p_x and p_y components of the 3-D point are scaled proportionally to the depth p_z. The coordinates p'_x and p'_y will fall on the window drawn in Figure 12. The images seen in Figure 11 are generated by OpenGL using the perspective projection.

2. Learning The Appearance Model

Now that a realistic finger can be generated for arbitrary states a more compressed representation will be formulated. This new representation will not require perspective projection, texture mapping or rendering bicubic surfaces. These requirements will speed up the rendering speed considerably. This new appearance model will also attempt to learn the shading effect caused by a fixed light source. The appearance model can be formulated for many cameras. Here it will be explained for one.

The appearance model encapsulates the ***where*** and the ***what*** of the articulated object. Simply stated: ***where*** we see ***what***. The intuition behind this idea will now be presented.

2.1 Where

Figure 13 illustrates the concept of ***where***. Each link has an associated cylinder and spherical end caps. The cylinder has roughly the same diameter as the finger link. As shown in Figure 13, these shapes are used to define 3-D points uniformly distributed around the finger model. These points are projected onto the viewing plane if they are visible. Small windows are then defined on the viewing plane centered about the projected points and aligned with the projected link segment. One of these windows is labeled win_j in Figure 13 (c).

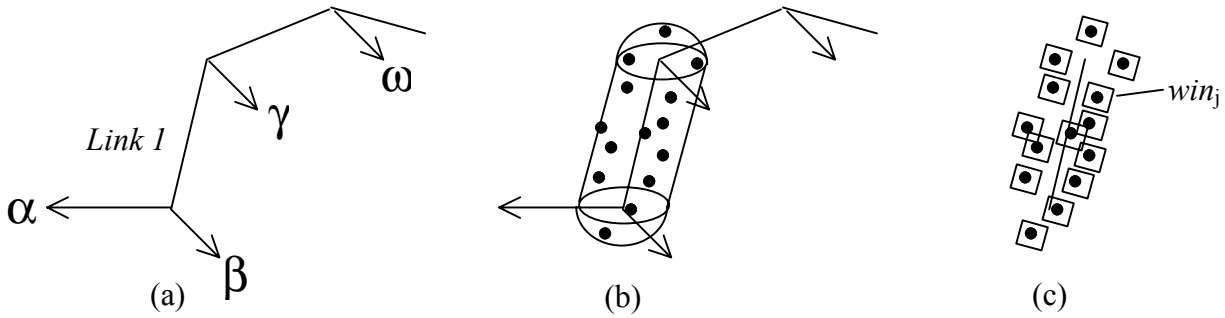

Figure 13 Window Locations

The location of these windows is determined by the state of the articulated object and the camera position in the world frame $\boldsymbol{F}_W$. The main appeal of these windows is that they each focus on a local region of the object.

2.2 What

Principal Component Analysis (PCA) is used to learn the behavior of the regions seen through the windows (***what***). To train with PCA a set of training images must be provided. PCA extracts a mean and a basis from these training images. Ideally this mean and basis can be combined to reconstruct any image in the training set. Points in the basis subspace represent images in the training set. Furthermore each vector in the PCA basis will not necessarily account for a lot of information. Even if some eigenvectors are discarded (the ones with small eigenvalues) the original images can still be reliably reconstructed with those remaining.

This greatly reduces the amount of information required to represent all the training images.

2.2.1 Principal Component Analysis

The windows defined in the previous section allow us to collect the training images. First a cluster of nearby object states is chosen. This cluster is centered around a mean state, and points are uniformly chosen around this mean. Then for each point in this cluster, images are collected from each window.

Figure 14 illustrates how the object configuration space is partitioned. Note that for the first link (with 2 d.o.f.) the space can be represented in 2-D. The dimensionality is equal to the number of d.o.f. of the parent links added to the number of d.o.f. of the link being trained. For *Link 2* the space will be 3-D and for *Link 3* 4-D.

The gray area represents a region in space from which the cluster of space points is drawn. Later on, when reconstructing the appearance, the correct local models will be

found by determining which one of these regions the object state belongs to. Clusters corresponding to different μ_α and μ_β values are chosen as to overlap each other to better cover the configuration space. The advantage of choosing nearby points is that the finger region seen from win_j will have similar deformations. This similarity will make the PCA technique more susceptible to learning during the training.

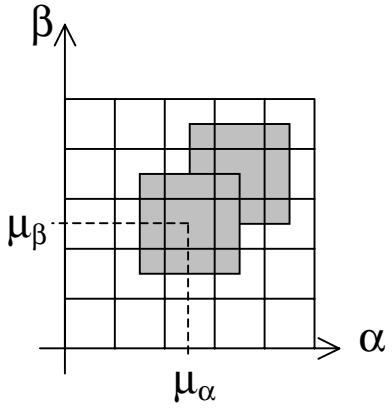

Figure 14 Configuration Space Partition

It is preferable that win_j stay visible for each sample in the cluster. As mentioned before, for a given object pose some windows will be behind the object and thus ignored. Therefore for a given cluster of points all windows that are not visible for all the cluster points are discarded. This insures that the PCA model associated with each window will be valid over the whole partition of interest. To recapitulate, for each region in Figure 14 and each valid window a PCA model is generated.

Each PCA model consists of a mean and a basis (eigenvectors). The first step of performing PCA involves formatted the training data. Equation 14 shows how each image is represented by one vector $\boldsymbol{x}_i$.

$$x_i = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_N \end{bmatrix} \tag{14}$$

As shown in Equation 15 the mean is a vector computed from the training image vectors.

$$\bar{x} = \frac{1}{M} \sum_{i=1}^{M} x_i \tag{15}$$

The mean vector (representing the mean image of the training set) is then subtracted from each training image, resulting in the component of images that deviate from the mean. Equation 16 shows how the resulting images are arranged to form a single matrix *X*.

$$X = \left[(x_1 - \bar{x}) \quad (x_2 - \bar{x}) \quad (x_3 - \bar{x}) \quad \ldots \quad (x_M - \bar{x}) \right] \tag{16}$$

The $N \times M$ matrix *X* is used to compute a $N \times N$ symmetric covariance matrix from which the eigenvectors (and eigenvalues) are extracted. The covariance matrix is shown in Equation 17.

$$C = X \cdot X^T \tag{17}$$

The eigenvalues (λ_i) and corresponding eigenvectors ($\mathbf{e}_i$) are solutions to Equation 18. Solving eigenvalues and corresponding eigenvectors is a non-trivial task, and many methods exist.

$$C \cdot e_i = \lambda_i e_i, \; i = 1 \ldots N \tag{18}$$

The method used here first transforms the covariance matrix into Tridiagonal form. The eigenvalues and eigenvectors are then extracted from the Tridiagonal matrix. Window images associated with cluster points can be represented by points in the PCA model subspace. This subspace is formed by the eigenvectors.

The remaining task is to find a (continuous) function that maps the configuration space points to the subspace points.

2.2.2 PCA Subspace Manifolds

Figure 15 illustrates the PCA training process. PCA provides us with a new representation for the training images. However this representation is not complete unless a mapping between configuration space points and subpace representation points is defined.

