RECONSTRUCTION OF 3D RIGID BODY MOTION IN A VIRTUAL ENVIRONMENT FROM A 2D IMAGE SEQUENCE

A Thesis

by

SUMANTRA DASGUPTA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2003

Major Subject: Electrical Engineering

RECONSTRUCTION OF 3D RIGID BODY MOTION IN A VIRTUAL ENVIRONMENT FROM A 2D IMAGE SEQUENCE

A Thesis

by

SUMANTRA DASGUPTA

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

May 2003

Major Subject: Electrical Engineering

ABSTRACT

Reconstruction of 3D Rigid Body Motion in a Virtual Environment

from a 2D Image Sequence. (May 2003)

Sumantra Dasgupta, B.E., BIT MESRA, Ranchi, India.

Chair of Advisory Committee: Dr. Aniruddha Datta

This research presents a procedure for interactive segmentation and automatic tracking of moving objects in a video sequence. The user outlines the region of interest (ROI) in the initial frame; the procedure builds a refined mask of the dominant object within the ROI. The refined mask is used to model a spline template of the object to be tracked. The tracking algorithm then employs a motion model to track the template through a sequence of frames and gathers the 3D affine motion parameters of the object from each frame. The extracted template is compared with a previously stored library of 3D shapes to determine the closest 3D object. If the extracted template is completely new, it is used to model a new 3D object which is added to the library. To recreate the motion, the motion parameters are applied to the 3D object in a virtual environment. The procedure described here can be applied to industrial problems such as traffic management and material flow congestion analysis.

ACKNOWLEDGEMENTS

The material is based upon work supported by the Texas Advanced Research (Technology) Program under Grant No. 00512-0387-1999. I would like to acknowledge the continued support of Dr. Amarnath Banerjee and Dr. Aniruddha Datta. I would also like to acknowledge the assistance of Mr. Karthik Chander in developing the shape library and VRML code.

TABLE OF CONTENTS

Page

ABSTRACT……………………………………………………………………….. iii

ACKNOWLEDGEMENTS…………………………………………………….... iv

TABLE OF CONTENTS………………………………………………………… v

LIST OF FIGURES……………………………………………………………...vii

CHAPTER

I. INTRODUCTION……………………………………………………………….1

II. VIDEO SEGMENTATION AND TRACKING…………………………….....5

2.1 MANUALSEGMENTATION……………………………………………….5
2.2 AUTOMATIC REFINEMENT………………………………………………6
2.2.1 MODE 1:FOR A COARSE USER DEFINED MASK………………………..6
2.2.2 MODE 2:MORPHOLOGICAL WATERSHED ALGORITHM……………...6
2.3 SHAPE SPACE MODELING………………………………………………10
2.3.1 SPLINE CURVES………………………………………………………10
2.3.2 SHAPE SPACE…………………………………………………………11
2.3.3 CAMERA PERSPECTIVE AND AFFINE SPACE…………………………12
2.3.4 KEY-FRAMES…………………………………………………………15
2.4 TRACKING……………………………………………………………....15
2.4.1 FITTING SPLINE TEMPLATES…………………………………………15
2.4.2 PROBABILISTIC MODELS OF SHAPE………………………………....17
2.4.3 MODELS OF MOTION…………………………………………………..18
2.4.4 CONSTRAINED AND UNCONSTRAINED MOTION……………………20
2.4.5 BLOCK MATCHING………………………………………………… 20
2.4.6 CONTOUR TRACKING…………………………………………………25

III. TEMPLATE MATCHING……………………………………………………..27

3.1 SITUATIONS ENCOUNTERED DURING VIDEO CAPTURE………………27
3.2 TEMPLATE MATCHING PROCESS……………………………………..30

CHAPTER Page

IV. RESULTS AND CONCLUSION………………………………………………...34

4.1 EXPERIMENTAL SETUP…..…………………………………………………..34
4.2 DISCUSSIONS AND FUTURE WORK………………………………………..38

REFERENCES……………………………………………………………………..40

VITA………………………………………………………………………………..43

LIST OF FIGURES

Page

Figure 1. A video frame and region of interest (ROI) selection…………………. 5

Figure 2. Relationship between B, B_{in}, B_{out} and B_{init}……………………………. 7

Figure 3. Object boundary, cluster centers and distance………………………… 8

Figure 4: Object selection by user……………………………………………….. 9

Figure 5: Cleaning of mask by watershed algorithm……………………………. 9

Figure 6. 3D imaging geometry…………………………………………………. 13

Figure 7. Object mask and the search area……………………………………… 21

Figure 8. Reduction in search space from frame I_{k-1} to I_k………………………. 22

Figure 9. Steps in the block matching algorithm……………………………….. 24

Figure 10. Global rotation due to point of view of camera……………………... 27

Figure 11. Effects of roll, pitch and yaw……………………………………….. 29

Figure 12. Rotation of object by 180 degrees to obtain left face………………. 31

Figure 13. Two shapes from the template library: (a) car and (b) minivan……. 32

Figure 14. An unknown shape, which matches with the car in the template library……………………………………………………………. 33

Figure 15. Schematic diagram of experimental setup…………………………. 34

Figure 16. Motion information and masks from a left turn sequence…………. 37

CHAPTER I

INTRODUCTION

Separation of moving objects from a static background in a video sequence is a well known segmentation and tracking problem. In recent years, there has been growing interest in this problem due to such diverse applications like object-based video compression (e.g. MPEG4/7 specifications), video surveillance and monitoring, analysis of medical images and automatic target detection and tracking [1].

The previous work on segmentation can be categorized into three broad categories: 1) intra-frame segmentation and inter-frame tracking, 2) dense motion clustering or segmentation, and 3) frame differencing. In the first approach, frames are segmented independently using traditional segmentation techniques (i.e. exploiting the intensity and texture of images) and then the segmentations are compared in a frame-by-frame basis [2]. This method works well with a small number of regions. Segmentation using intensity and texture is difficult and often leads to over-segmentation. To remedy the situation, 3D segmentation of images have also been attempted but it is computationally intensive [3]. In the second approach, a dense motion field is used for segmentation. Pixels with similar motion vectors are grouped together ([4], [5]). This method is suitable for situations where motion estimation produces reliable motion fields, which is rare in case of real-world image sequences. In the third approach, frame differencing is used to mark out moving pixels [6]. The pixel-by-pixel difference between two frames is taken.

The journal model is *IEEE Transactions on Automatic Control*.

If a particular pixel position has a large frame difference then it is related to a moving object. This process has the advantage of being computationally efficient, but does not produce whole objects. Moreover, it is very sensitive to noise and is inapplicable for cases where global motion exists.

If object tracking is the ultimate goal of image sequence segmentation, which is true in our case, then semi-automated techniques can be both reliable and fast. In semiautomatic segmentation, the user provides manual segmentation in the first few frames. The segmented portion is tracked automatically by the procedure in the subsequent frames. The elegance of the method lies in the way it reduces the semantic overload of object recognition on the part of the procedure. The concept becomes clear if we try to differentiate between a region and an object. The concept of a region is supposed to denote an area in a frame characterized by its homogeneity of one or more quantitative features, such as color, texture, gray level, motion, etc. On the other hand, an object is defined as in the framework of MPEG4 as "an entity in a scene that a user is allowed to access (seek, browse), and manipulate (cut and paste)" [7]. Unlike regions, objects have more semantic content, but lack the global coherence in color, texture and movement. Computers can be easily programmed to separate out regions, but not objects because of this semantic factor. It is here that an user can provide suggestion to the computer running the region separation algorithm, (which is relatively easy on his/her part) in the form of initial segmentation of whole objects. Once this is done, the region separation procedure can treat the manually segmented object as a set of regions, grouped together by the user, and identify the object in the subsequent frames. In fact, the procedure can improve the initial segmentation by various automated techniques and then use the improved segmentation to track similar group of regions in subsequent frames, as is shown in this paper. Some of the existing approaches can be found in the literature ([8], [9], [10], [11], [12]).