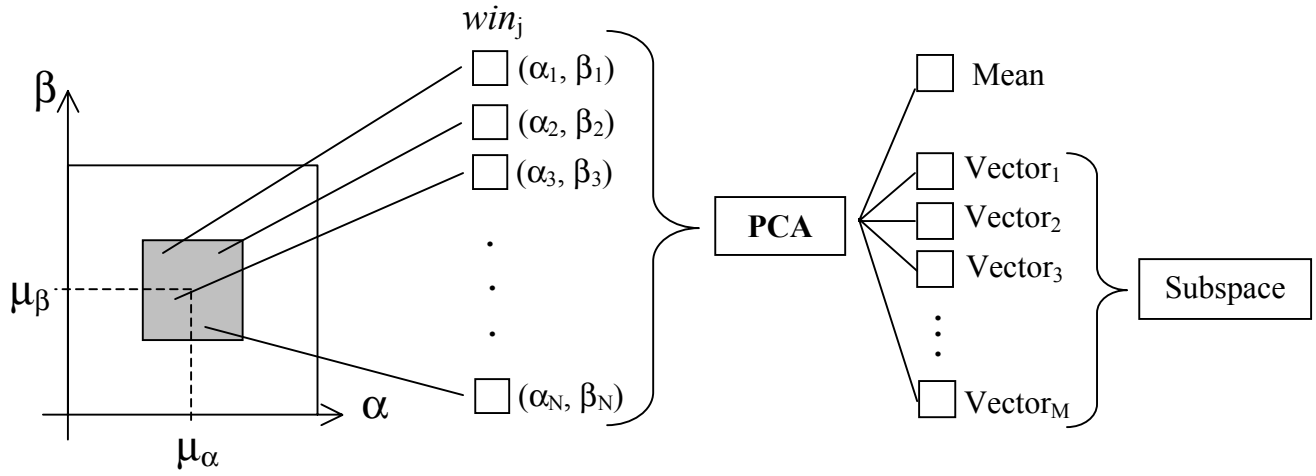

Figure 15 PCA Training Process

The M-dimensional subspace points associated with the (α_i, β_i) points define M discrete distributions defined over α and β (α and β being constrained to a specific region). However a continuous distribution is preferable, allowing us to generate the objects appearance for any object state. To achieve this a polynomial function is fit to the discrete data using the Least-Squares method.

These M polynomial functions form a subspace point parameterized by α and β. In the following section these functions will be used to generate the learned object appearance.

2.3 Rendering The Appearance Model

This section will present the steps involved in the reconstruction of the objects appearance for a given state (α, β, γ and ω). The reconstruction will be illustrated for the first link. The reconstruction of the others links is analogous.

The first step involves determining which partition of the configuration space the state belongs to. This partition allows us to know which windows are going to be used with their associated PCA models and polynomial functions. This information was saved during the training. Information about the ***where*** is essentially the window positions. The ***what*** information is contained in the PCA models with the polynomial functions.

Equation 19 shows how to reconstruct the appearance of a single window. The polynomial function $f_i(\alpha, \beta)$ is a scalar that weights $Basis_i$ according to its contribution to the image. The M weighted basis vectors are then summed up with the mean vector μ.

$$win_j = \mu + \sum_{i=1}^{M} f_i(\alpha, \beta) \cdot Basis_i \qquad (19)$$

The process in Equation 19 is repeated for all windows that are visible during training for the partition of interest. The final step requires us to simply draw the windows according to the method illustrated in section 2.1.

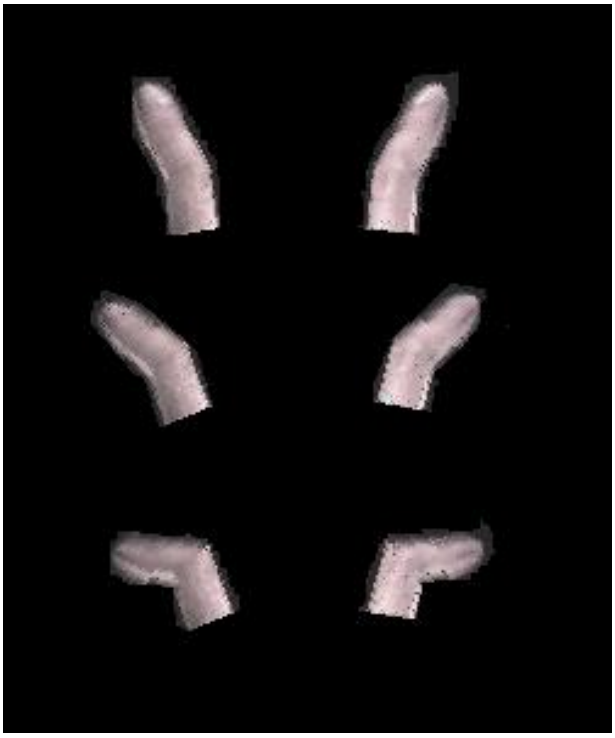

Figure 16 Appearance Model Drawn for Different States

Figure 16 shows us the appearance model drawn for three different states. Two distinct views of the object can be generated since the training was done with two views. These two views help in regularizing the tracking problem.

CHAPTER 3

Tracking Articulated Objects

In this section the tracking algorithm will be described. The various tracking components will be discussed including the appearance model and how it interacts with the other components. Important real-time considerations will also be addressed.

1. System View

The system view of the tracking algorithm can be seen in Figure 17. The algorithm starts by computing a predicted state ($\boldsymbol{X}d_{t+1}$) according to the system dynamics. This Predicted state is then refined by testing configurations near this predicted state thus obtaining an observation measure ($\boldsymbol{X}w_{t+1}$). These two quantities are then combined to form our new state estimate ($\boldsymbol{X}s_{t+1}$). Finally the dynamic states of the system are updated.

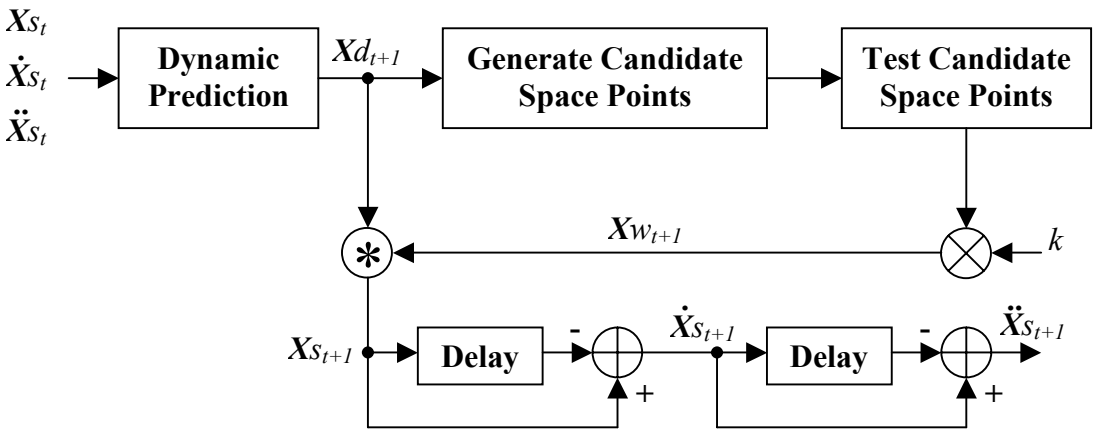

Figure 17 Diagram of System

The system variables are represented by gaussian random variables. This system can be qualified as a Kalman Filter since it contains a prediction model a measurement model and an update model.

1.1 Predictive Model Of Object State

The predictive model assumes constant acceleration between frames. Equation 20 shows the basic formulae for a constant acceleration model.

$$x(t+\Delta_t) = \tfrac{1}{2}\Delta_t{}^2 \cdot \frac{d^2}{dt^2}x(t) + \Delta_t \cdot \frac{d}{dt}x(t) + x(t) \qquad (20)$$

This Equation will be modified and formulated for gaussian random variables. Two rules pertaining to normal random variables are used. These rules are described below:

Rule 1:

If $Y = aX$, where X is normal with mean μ and standard deviation σ,

then Y will be normal with mean $a\mu$ and standard deviation $|a|\sigma$.

Rule 2:

If $S = \Sigma X_i$, where the X_i are normal with mean μ_i and standard deviation σ_i,

then S will be normal with mean $\Sigma\mu_i$ and standard deviation $(\Sigma\sigma_i^2)^{1/2}$.

Combining Equation 20 with the above rules we obtain a formulae for $\boldsymbol{X}d_{t+1}$ in Equation 21. Note that the state $\boldsymbol{X}s_t$ and its first and second derivatives are distinct normal random variables with their own mean and standard deviation,

$$\begin{aligned} \mu_{d\,t+1} &= \tfrac{1}{2}\Delta_t^2\,\ddot{\mu}_{s_t} + \Delta_t\,\dot{\mu}_{s_t} + \mu_{s_t}, \\ \sigma_{d\,t+1} &= \sqrt{\left(\left(\tfrac{1}{2}\Delta_t^2\right)^2 \ddot{\sigma}_{s_t}\right)^2 + \left((\Delta_t)^2\,\dot{\sigma}_{s_t}\right)^2 + \left(\sigma_{s_t}\right)^2}\,. \end{aligned} \qquad (21)$$

The equations describing the system variables will refer to the mean by μ and to the variance by σ. Equation 21 independently predicts future values for each dimension of the articulated object.

This prediction will prove to be useful in generating candidate samples as well as constraining the observation.

1.2 Hypothesis Validation Scheme

Once the predicted state of the model has been predicted, the input images (with the object at the new pose) must be tested. The search space can be restricted using information provided from the prediction.

1.2.1 Generating Candidate Space Point

The goal here is to chose hypothesis points in the α-β plane (*Link 1*). Since the solution will be close to the predicted state mean μ_{dt+1} for α and β, it makes sense to choose the candidate points nearby this prediction.

Of course the size of the region around this predicted mean will be dictated by the variance of the predicted state σ_{dt+1}. Ideally test points around the mean should be chosen with a density that dies off according to the variance. This involves solving the nonuniform quantization problem which is illustrated in Equation 22 for the one dimensional case. When choosing N samples the actual sample values would correspond to the x_i's .

$$D = \int_{-\infty}^{a_1} (x - x_1)^2 f_X(x)dx + \sum_{i=1}^{N-2} \int_{a_i}^{a_{i+1}} (x - x_{i+1})^2 f_X(x)dx + \int_{a_{N-1}}^{\infty} (x - x_N)^2 f_X(x)dx \quad (22)$$

However to simplify the matter we choose to uniformly sample inside a radius centered about the mean. This radius is a function of the variance. The smaller the variance the closer the samples will be to the mean.

1.2.2 Testing Candidate Space Point

Once the candidate points are chosen a confidence measure must be associated with each of them. This is achieved by using the appearance model described in Chapter 2. The idea is to generate the object appearance for each of the candidate samples and compare the generated image against the new observed image (one link at a time). The Sum of the Squared Differences (SSD) was chosen as a measure of similarity. The algorithm performs SSD on the RGB channels as well as all the available views (these views must correspond to the ones used in training).

The sample with the smallest associated SSD value would be the best choice. However it is also desirable to have a confidence associated with the measurement.

1.2.3 Measurement Confidence

Figure 18 shows us a possible response of our appearance model when it is compared to input images containing the object in some novel pose. In the case of *Link 1* (which has 2 d.o.f.) the response would be represented by a surface.

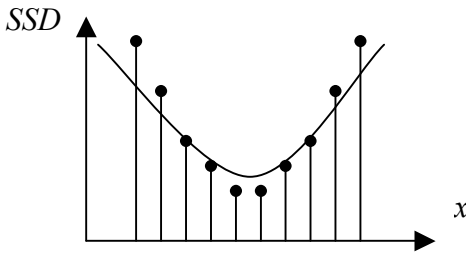

Figure 18 SSD Response

In [12] Matthies *et al.* choose to use the curvature of the SSD response as a measure of confidence. The sharpness of the SSD response provides a good measure of dissimilarity between the minimum and its neighbors.

The curvature can be extracted from the 1-D response of Figure 18 by fitting a second order polynomial to the data. The coefficients of the second order term are inversely proportional to the confidence on the minimum. In the case where the SSD response is a surface, a paraboloid can be fit to the data. Since the two d.o.f. of the link are independent the paraboloid can take the form of Equation 23. In this case *a* and *b* would be inversely proportional to the variance of α and β.

$$ssd = a\alpha^2 + b\beta^2 + c\alpha + d\beta + e \tag{23}$$

Assigning a confidence value to the observed state will prove to be very useful in the following section.

1.3 Merging Of Measurement and Prediction

The merging procedure for predicted state ($\boldsymbol{X}d_{t+1}$) and the measured state ($\boldsymbol{X}w_{t+1}$) is explained in Equation 24.

$$\mu_{s_{t+1}} = \frac{\mu_{d_{t+1}}}{\sigma_{d_{t+1}}} + \frac{\mu_{w_{t+1}}}{\sigma_{w_{t+1}} \cdot k} \Big/ \frac{1}{\sigma_{s_{t+1}}}$$
$$\sigma_{s_{t+1}} = 1 \Big/ \sqrt{\frac{1}{{\sigma_{d_{t+1}}}^2} + \frac{1}{\left(\sigma_{w_{t+1}} \cdot k\right)^2}} \qquad (24)$$

In Equation 24 the term k is called the Kalman Gain. The Kalman Gain weights the relative contribution of the new measurement to the prior expectation. This combination yields the new state $\boldsymbol{X}s_{t+1}$.

1.4 Update of System Dynamics

The Equations for the update of the state speed and acceleration are derived from the same fundamental rules stated in section 1.1. Equations 25 and 26 show the update formulas for the speed and the acceleration.

$$\dot{\mu}_{s_{t+1}} = \frac{\mu_{s_{t+1}} - \mu_{s_t}}{\Delta_t}, \quad \dot{\sigma}_{s_{t+1}} = \frac{\sqrt{\sigma_{s_{t+1}}^2 + \sigma_{s_t}^2}}{\Delta_t} \qquad (25)$$

$$\ddot{\mu}_{s_{t+1}} = \frac{\dot{\mu}_{s_{t+1}} - \dot{\mu}_{s_t}}{\Delta_t}, \quad \ddot{\sigma}_{s_{t+1}} = \frac{\sqrt{\dot{\sigma}_{s_{t+1}}^2 + \dot{\sigma}_{s_t}^2}}{\Delta_t} \qquad (26)$$

These quantities are used by the first stage of the system in Figure 17.

2. Real-Time Considerations

Real-time constraints are inherent to tracking problems. Some algorithms require many iterations to finally converge on the right solution. This is not desirable since little time should be spent on each frame.