There are existing methods for 3D object reconstruction from 2D data using stereo vision and telemetry-based depth recovery ([13], [14], [15]). These methods help in reconstructing static 3D objects from 2D images. The proposed method looks at perspective 2D video data from a single fixed camera, identifies the contour representing the object, which is used for reconstructing the object in 3D. In addition, the method tracks the object to obtain its motion information in time, which is used in a virtual environment to play back the sequence of events. The concept of recreating motion in virtual reality is not new. It was conceived in [16], where the concept of virtualized reality is introduced. Virtualized Reality is said to be superior to virtual reality in that it knows the depth information of every real world pixel. Construction of virtual world from real scenes is further explored in [17]. Modelling of dynamic scenes is dealt in [18]. Recent works in this field include [19] [20] [21]. [19] [21] discuss moving object extraction using pre-stored background information and [20] deals with motion extraction using non-parametric probabilistic model (for local motion).

The proposed method consists of two main steps. First, the object and motion information is collected from the given video sequence. Appropriate probabilistic methods and motion models are described to deal with the whole data acquisition process[22]. Then, this information is mapped to produce corresponding 3D objects in a virtual environment. These 3D objects are used in the reconstruction of object behavior in the virtual environment. The next chapter describes the video sequence segmentation, while the third chapter describes the mapping process of the segmentation information to 3D objects in the virtual environment.

CHAPTER II

VIDEO SEGMENTATION AND TRACKING

The video sequence segmentation algorithm is divided into four components. Section 2.1 describes the initial user level segmentation, while Section 2.2 describes the automated refinements of the initial segmentation. Section 2.3 discusses models of objects, suitable for implementation in a computer. It also discusses issues related to camera perspective and calibration. Finally, Section 2.4 describes the automatic tracking of the object of interest in subsequent frames.

2.1 MANUAL SEGMENTATION

The user can interactively select the region of interest (ROI) in the first frame. Upon completion, the procedure stores a cropped portion of the ROI and its binary mask. This is shown in Figure 1. These form the basis for all subsequent automated segmentation and tracking. If there are multiple moving objects and the user is interested in separating out an object moving at a specific rate, then that optional information can be provided along with the initial segmentation. Velocity tuned filters (VTF) [23] can be used in such circumstances.

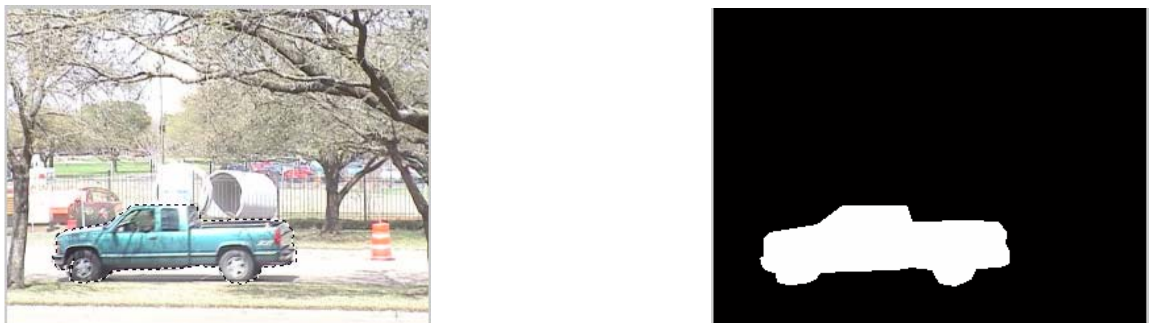


Figure 1. A video frame and region of interest (ROI) selection.

2.2 AUTOMATIC REFINEMENT

There are two modes of refining the initial segmentation depending on the type of mask that is needed.

2.2.1 MODE 1: FOR A COARSE USER DEFINED MASK

The automatic refinement of the initial mask is broken down into a series of steps: (i) edge detection within ROI using *Sobel/zero-crossing techniques/Canny* ([1], [24]), (ii) blackening out spurious edges using extended *Laplacian filter/color differencing*, (iii) predicting missing edge points using simple slope predictor, (iv) scanning for object edges from outside - left, right, top, bottom, and building a refined mask by blackening out everything outside edge points and whitening everything within, (v) apply *erosion and dilation* [1] to angular edges, and (vi) give the user an option to choose the final mask that will be used for subsequent automated tracking. This method is suitable for objects with straight edges.

2.2.2 MODE 2:MORPHOLOGICAL WATERSHED ALGORITHM

The algorithm can be broken down into a series of steps:

1. Mark the region of uncertainty: Erosion and dilation operators are applied to the user mask B_{init} to obtain B_{in} and B_{out}

$$B_{in} = \varepsilon\ (B_{init}) \qquad (1)$$

$$B_{out} = \delta\ (B_{init}) \qquad (2)$$

where ε and δ are the morphological erosion and dilation operators respectively. The following inequality is satisfied.

$$B_{in} \subseteq B_{init} \subseteq B_{out} \tag{3}$$

Also,

$$B_{in} \subseteq B \subseteq B_{out} \tag{4}$$

where B is the real video object. The region of uncertainty is the region between B_{in} and B_{out}. The pixels in this region are to be labeled inside or outside the object. Figure 2 shows the relationship between B, B_{init}, B_{in} and B_{out}.

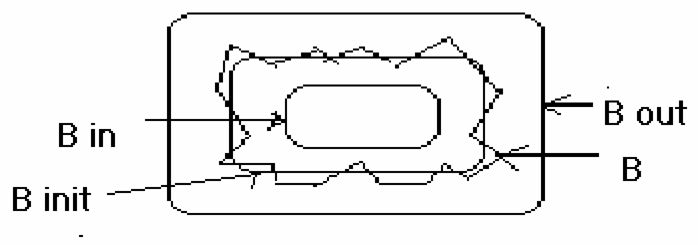


Figure 2. Relationship between B, B_{in}, B_{out} and B_{init}.

2. Create the cluster-centers: The cluster-centers are created along the contours of B_{in} and B_{out}. Both the RGB components and the pixel positions are considered. So, each cluster-center is a five dimensional vector with three color components and two position components. In this case, there are five consecutive edge points to form the cluster-centers. Each cluster-center is either 'in' or 'out' cluster-center, as shown in Figure 3.

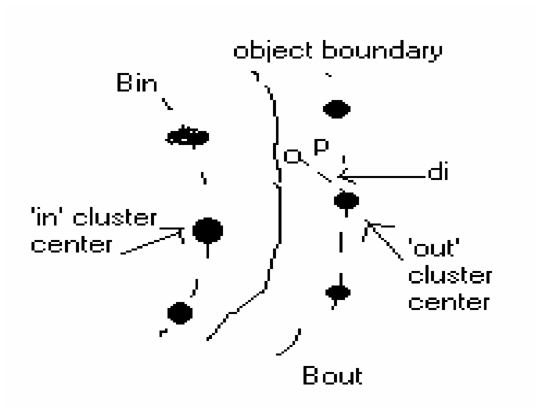


Figure 3. Object boundary, cluster centers and distance.