Tracking already reduces the amount of computation required since it is assumed that the object will undergo small changes between frames. The solution for a given frame can be used to bootstrap the search in next frame.

Articulated objects have the added difficulty of a large configuration space. In the next section a method will be presented to reduce the complexity caused by this type of objects.

2.1 Recursive Nature of Tracking

The example tracked object has four d.o.f.. The first link has two d.o.f., links 2 and 3 have one d.o.f.. Since *Link 1* is independent from links 2 and 3, the angles defining the position of *Link 1* can be solved for ignoring the angles defining the position of *Link 2* and *Link 3*. In the same way *Link 2* is independent of *Link 3,* therefore the *Link 2* angle can be solved for while ignoring *Link 3*. This sums up the recursive structure of the algorithm. This type of structure makes the algorithm of order O(N) where N is the number of links.

2.2 Memory Management Scheme

The appearance model generated by the training phase requires a lot of memory when stored. At any given time the tracking algorithm needs to access only a small portion of the data. Consequently the whole appearance model never needs to be loaded in memory, PCA models can be loaded on demand. A caching strategy is used to minimize the overhead of disk accesses, using predictions to retain the appropriate models in memory. This approach greatly speeds up the tracking algorithm since the same PCA models are often reused in consecutive frames.

CHAPTER 4

Experimental Results

The performance of the non-rigid articulated object-tracking algorithm will now be evaluated. Tests were conducted using computer-generated images as well as real images captured with color cameras. Although the algorithm allows the use of multiple cameras, in the experiment two viewpoints proved to be sufficient.

In the following sections the experimental setup will be described as well as the tracking results.

1. Real Image Acquisition

Images of a moving finger are obtained using two ***3COM BigPicture*** color cameras. The OpenGl perspective projection was matched to the camera optics and also to the disposition of the cameras. The camera positions and calibration procedure will now be described.

1.1 Experimental Setup

Figure 19 shows a top view of the camera positions as well as the subject's hand position. The camera view vectors coincide with a straight angle, this is chosen as to maximize the independence of information provided by the two cameras. The subject's

hand rests on the horizontal bar. This immobilizes it and keeps the camera centered on the base of the index finger, identical to the training configuration in the OpenGL environment.

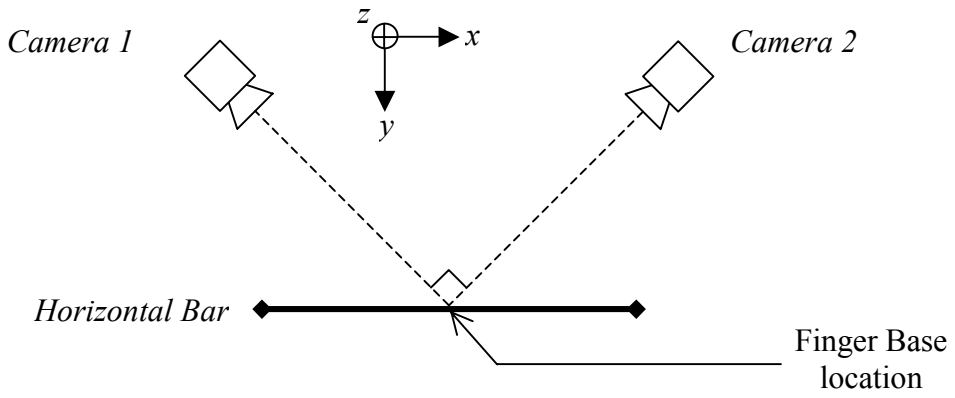

Figure 19 Experimental Setup

1.2 Camera Calibration

The calibration task involves finding the OpenGL perspective projection focal length that corresponds to the real camera focal length. The OpenGL units should correspond to the real world units (centimeters).

This was achieved by devising the following procedure. The picture of a square shape was used. The square had a 3.8 *cm* side and was viewed from a distance of 26.5 *cm*. Thus obtaining the picture in Figure 20. The width in pixels of the square was then determined. The square was found to have a width of 96 pixels.

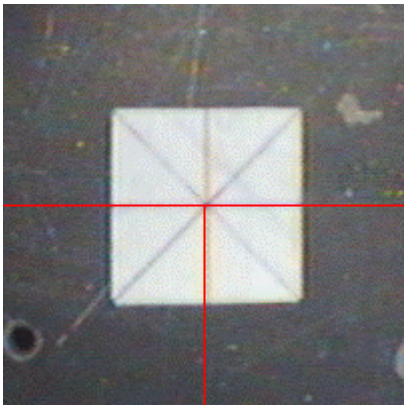

Figure 20 Calibration Picture

Finally a square shape was rendered in the OpenGL environment. The square had a width of 3.8 OpenGL units and was placed at a distance of 26.5 OpenGL units from the focal point. The perspective projection focal length in OpenGL was then adjusted until the OpenGL generated picture of the square shape had 96 pixels for its width.

This insures that dimensions in the OpenGL environment are the same as dimensions the real environment. This was done assuming the pinhole model for the cameras.

2. Tracking Results

In this section the results of our tracking algorithm will be presented. Tracking was performed on synthetic images as well as on real images. Tracking on synthetic images gives us great insight on the behavior of the algorithm. When generating synthetic images knowledge about the real state of the articulated object is available. However with real images there is no ground truth information readily available. Therefore the

synthetic sequences allow for comparison between the computed solution and the actual solution. For real sequences the algorithm performance estimation can be done visually.

2.1 Synthetic Images

As mentioned before, ground truth data is available for the synthetic images. In Figures 21 and 22 the real state, the state velocity and acceleration can be compared to their estimated counterparts. Since ground truth information is available the system state, speed and acceleration are initialized with the actual values.

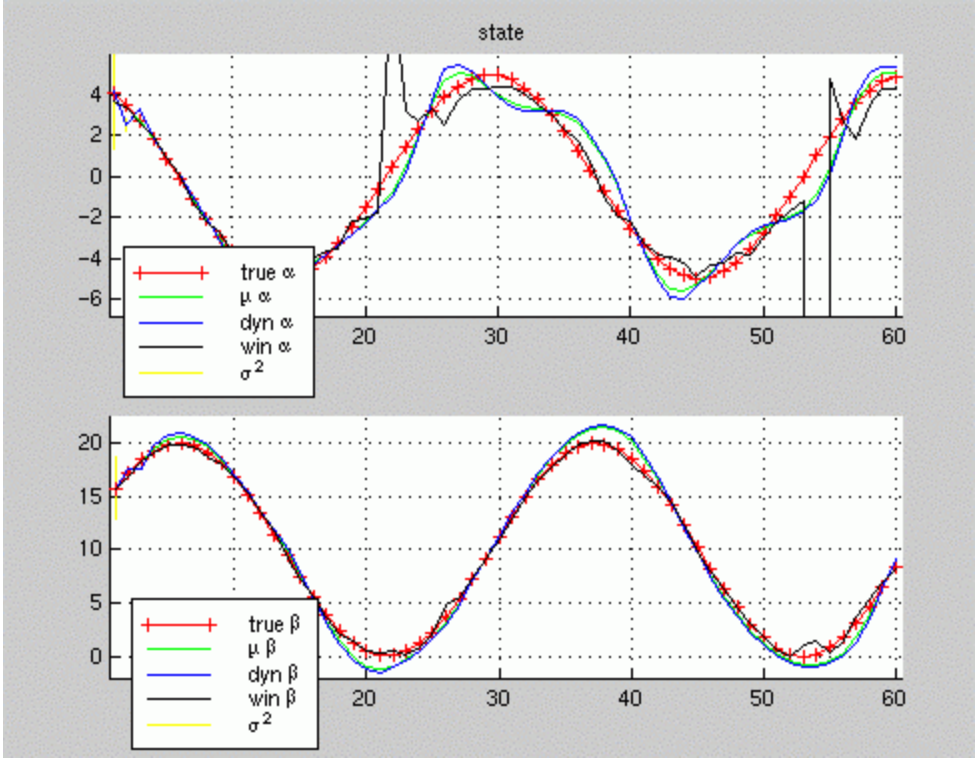

Figure 21 Object State

Figure 21 shows the response of the tracking system for sine shape inputs for the two degrees of freedom of the first link. Four quantities are plotted: the actual solution,

the constant acceleration prediction mean, the observed state mean and the final solution mean. Observe that these quantities follow each other reasonably well. The top response of Figure 21 (state α) has an untrue observation component at time steps 23 and 53. However this does not affect the tracking solution. The confidence measures insure that there will be more contribution from the predictive component and less from the observation at those times.

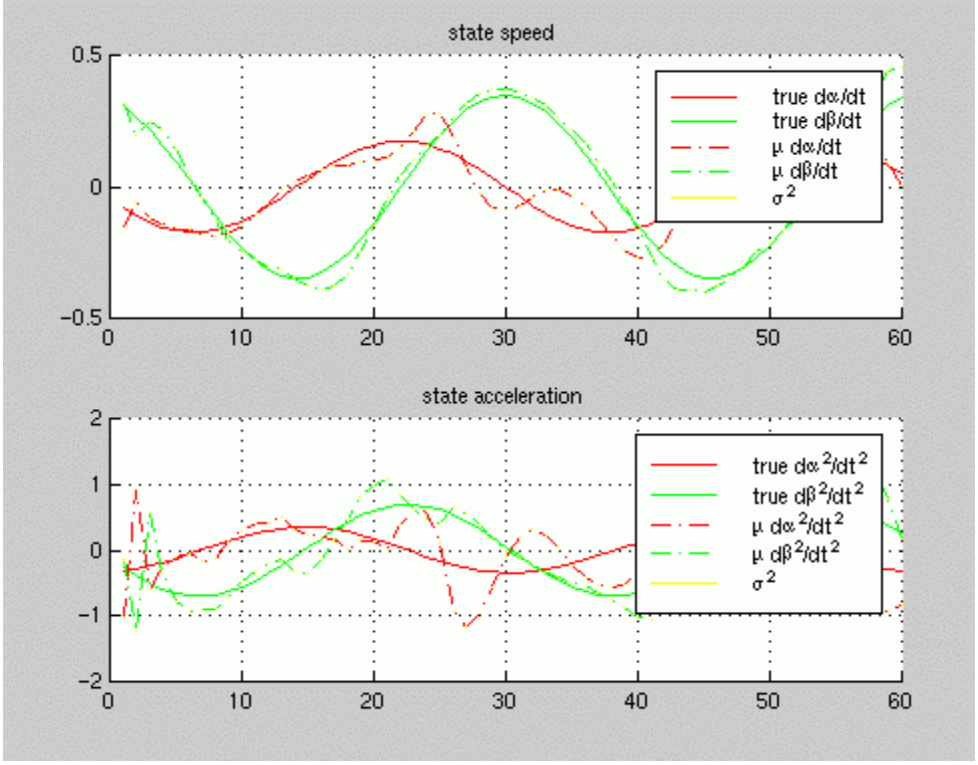

Figure 22 Object State Dynamics

Figure 22 reveals additional insight in the behavior of the algorithm. In the top and bottom plots the α state speed and acceleration are the waveform with smaller amplitude. The disturbances in Figure 22 are reflected in Figure 21, however the overall estimations reliably follow the real speed and acceleration.

2.2 Real Images

As mentioned before ground truth information is not available for the real image sequences. Visual verification is the only means available to evaluate the algorithm. To allow for visual verification a stick Figure representing the structure of the object was overlaid on the actual images. This overlaid shape is separated in three parts drawn with different shades, one representing each link.

The analysis of the first frame cannot be concluded as fast as for the subsequent frames; a large part of the state space has to be tested since no prior positional information exist. During this phase the users finger is assumed to be still until the algorithm is in lock.

Figures 23 to 32 show stereo views as well as estimated configuration of the object. Observe that the solution accurately depicts the pose of the finger. In the bottom pair of each image the appearance model that was obtained from the training is also overlaid on the real images. The appearance model is drawn for the estimated solution. Note that it represents well the actual appearance of the finger. The algorithm performs well even when the object is not well lit. Figures 25 to 30 illustrate this for *Link 3*. Because of the recursive structure of the algorithm the accuracy of the results for one link affect the accuracy of the subsequent link solutions. For example in Figure 31 the first link has some error which causes *Link 2* and especially *Link 3* to be misaligned with the actual structure of the finger.