3. Decide initial markers for the uncertain points: The concept of distance can be defined as follows. Suppose a cluster-center C is characterized by $\{r,g,b,x,y\}$, where the first three parameters specify the RGB color components and the last two parameters specify the pixel coordinates The distance of a pixel P, characterized by $\{r_i,g_i,b_i,x_i,y_i\}$, from C is given by,

$$d_i = | r-r_i | + | g-g_i | + | b-b_i |, \; P \in N(C) \tag{5}$$

where N(C) is a neighborhood of C. In our case, all N(C)'s are subsets of B_{out}-B_{in}.

Each cluster-center has a hierarchical queue associated with it. The pixels belonging to the N(C) of the cluster-center are arranged in the queue in the order of increasing distance. Since neighborhoods of cluster-centers overlap, there may be multiple occurrences of the same pixel in different queues.

4. Flooding: The multiple occurrence of pixels is resolved and the uncertain pixels are given their final in/out labels. For each cluster-center, the pixels are chosen in the order of increasing distance. Once a pixel is grouped to a cluster-center, the cluster-center is updated and any other occurrence of the pixel in other queues is erased.

Practical considerations: Figure 5 illustrates the effect of watershed as applied to the user selection (Figure 4). Shadow beneath the vehicle wheels causes minor distortions in and around the wheels. These distortions can be removed by reapplying the watershed algorithm with smaller erosion and dilation masks.

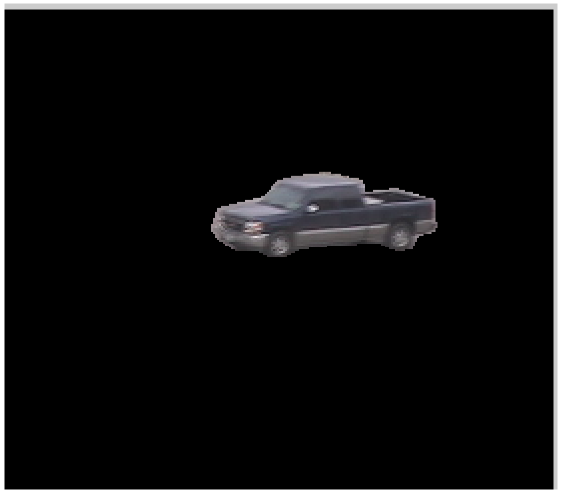


Figure 4. Object selection by user.

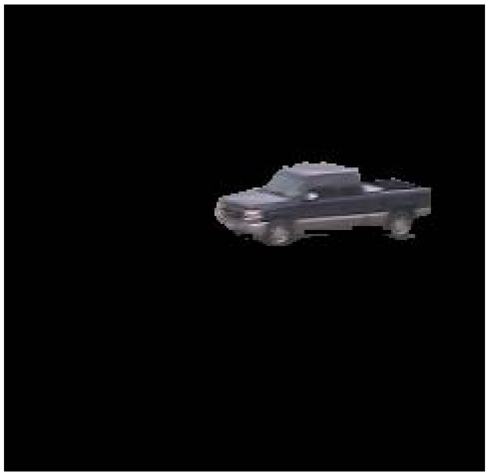


Figure 5. Cleaning of mask by watershed algorithm.

2.3 SHAPE SPACE MODELING

2.3.1 SPLINE CURVES

Visual curves can be represented in terms of parametric spline curves. In a spline curve, the coordinates of the curve (x(s) ,y(s)) are traversed as the parameter s increases. The functions x(s) and y(s) are known as splines. A spline of order d is a piecewise polynomial function, with concatenated segments of order d, joined together at breakpoints.

B-splines consist of curve segments whose polynomial coefficients depend on just a few control points. In the B-spline form, the spline function x(s) is made as a weighed sum of N_B basis functions $B_n(s)$, n=0,1,…., N_B-1. A parametric B-spline curve, in the interval [0,L] is made of a pair of such spline function and can be represented as,

$$\mathbf{r}(s)=(x(s),y(s)) \tag{6}$$

$$\mathbf{r}(s) = \sum_{n=0}^{N_B-1} B_n(s)\mathbf{q}_n \qquad \text{for } 0 \le s \le L \tag{7}$$

where $\mathbf{q}_n$ is a control point (q_n^x, q_n^y). In a more compact form, it can be written as,

$$\mathbf{r}(s) = U(s)\mathbf{Q} \tag{8}$$

where,

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_x \\ \mathbf{Q}_y \end{pmatrix} \tag{9}$$

and

$$U(\mathrm{s}) = \begin{pmatrix} \mathbf{B}(\mathrm{s})' & 0 \\ 0 & \mathbf{B}(\mathrm{s})' \end{pmatrix} \tag{10}$$

with $\mathbf{Q}_x$ and $\mathbf{Q}_y$ being the vectors of x and y coordinates of the control points and **B**(s) is the vector of B-spline weights.

We can define a norm ||.|| for a B-spline curve which is induced by the Euclidean distance measure in the image plane,

$$||\mathbf{Q}||^2 = \mathbf{Q}^T\ \mathbf{U}\mathbf{Q} \tag{11}$$

where,

$$\mathbf{U} = 1/\mathrm{L}\int_0^{\mathrm{L}} U(\mathrm{s})' U(\mathrm{s})\, ds \tag{12}$$

The centroid of a curve is defined

$$\bar{\mathbf{r}} = 1/\mathrm{L}\int_0^{\mathrm{L}} \mathbf{r}(\mathrm{s})\, ds \tag{13}$$

The area of a curve is defined as

$$\mathrm{A} = \int_0^{\mathrm{L}} |\, \mathbf{r}(\mathrm{s}), \mathbf{r}'(\mathrm{s})\, |\, ds \tag{14}$$

2.3.2 SHAPE SPACE

A shape-space S=L(W,$\mathbf{Q}_0$) is a linear mapping of a shape-space vector **X** to a spline vector **Q.** The dimension N_X of the shape space vector is always less than N_Q, the dimension of the spline vector.

$$\mathbf{Q} = \mathrm{W}\mathbf{X} + \mathbf{Q}_0 \tag{15}$$

where W is a N_Q x N_X shape matrix and $\mathbf{Q}_0$ is a template curve in spline space.

The norm over *S* is

$$||\mathbf{X}||^2=\mathbf{X}^T \mathbf{HX} \quad (16)$$

where,

$$\mathbf{H}=W^T \mathbf{U}W \quad (17)$$

The norm over *S* has a geometric interpretation,

$$||\mathbf{X}||^2=||\mathbf{Q}-\mathbf{Q}_0|| \quad (18)$$

is the average displacement of the curve parameterized by **X** from the template curve. The inverse mapping from spline space to shape space can be defined as,

$$\mathbf{X}=W^+ (\mathbf{Q}-\mathbf{Q}_0) \quad (19)$$

where $W^+ =\mathbf{H}^{-1} W^T \mathbf{U}$ is the pseudo-inverse of W.

2.3.3 CAMERA PERSPECTIVE AND AFFINE SPACE

The modeling of the object is based on the perspective projection model. The imaging geometry of a perspective camera is shown in Figure 6. The origin of the 3-D coordinate system (X, Y, Z) lies at the optical center (C) of the camera lens. The retinal plane or the image plane is normal to the optical axis Z and is offset from C by the focal length *f*. Images of unoccluded 3-D objects in front of the camera are formed on the image plane. The 2-D image plane coordinate system is centered at the principal point, which is the interSection of the optical axis with the image plane.