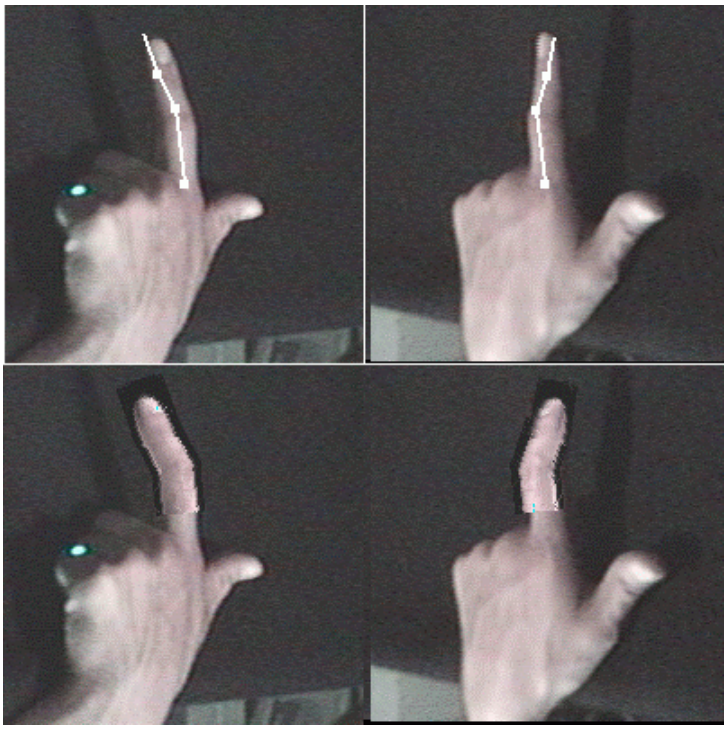

Figure 23 Frame 1

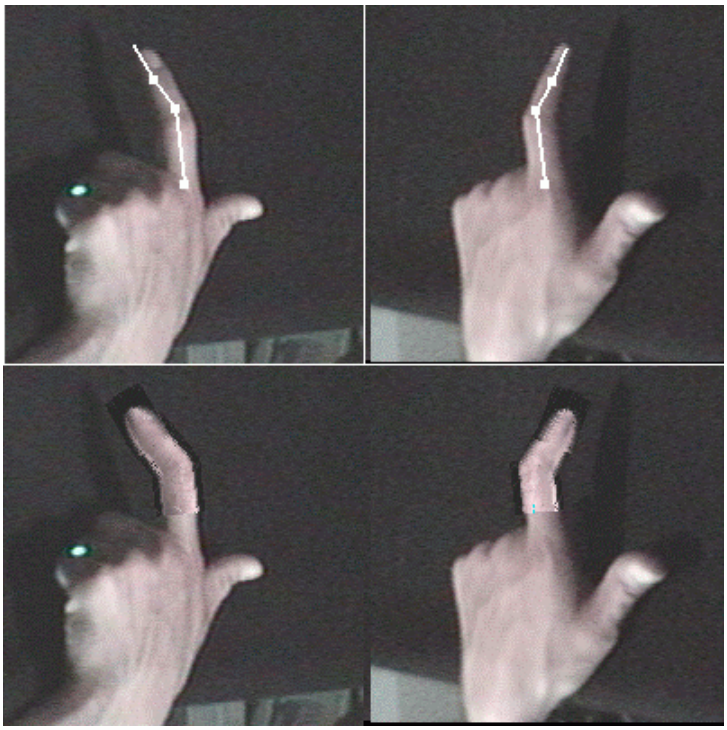

Figure 24 Frame 6

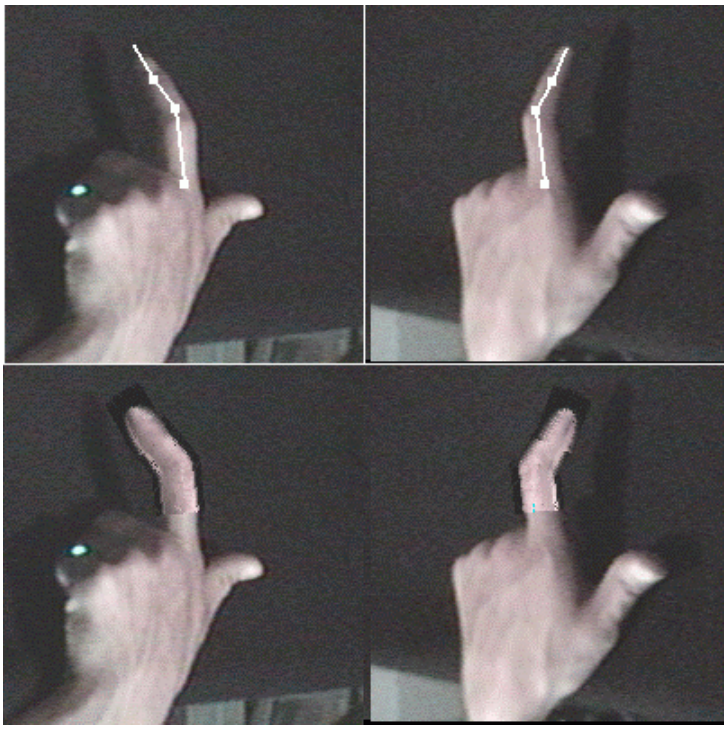

Figure 25 Frame 11

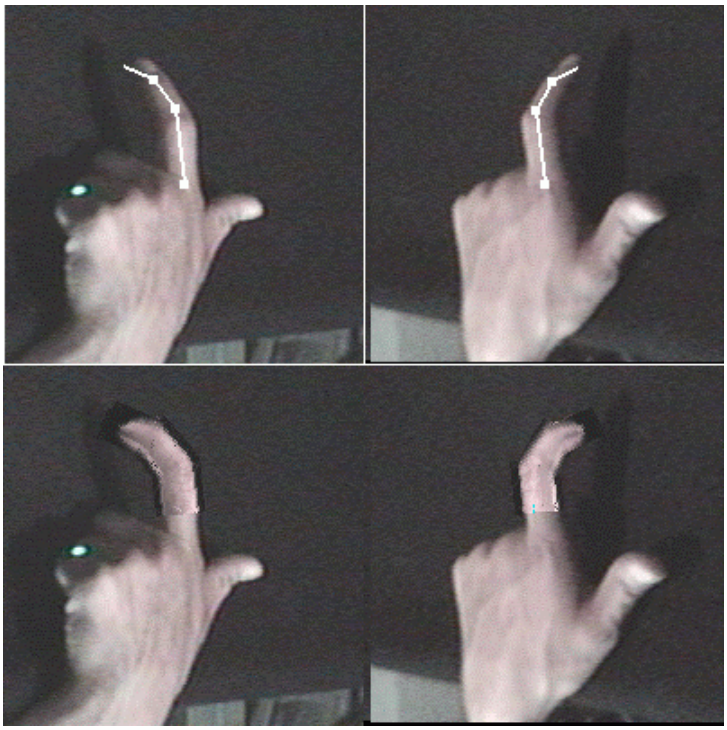

Figure 26 Frame 16

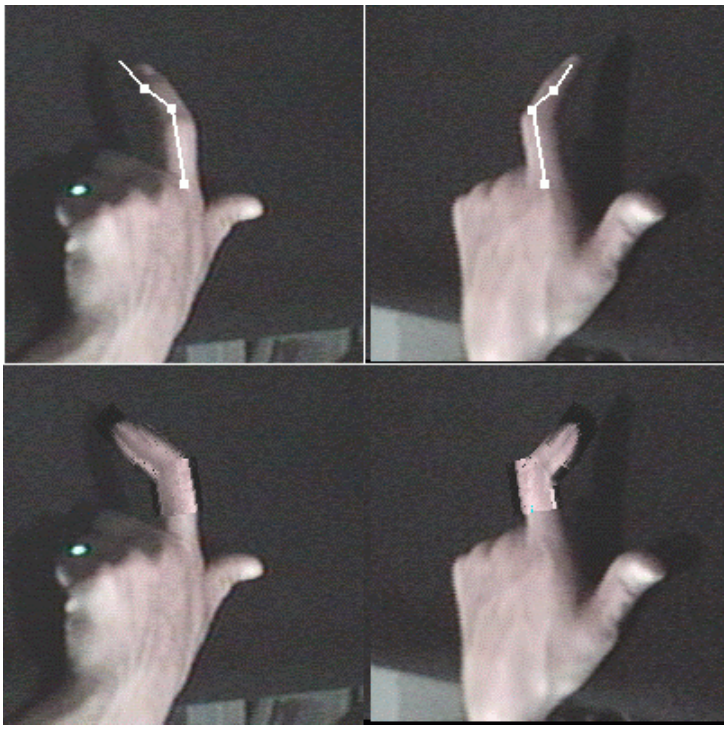

Figure 27 Frame 21

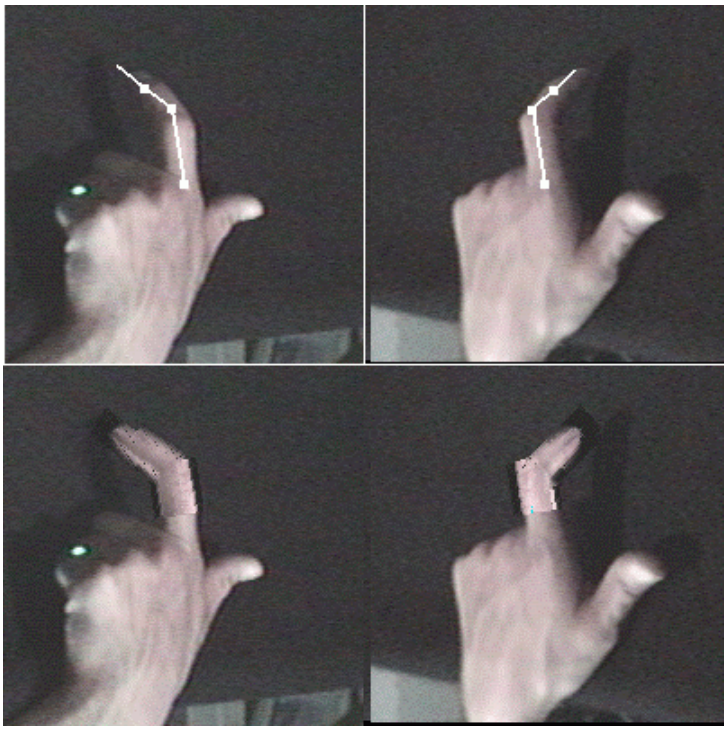

Figure 28 Frame 26

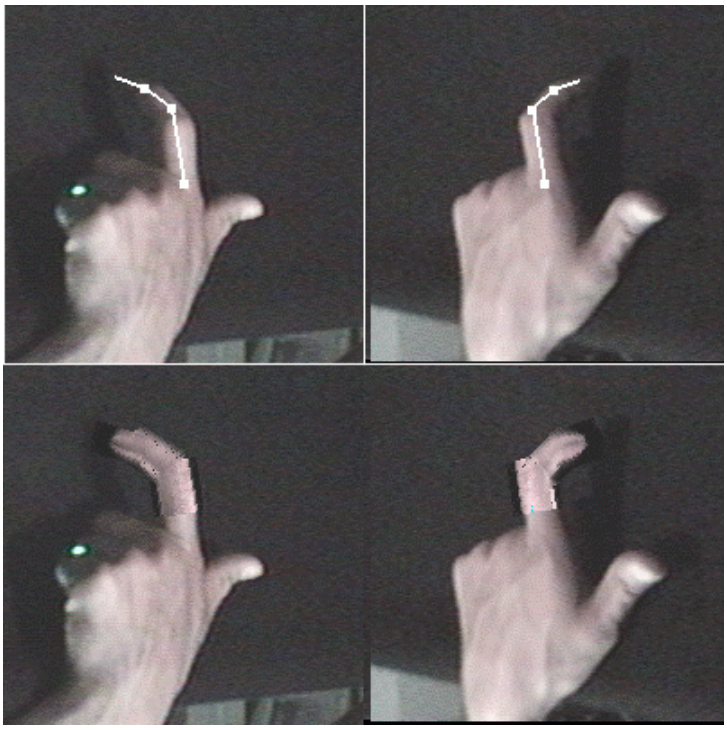

Figure 29 Frame 31

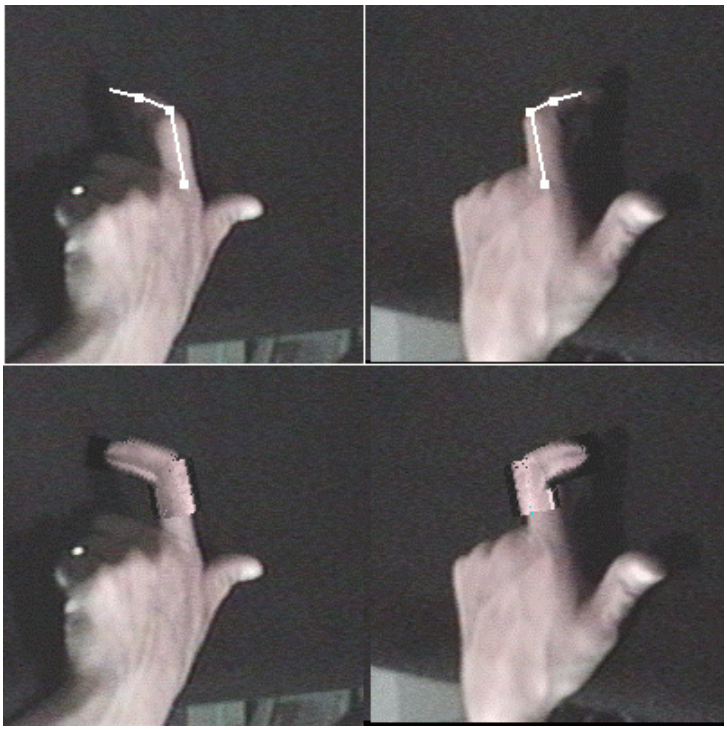

Figure 30 Frame 36

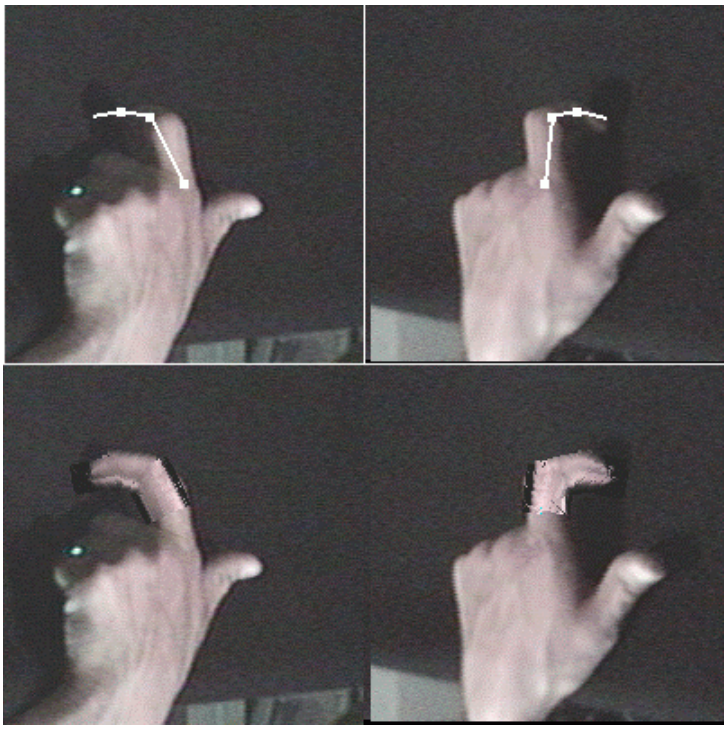

Figure 31 Frame 41

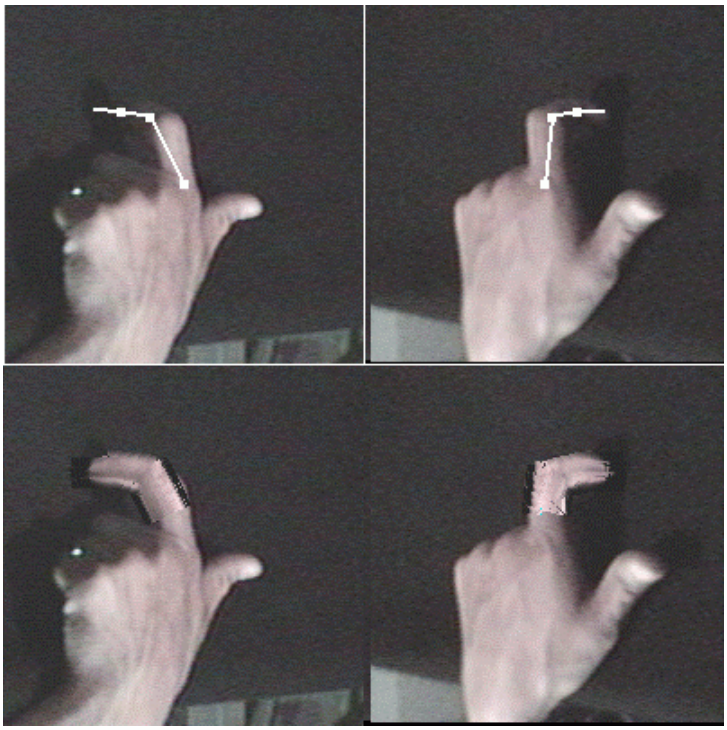

Figure 32 Frame 46

CHAPTER 5

Conclusions

A framework for tracking non-rigid articulated objects moving in a 3-D space has been presented. The algorithm uses a relatively precise model of the object's appearance to learn an appearance model. This appearance model does not account for the exact images from the training but it was shown that it reliably generates the overall appearance of the object. This appearance includes textures, deformations, lighting and perspective projection effects. The learnt appearance model is then used within the tracking system to test hypothesis states with the observed images. The tracking algorithm also used a predictive model to reduce the search space. To reduce the complexity associated with the dimensionality of the object, the tracking algorithm was designed with a recursive structure. As shown by the results this learnt model can be rendered efficiently enough to meet the real-time constraints imposed by the tracking. Measurement error will impact higher levels of processing in different ways. In the case of pose classification and dynamic gesture recognition these errors appear to be negligible. It was assumed that other objects would not occlude the articulated object since it was the only finger being tracked. However our tracking algorithm can handle self-occlusions that occur when one link hides another. This characteristic is again embedded in the generative model.

This thesis raises many important issues that require some attention. Some of these concern the overall tracking system; others are more specific to the observation model used in the tracking.

In the case of hand pose tracking the fingers can be tracked independently, it is therefore inevitable that some fingers occlude others. However even in this case the occlusions can be reliably predicted. Since the object's state is being tracked the approximate state of the object is always known. By using a simple object model occlusion regions can be determined for a given state. Using this information these occluded regions can be ignored by not testing the appearance model against them. The structure of our appearance model allows us to do this efficiently because it is separated in small parts (see Figure 13). The parts that are occluded can be ignored altogether saving time and excluding responses that are doomed in advance.