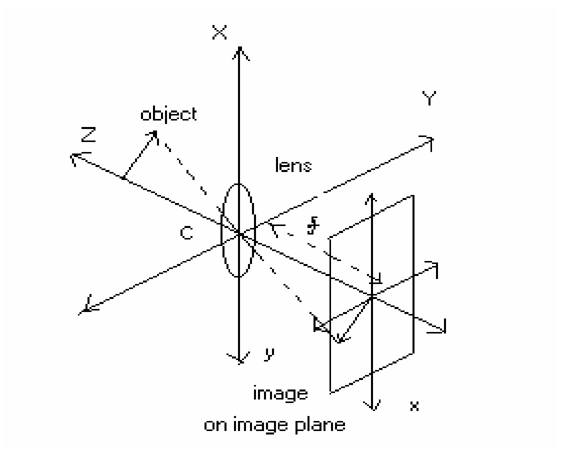


Figure 6. 3D imaging geometry.

The orientation of (x, y) is flipped with respect to (X, Y) in the Figure, following the principles of transmissive optics. The two coordinate systems are related by the perspective equations:

$$\mathbf{r}(s) = f/Z(s)\begin{pmatrix} X(s) \\ Y(s) \end{pmatrix} \tag{20}$$

where $\mathbf{r}(s)$ is the curve on the image plane and $(X(s),Y(s),Z(s))^T$ is the 3D object. The focal length f of a camera can be calculated using the formula,

$$f = f_{nom}\,(1-f_{nom}/Z_c\,)^{-1} \tag{21}$$

where f_{nom} is the nominal focal length usually printed on the side of the lens housing and Z_c is the working distance.

Let us consider an object whose center is at $\mathbf{R}_C = (X_C, Y_C, Z_C)$ and whose span is described by the curve $\mathbf{R}(s)$. If we assume that the object diameter is much less than the working distance (assumption for orthographic projection) we may write the perspective equation as,

$$\mathbf{r}(s) = f/Z_C(s)\begin{pmatrix} X_C + X(s) \\ Y_C + Y(s) \end{pmatrix} \tag{22}$$

If we suppose that the object contour $\mathbf{R}_C$+ $\mathbf{R}$(s) derives from $\mathbf{R}_0$(s) using a rotation by *R* and a translation by $\mathbf{R}_C$, we get

$$\mathbf{r}(s) = \mathbf{u} + f/Z_C(s) R_{2x2}\begin{pmatrix} X_0(s) \\ Y_0(s) \end{pmatrix} \tag{23}$$

where $\mathbf{u}$ is the orthographic projection of the vector $\mathbf{R}_C$ and R_{2x2} is the upper-left 2x2 block of the rotation matrix *R.* The previous equation is analogous to a 2D affine space. Non-planar objects are better approximated by the 3D affine space model. Consider the object to be a 3D curve

$$\mathbf{R}_0(s)=(X_0(s),Y_0(s),Z_0(s))^T \tag{24}$$

This object when projected orthographically gives the image curve,

$$\mathbf{r}(s) = \mathbf{u} + f/Z_C(s) R_{2x3}\begin{pmatrix} X_0(s) \\ Y_0(s) \\ Z_0(s) \end{pmatrix} \tag{25}$$

This can be expressed as a 8-parameter affine transformation,

$$\mathbf{r}(s)=\mathbf{u}+M\,\mathbf{r}_0(s)+\mathbf{v}Z_0(s) \tag{26}$$

with

$$R_{2x3} = Z_C / f\,(M \mid \mathbf{v}) \tag{27}$$

In shape space, this is interpreted as,

$$W = \begin{pmatrix} 1 & 0 & \mathbf{Q_{0x}} & 0 & 0 & \mathbf{Q_{0y}} & \mathbf{Q_{0z}} & 0 \\ 0 & 1 & 0 & \mathbf{Q_{0y}} & \mathbf{Q_{0x}} & 0 & 0 & \mathbf{Q_{0z}} \end{pmatrix} \tag{28}$$

and

$$\mathbf{X}=(u_1, u_2, M_{11}-1, M_{22}-1, M_{21}, M_{12}, v_1, v_2) \tag{29}$$

2.3.4 KEY-FRAMES

In case of multiple template frames, one can use the concept of key-frames in shape space. Suppose there are three template frames $\mathbf{r}_0$,$\mathbf{r}_1$ and $\mathbf{r}_2$ (e.g. the front view, side view and back-view of a car). Any intermediate view, composed of rotation, translation and zooming of the three templates can be modeled in shape space using the shape space matrix,

$$W = \begin{pmatrix} 1 & 0 & \mathbf{Q}_{0x} & -\mathbf{Q}_{0y} & \mathbf{Q}_{1x} & -\mathbf{Q}_{1y} & \mathbf{Q}_{2x} & -\mathbf{Q}_{2y} \\ 0 & 1 & \mathbf{Q}_{0y} & \mathbf{Q}_{0x} & \mathbf{Q}_{1y} & \mathbf{Q}_{1x} & \mathbf{Q}_{2y} & \mathbf{Q}_{2x} \end{pmatrix} \tag{30}$$

2.4 TRACKING

The tracking process is used to automatically track the user defined template through a sequence of frames. The Section begins with the description of curve matching algorithms. Subsequently probabilistic models of shape are described. The Section then goes on to describe some feasible models for capturing the dynamics of motion. Finally, the curve matching algorithm is blended with a proper dynamic model to invoke dynamic contour tracking.

2.4.1 FITTING SPLINE TEMPLATES

Suppose we are given an initial shape estimate $\mathbf{r}_m(s)$(or $\mathbf{X}_m$ in shape space). The problem is to fit it to a image curve feature $\mathbf{r}_f(s)$. More rigorously, the problem is stated as:

Given an initial estimate $\mathbf{r}_m(s)$(or $\mathbf{X}_m$ in shape space) with normals $\mathbf{n}_m(s)$ drawn to it and a regularization weight matrix S_m, solve,

$$\mathrm{Min}_X\ (\mathbf{X}-\mathbf{X}_m)^T S_m (\mathbf{X}-\mathbf{X}_m) + \Sigma_{i=1 \text{ to } N}\ 1/\sigma_i^2\ (\nu_i - \mathbf{h}(s_i)^T\ [\mathbf{X}-\mathbf{X}_m])^2 \qquad (31)$$

The problem is solved as follows:

1. Choose samples s_i, i=1,…,N with $s_1=0, s_{i+h}=s_i+h$, $s_N=L$.
2. Apply block matching to establish the position of $\mathbf{r}_f(s)$. For each s_i, extend the normals from the template curve $\mathbf{r}_m(s)$ to the estimated curve $\mathbf{r}_f(s)$+/-δ, to find $\mathbf{r}_f(s_i)$.
3. Initialize the “information matrix” , $S_0=0$. The information matrix S_i is a measure of the strength of each intermediate estimate $\mathbf{X}_i^f$**,** taking account of the first i data points.
4. Initialize the “information weighed sum”, $\mathbf{Z}_0=0$. The $\mathbf{Z}_i$ matrix accumulates the influence of the mean shape and the individual measurements, each with it’s proper weight.
5. Iterate for i=1,…,N:

   $\nu_i = (\mathbf{r}_f(s_i) - \mathbf{r}_m(s_i))\ \mathbf{n}_m(s_i)$;

   $\mathbf{h}(s_i)^T = \mathbf{n}_m(s_i)^T U(s_i) W$;

   $S_i = S_{i-1} + 1/\sigma_i^2\ \mathbf{h}(s_i)\ \mathbf{h}(s_i)^T$;

   $\mathbf{Z}_i = \mathbf{Z}_{i-1} + 1/\sigma_i^2\ \mathbf{h}(s_i)\ \nu_i$;

   $\sigma_i = \sqrt{N}$;

6. The aggregate observation vector is $\mathbf{Z}=\mathbf{Z}_N$ and the associated statistical information vector is $S= S_N$.
7. The best fitting curve in shape space is given by,

$$\mathbf{X}^f=\mathbf{X}_m+(S+S_m)^{-1}\mathbf{Z} \qquad (32)$$

One practical consideration in the regularization problem is to nullify the effect of the initial template $\mathbf{X}_m$ on the translation of the object. This can be affected by the use of projection. operators.

$$S_m=\alpha E^{d'}\mathbf{H}E^d. \qquad (33)$$

In general, if the invariant sub-space has a shape matrix W_s with W_s^+ as it's pseudo-inverse,

$$E^d= I-E^s \quad \text{where } E^d= W^+ W_s W_s^+W. \qquad (34)$$

2.4.2 PROBABILISTIC MODELS OF SHAPE

The deterministic approach in the previous Section generates a unique estimate $\mathbf{X}^f$ of a curve shape from data $\mathbf{r}_f(s)$, moderated using regularization towards a mean shape, the template $\mathbf{X}_m$. If we treat $\mathbf{X}^f$ as a mean or mode of an entire probability distribution, then the fitting problem is a solution which generates a family of curves sharing a common distribution.

The distribution for a curve shape $\mathbf{X}$ is expressed in the form of a posterior density function $p(\mathbf{X}|\mathbf{Q}_f)$, which is the distribution of $\mathbf{X}$ conditioned on $\mathbf{Q}_f$. Using Bayes' formula, this can be expressed as a product of prior density $p(\mathbf{X})$ and observation density $p(\mathbf{Q}_f|\mathbf{X})$,

$$p(\mathbf{X}|\mathbf{Q}_f)=K\,p_0(\mathbf{X})\,p(\mathbf{Q}_f|\mathbf{X}) \qquad (35)$$

where K is a proportionality constant. The prior density can be expressed as a gaussian distribution,

$$p_0(\mathbf{X})=K\exp{-1/2}(\mathbf{X}-\mathbf{X}_m)^T S_m(\mathbf{X}-\mathbf{X}_m) \qquad (36)$$

where K is some proportionality constant.

Similarly, the observation density can be defined as another gaussian distribution,

$p(\mathbf{Q}_f\,|\mathbf{X})=K\exp{-1/2}\ N_X/\ \rho_f^2||\mathbf{Q}-\mathbf{Q}_f||$ (37)

where K is a proportionality constant. Finally, the solution for the posterior distribution is given by the distribution,

$\mathbf{X}|\mathbf{Q}_f \sim N(\mathbf{X}^f, P)$ (38)

with mean $\mathbf{X}^f = S^{-1}(S_m \mathbf{X}_m + N_X/\ \rho_f^2 \mathbf{H}\ \mathbf{X}_f)$ (39)

$S = S_m + N_X/\ \rho_f^2 \mathbf{H}$ (40)

$S_m = N_X/\ \rho_0^2 \mathbf{H}.$ (41)

$P=(S+S_m)^{-1}.$ (42)

N_X = dimension of the shape-space.

2.4.3 MODELS OF MOTION

The motion model is used to extrapolate motion between frames. One can model the interpolation of motion between frames as a Markov process,

$p(\mathbf{X}(t_k)|\ \mathbf{X}(t_1)\ldots.\ \mathbf{X}(t_{k-1}))=p(\ \mathbf{X}(t_k)|\ \mathbf{X}(t_{k-1})).$ (43)

This is a first order Markov process which cannot negotiate direction changes and oscillatory motion. So, one needs to consider higher order processes. A second order process is sufficient for most practical purposes. Such a second order equation maintaining isotropy and spatial uniformity is given below.

$\mathbf{X}(t_k)-\mathbf{X}_m= A_2\ (\mathbf{X}(t_{k-2})-\mathbf{X}_m)+ A_1\ (\mathbf{X}(t_{k-1})-\mathbf{X}_m) +B_0\ \mathbf{w}_k.$ (44)

The mean state solution to this equation is given by,

$X(t_k)- X_m =A(X(t_{k-1})- X_m)$ (45)

and $P(t_k)=A\,P(t_{k-1})A^T+BB^T$ (46)

where,

$$A=\begin{pmatrix}0 & I\\ A_2 & A_1\end{pmatrix} \quad (47)$$

$$B=\begin{pmatrix}0\\ B_0\end{pmatrix} \quad (48)$$

$$X_m=\begin{pmatrix}X_m\\ X_m\end{pmatrix} \quad (49)$$

$X(t_k)=(\,\mathbf{X}^f(t_{k-1})\ \mathbf{X}^f(t_k))^T.$ (50)

$$P(t_k)=\begin{pmatrix}P''(t_k) & P'(t_k)'\\ P'(t_k) & P(t_k)\end{pmatrix} \quad (51)$$

$P''(t_k)=E[(\mathbf{X}(t_{k-1})-\mathbf{X}^f(t_{k-1}))(\,\mathbf{X}(t_{k-1})-\mathbf{X}^f(t_{k-1}))^T]$ (52)

$P'(t_k)=E[(\mathbf{X}(t_k)-\mathbf{X}^f(t_k))(\,\mathbf{X}(t_{k-1})-\mathbf{X}^f(t_{k-1}))^T]$ (53)

$P(t_k)=E[(\mathbf{X}(t_k)-\mathbf{X}^f(t_k))(\,\mathbf{X}(t_k)-\mathbf{X}^f(t_k))^T]$ (54)

$\mathbf{w}_k$ is white gaussian noise and $P(t_k)$ is the "covariance matrix" solved by using the "Riccati" equation. A_2, A_1 and B_0 are harmonic motion parameters in shape space. They can be expressed as,

$A_2= a_2 I_N$ with $a_2=-\exp(-2\beta\tau)$ (55)

$A_1= a_1 I_N$ with $a_1=2\exp(-\beta\tau)\cos(2\pi f\tau)$ (56)

and $B_0= b_0/\sqrt{N}.\mathbf{H}^{-0.5}$. (57)

$N=N_x$ the dimension of the shape space and β, f are the harmonic motion parameters.

2.4.4 CONSTRAINED AND UNCONSTRAINED MOTION

Motion can be decomposed into constrained and unconstrained motion using projection operators as described at the end of Section 2.4.2. Unconstrained motion is in general the translational part of motion (x,y,z) and planar rotation(the so called “Euclidean subspace” in shape-space). If the overall shape-space being used is a 3D affine space, constrained motion is described by the remaining portion of 3D affine space(i.e there is a limit to which the parameters can change from frame to frame). In the present research, block matching has been used to implement unconstrained motion and morphological watershed algorithm has been used to take care of constrained motion(local motion).

2.4.5 BLOCK MATCHING

This algorithm can be used to implement the unconstrained portion (the Euclidean portion) of motion tracking. Tracking tracks the inter frame motion of the centroid of the object. It involves the following steps.

1) Estimation of velocity: The velocity of the rigid body is calculated using block matching. It is an efficient method for estimating local motion. In this paper, the block-matching algorithm has been modified to do object matching algorithm. The aim of the algorithm is to calculate the motion vector d that defines the movement of the rigid body between the current frame and the previous frame. The method can be described by the following minimization:

$$\begin{aligned} &\mathrm{Min}_{d \in \mathbf{P}}\ \varepsilon(d), \\ &\varepsilon(d) = \sum_{n \in B} \Phi\left(I_k(n) - I_{k-1}(n - d)\right) \end{aligned} \qquad (58)$$

where **P** is the search area to which d belongs, and is defined as: $\mathbf{P} = \{n = (n_1, n_2) : -P \le n_1 \le P, -P \le n_2 \le P\}$, and B is the object mask defined in frame I_{k-1}. The idea is illustrated in Figure 7, where an object is sought within I_k which best matches the mask of the object from I_{k-1}. This is actually done by moving the mask in I_{k-1} over the search area **P** and extracting that portion of the frame I_k which can be viewed through the shifted mask. The shifted mask from I_{k-1} and the extracted portion of I_k are then processed to minimize ε (d).