In our approach the initial model rendered by OpenGL was created manually. The shape and the texture are taken from the same subject as for the real sequences. Ideally each subject should have its own OpenGL model on which the training is performed. However this would require us to tune a model for each user. An automated OpenGL model construction would be extremely useful here. This model could have some characteristics that remain unchanged like the number of d.o.f. of each joint and the number of links. The model would also have some parameters that would be automatically tuned for each user like the length of the links and the width of the fingers. The parameters could be learned from images of the users hand at key poses. The OpenGL model texture could also be extracted during this phase. This would improve algorithm performance since the generative model would be tailored to the user's hand.

In the current setup the models accounting for the appearance (windows) are uniformly distributed so as to cover the finger in any given state. However when testing hypothesis solutions, some of these models might not account for any important features. It would be desirable to discard such windows. In fact during the training, window locations could be chosen automatically so as to cover as many features as possible. This would reduce the size of the appearance model since featureless locations are not represented. Furthermore the low-level mechanism used in our approach was PCA. This is well suited for representing the object's appearance. However while tracking it is more important to recognize the object then to represent actual appearance. Therefore it would be interesting to further compress the PCA representation to simply discriminate wrong hypothesis states without going through the whole reconstruction of the appearance. Following the same framework, it would be interesting to exchange the PCA model for some other type of learnt model that is better suited to detect finger edges and cusps.

There are many challenges in tracking systems. One of the important aspects of this thesis is that it accounts for non-rigid objects. Providing a solution from a continuous space also very important, as it allows for tracking arbitrary motions. The work presented here does not attempt to interpret the data. It is intended as a layer that extracts useful information that can be further interpreted according to the application's requirements.