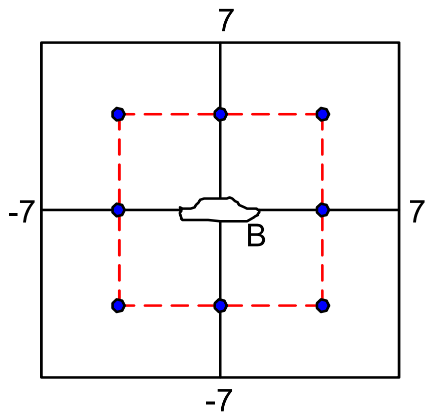


Figure 7. Object mask and the search area.

In the present case, Φ() is the statistical correlation ρ_{xy} between two image frames, where

$$\rho_{xy} = \text{COVARIANCE}(x,y)/\sqrt{\text{VARIANCE}(x)}.\sqrt{\text{VARIANCE}(y)} \qquad (59)$$

A thorough search for d Є **P** giving the minimum error ε is computationally expensive. So, a logarithmic three-step search space has been used. Assuming that $P=2^k-1$ and $P_l=(P+1)/2^l$, where k and l are integers, the new reduced area of search is defined as follows:

$$\mathbf{P}_l=\{n: n=(\pm P_l, \pm P_l) \text{ or } n=(\pm P_l, 0) \text{ or } n=(0, \pm P_l) \text{ or } n=(0,0)\}$$

So, the search space $\mathbf{P}_l$ is reduced to the vertices, midway between vertices and the central point of the half sized original rectangle **P**, as shown in Figure 8.

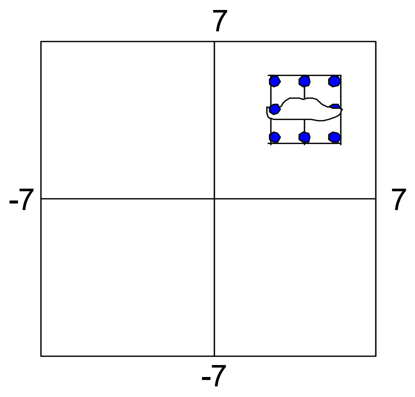


Figure 8. Reduction in search space from frame I_{k-1} to I_k.

The block matching fails when the object rotates substantially bringing into view a new perspective of the object. Cases include old faces turning away, new faces turning in, and new edges appearing in the frame. In this case, the user is required to redefine the contours of the object in the current frame for further automatic tracking. This marks a transition from automatic tracking to manual tracking.

Rotation of objects where no new faces appear has been modeled as a uniformly changing d vector (the inter frame differences in d are stored). Moreover, the visible faces have to be rotation compensated, which is described in Chapter III.

The block-matching algorithm also fails in case of occlusion, where stray objects block the object of interest from the field of view of the user. This case is taken care of by ignoring the frames where the object is occluded and treating the last frame where the object appeared as I_{k-1} and the frame where it reappears as I_k. The motion information is interpolated in between.

For z-axis motion, a minor modification is done to the block-matching algorithm. Before comparison, the mask is expanded or contracted to model zooming out and zooming in.

The block-matching algorithm stores the object mask and the motion parameters. The object mask defines the region of interest B for further processing (constant for linear motion; changes for rotation). The motion parameters are used for motion extrapolation and motion reconstruction in 3D.

2) Motion extrapolation: Assuming uniform linear motion, the vector d from the previous pair of frames (i.e. I_{k-2} and I_{k-1}) can be used as an estimate of d for the current pair of frames (i.e. I_{k-1} and I_k). This saves the overhead of block matching by eliminating the search steps. A change in d indicates a break in uniform linear motion, which might be due to uniform rotation. This can be extrapolated using the difference (d) parameter stored for previous frame pairs. In case of non-uniform motion, the block-matching algorithm has to be invoked to recalculate the parameters.

3) Reconstruction of motion parameters during user intervention: During user intervention, there is a transition from the auto mode to the manual mode. The motion parameters are recreated by finding the coordinates of the mid point of the cleaned mask provided by the user.

The three steps are illustrated in the flowchart shown in Figure 9.

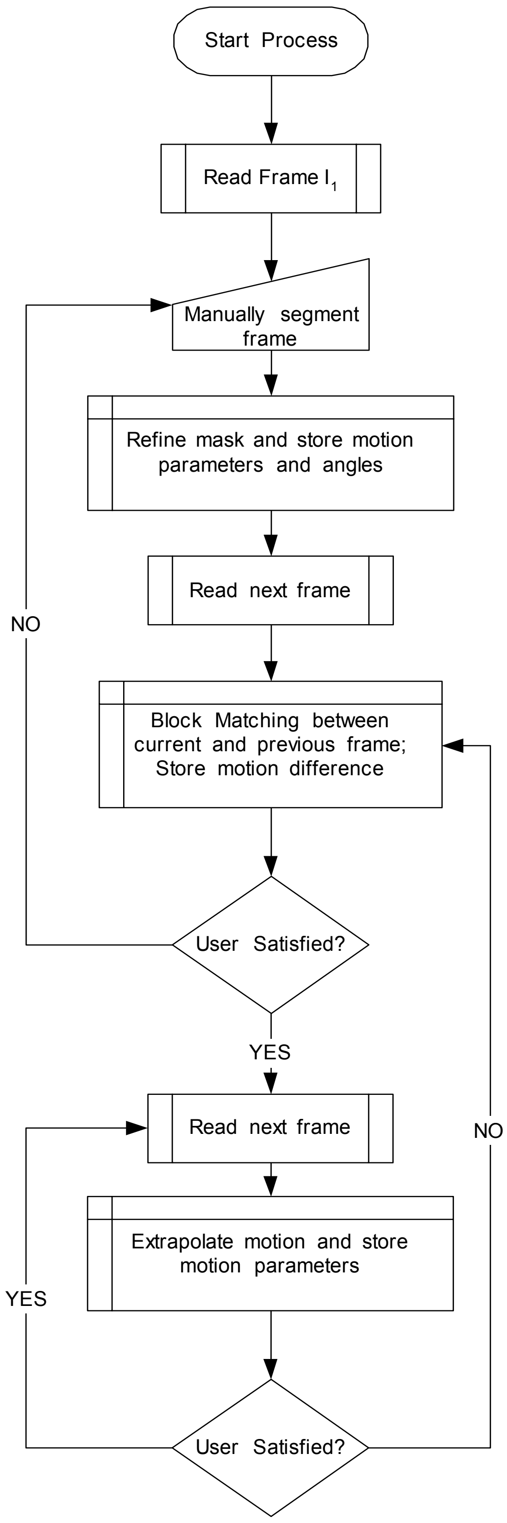

Start Process
Read Frame I_1
Manually segment frame
Refine mask and store motion parameters and angles
Read next frame
Block Matching between current and previous frame; Store motion difference
User Satisfied?
NO
YES
Read next frame
Extrapolate motion and store motion parameters
User Satisfied?
YES
NO

Figure 9. Steps in the block matching algorithm.

2.4.6 CONTOUR TRACKING

The contour tracking algorithm is implemented using the motion model described in Section 2.4.3 along with the curve fitting algorithm of Section 2.4.1. It can be described as follows:

1. Initialization: Assuming that the initial position is known, one can run the curve fitting algorithm with constant artificial measurement. This will force the covariance values and the positional values to settle to their steady state values.
2. Prediction:

   $P''(t_k)=E[(\mathbf{X}(t_{k-1})-\mathbf{X}^f(t_{k-1}))(\mathbf{X}(t_{k-1})-\mathbf{X}^f(t_{k-1}))^T]$

   $P^{P}{}''(t_k)= P(t_{k-1})$

   $P^{P}{}'(t_k)= A_2P'^T(t_{k-1})+ A_1P(t_{k-1})$

   $P^P(t_k)= A_2P''(t_{k-1}) A_2^T+ A_1P'(t_{k-1}) A_2^T + A_2P'^T(t_{k-1}) A_1^T + A_1P(t_{k-1}) A_1^T + B_0B_0^T.$

   $\mathbf{X}^{P}{}'(t_k)= \mathbf{X}^f(t_{k-1})$

   $\mathbf{X}^P(t_k)= A_2\mathbf{X}^{f'}(t_{k-1})+ A_2\, \mathbf{X}^f(t_{k-1})+(I-A_2- A_1)\mathbf{X}_m.$
3. Measurement: Apply the curve fitting algorithm in Section 2.4.1 with $\mathbf{X}^P(t_k)$ as the mean estimated contour. Obtain the aggregated observation vector $\mathbf{Z}(t_k)$ and the information vector $S(t_k)$.

4. Assimilation:

   $K'(t_k) = P^{P'}(t_k)[\ S(t_k)\ P^{P}(t_k)+I]^{-1}$.

   $K(t_k) = P^{P}(t_k)[\ S(t_k)\ P^{P}(t_k)+I]^{-1}$.

   $P''(t_k) = P^{P''}(t_k)-K'(t_k)\ S(t_k)\ P^{P'}(t_k)$.

   $P'(t_k) = P^{P'}(t_k)-K(t_k)\ S(t_k)\ P^{P'}(t_k)$.

   $P(t_k) = P^{P}(t_k)-K(t_k)\ S(t_k)\ P^{P}(t_k)$.

   $\mathbf{X}^{f'}(t_k) = \mathbf{X}^{P'}(t_k)+ K'(t_k)\mathbf{Z}(t_k)$

   $\mathbf{X}^{f}(t_k) = \mathbf{X}^{P}(t_k)+ K(t_k)\mathbf{Z}(t_k)$

CHAPTER III

TEMPLATE MATCHING

Camera maps the 3D world into 2D world co-ordinates. In order to recreate the 3D motion from the 2D frame sequence as captured by the camera, the template matching algorithm is used for the mapping of parameters from 2D to 3D. The predefined library has 2D shapes, which have a direct 1-1 correspondence with existing 3D models in VR. The template matching is done in 2D but the final modeling is done using 3D models. Prior to discussing the method, we discuss some of the situations that are encountered during the video capture process.

3.1 SITUATIONS ENCOUNTERED DURING VIDEO CAPTURE

The template-matching algorithm has to be made adaptive to the following situations.

1. Global rotation: The point of view of the camera while capturing the mobile sequence also adds to the rotation of the scene (global rotation). As shown in Figure 10, when the camera plane is orthogonal to the object plane, there is no global rotation; however, when the camera plane is not orthogonal to the object plane, global rotation results. In terms of rotational dynamics, it is similar to the roll motion of a ship or airplane.

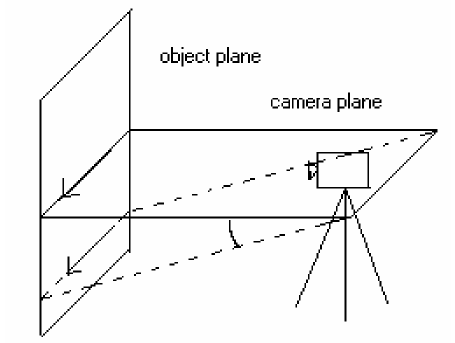


Figure 10. Global rotation due to point of view of camera.

2. Rotation of the object: Other than the roll motion resulting due to the camera perspective, the object may undergo pitch and yaw motions.

Cases 1 and 2 tend to misalign the object and the template (which is typically stored with zero roll, zero pitch and zero yaw angles). The object and the templates have to be aligned before any template matching can be done. This is done by applying *quaternion rotation operator* to the object [25]. Suppose **v** is a vector in R^3 (the pixel coordinates in a frame are vectors with zero z components). If **v** is to be rotated by an angle θ (counterclockwise) about the axis defined by a vector **k**, it has to be operated upon by a quaternion q as follows:

$\mathbf{v}' = q\mathbf{v}q^*$ (60)

where,

$q = \cos\theta/2 + \mathbf{k}\sin\theta/2$ (61)

$q^* = \cos\theta/2 - \mathbf{k}\sin\theta/2$ (62)

and **v**' is the rotated vector. (60) can also be written as [24]:

$\mathbf{v}' = \boldsymbol{L}_q(\mathbf{v})$ (63)

In the present case, the roll, pitch and yaw axes are defined by vectors **r**, **p** and **y** respectively. The camera elevation defines the roll angle, α. The pitch angle is calculated from the orientation of the segmented object in the frame, β. The yaw angle is difficult to calculate. In the present case, the yaw can be compensated for by manually rotating the object until the full-view of a single face (side/front/back) can be seen. Figure 11 demonstrates the concept, where automobile is the object of interest. To compensate for the roll and pitch angles, two quaternions are defined:

$r = \cos\alpha/2 + \mathbf{r}\sin2\alpha/2$ (64)

$p = \cos\beta/2 + \mathbf{p}\sin\beta/2$ (65)

If the object **O** is treated as an aggregate of vectors (with zero *z* components) then it can be compensated for roll and pitch by using,

$\mathbf{O}' = \boldsymbol{L}_{rp}(\mathbf{O})$ (66)

where **O**' is the roll and pitch compensated object and $\boldsymbol{L}_{rp}$ is the combined rotation operator due to quaternions r, p.

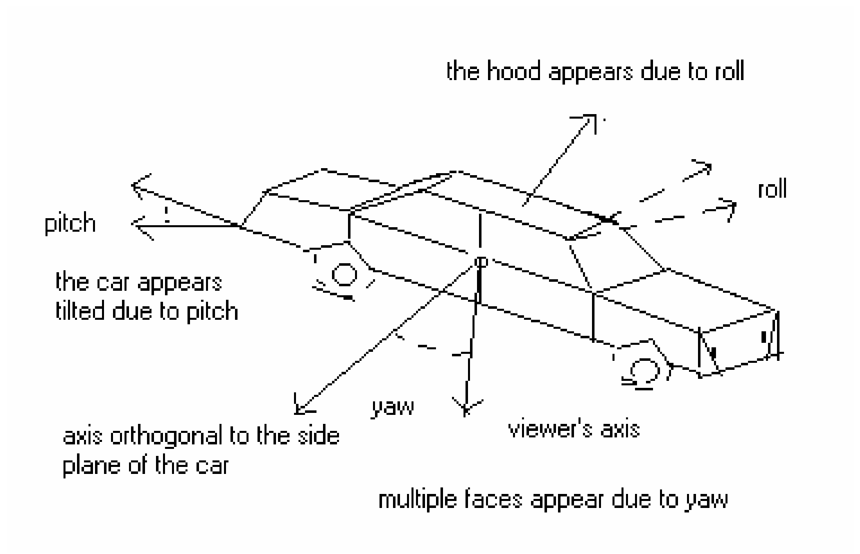

the hood appears due to roll
roll
pitch
the car appears tilted due to pitch
yaw
axis orthogonal to the side plane of the car
viewer's axis
multiple faces appear due to yaw

Figure 11. Effects of roll, pitch and yaw.