REFERENCES

[1] Alberola C. Juan F. *Human Hand Postures and Gesture Recognition: Towards a Human-Gesture Communication Interface*. In IEEE Image Processing, 1999. Vol.4 pp. 222-226.

[2] Delamarre Q. Faugeras O. *Finding Pose of Hand in Video Images: a stereo-based approach.* In Proc. of IEEE Automatic Face and Gesture Recognition, 1998. pp. 585 -590.

[3] De la Torre, F. Black, M.J. *A Framework for Modeling the Appearance of 3D Articulated Figure*. In Proc. IEEE Automatic Face and Gesture Recognition, 2000. pp. 368–375.

[4] Deutscher J. and Blake A. *Articulated Body Motion Capture by Annealed Particle Filtering*. In Proc. of IEEE Computer Vision and Pattern Recognition, 2000. pp. 126-133.

[5] Foley D. et al. *Computer Graphics: principles and practice*, Addison-Wesley, 1991. pp. 517-522.

[6] Goncalves L. Di Bernardo E. *Reach Out and Touch Space (Motion Learning)*. In IEEE Automatic Face and Gesture Recognition, 1998. pp. 234-239.

[7] Hauck A. and Lanser S. *Hierarchical Recognition of Articulated Objects from Single Perspective Views*. In Proc. of IEEE Computer Vision and Pattern Recognition, 1997. pp. 870 –876.

[8] Heap T. Hogg D. *Towards 3D Hand Tracking using a Deformable Model*. In IEEE Automatic Face and Gesture Recognition, 1996. pp. 140-145.

[9] Imagawa K. Shan Lu *Color-Based Hand Tracking System for Sign Language Recognition*. In IEEE Automatic Face and Gesture Recognition, 1998. pp. 462-467.

[10] Kwon K. and Zhang H. *Hand Pose Recovery with a Single Video Camera*. In Proc. of IEEE International Conference on Robotics and Automation, 2001. Vol. 2, pp. 1194 –1200.

[11] Lathuiliere F. Hervé J.-Y. *Visual Tracking of Hand Posture with Occlusion Handling.* In IEEE Pattern Recognition, 2000. Vol. 3 pp. 1129-1133.

[12] Matthies, L., Kanade, T. and Szeliski, R., *Kalman Filter-based Algorithms for Estimating Depth from Image Sequences*, International Journal of Computer Vision, 1989. pp. 209-236.

[13] Min B.-W. Yoon H.-S. *Hand Gesture Recognition Using Hidden Markov Models*. In IEEE Computational Cybernetics and Simulation, 1997. Vol. 5 pp. 4232 -4235.

[14] Sato Y. Kobayashi, Y. *Fast Tracking of Hands and Fingertips in Infrared Images for Augmented Desk Interface*. In IEEE Automatic Face and Gesture Recognition, 2000. pp. 462-467

[15] Wu A. Shah M. *A Virtual 3D Blackboard: 3D Finger Tracking using a Single Camera*. In Proc. Automatic Face and Gesture Recognition, 2000. pp. 536–543.

[16] Ying Wu Huang, T.S. *Capturing Articulated Human Hand Motion: A Divide-and-Conquer Approach*. In IEEE Computer Vision, 1999. Vol. 1 pp. 606-611.