3. Global translation: This occurs due to a moving camera. A static camera has been used here, suitable for surveillance of a fixed position or frame. So, there is no global translation in the present case.

4. Positional difference between template and object from frame: The templates are always stored with the origin at the bottom left corner. After segmentation, the object is translated to the bottom left corner of the frame.

5. Scaling: In order to have uniform scaling between object and stored templates, a

predefined number of maximum row pixel positions are used (320). Column pixels are scaled accordingly. Templates are always stored according to this scale. In case of an object scaling, it is done as follows: its maximum length, l is compared against the maximum length, m allowed. If l < m, scale up the row and column coordinates by a factor m/l; and if $l > m$, scale down the row and column coordinates by the same factor.

3.2 TEMPLATE MATCHING PROCESS

A segmented object **O** is chosen to be compared with the given template library. The algorithm has three main steps:

1. Compensate **O** for global rotation, pitch and yaw effects, as described in Section 3.1, which yields **O**'. In the present case, it is assumed that the pitch angle is zero and the roll angle is predefined by the camera tilt. The yaw angle is calculated in the tracking algorithm. So, the rotation operator defined above can be applied to extract a single face of the object of interest (e.g side, face/front, face/backface). In the template library, car shapes are stored with only the left faces visible. So, for template matching, the rotation compensation is such that **O'** shows its left face. If the right face is visible, the object is rotated 180 degrees to obtain the left face, as shown in Figure 12. It is assumed here that the sequence of frames contains atleast one sideview of the car. If there are none, then template matching compares front faces to determine the shape.

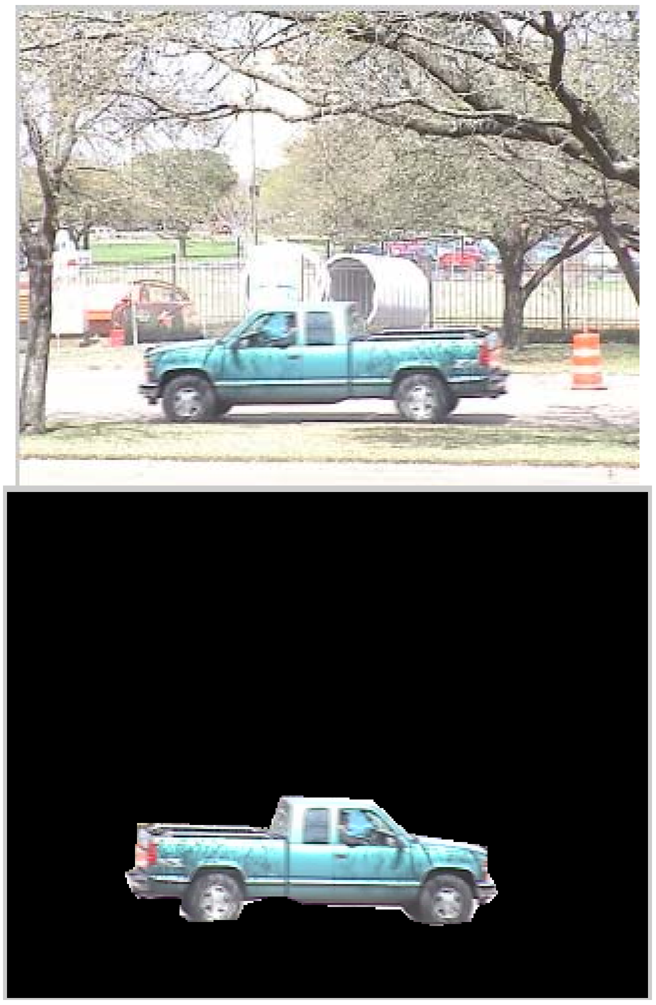
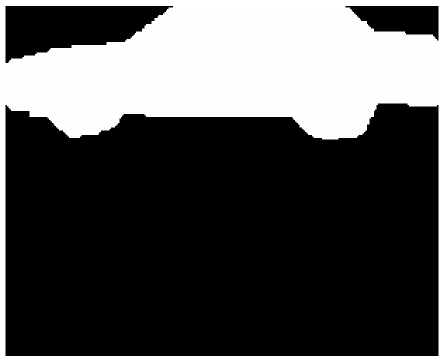


Figure 12. Rotation of object by 180 degrees to obtain left face.

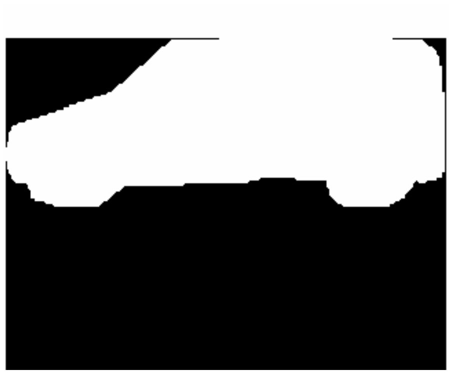
2. Using block matching, compare mask **O'** with all the templates in the stored library. Scaling and translation is necessary for this step. Store the best-match template. Since the mask **O'** is compared with all the templates, the library is to be kept small for better performance. Any minor customization in template shape is stored as separate information along with the template. The criterion used for block matching is correlation. If none of the templates meet a predefined correlation threshold, the mask **O'** can be assumed to be a new shape and can be added to the library.

3. Once the template has been found, the corresponding 3D object is retrieved and used in the virtual environment. The extracted motion information is then applied to the object.

Figure 13 shows some of the masks that are stored in the library, while Figure 14 shows an unknown shape that is being compared with the stored masks. Note how scaling is done. The top of the car is brought to the top row. The length of the car is made 320 pixels and the columns are scaled accordingly.



(a) car



(b) minivan.

Figure 13. Two shapes from the template library: (a) car and (b) minivan.

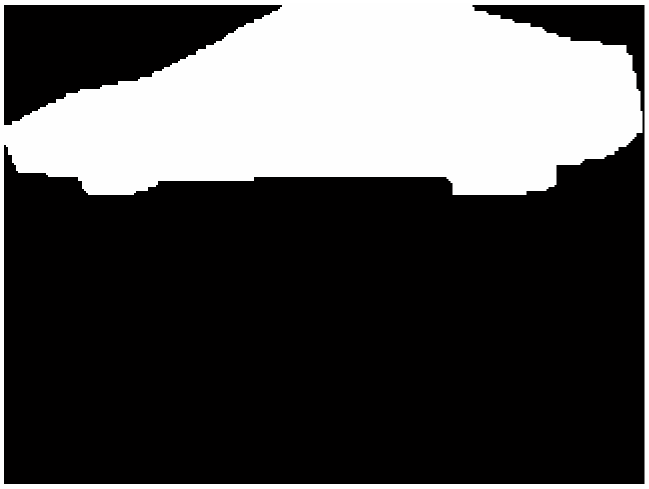


Figure 14. An unknown shape, which matches with the car in the template library.

Issues, such as growing (or learning) library, the order of template storage in the library, and using first-fit method rather than best-fit method for template search are not considered in this research. They are a good candidate for future work.

CHAPTER IV

RESULTS AND CONCLUSION

4.1 EXPERIMENTAL SETUP

The experimental setup consists of viewing a street junction from a vantage point (a building). The x and z span of the view are shown in Figure 15. The digital camera records at a rate of 29.97 frames per second (fps). In order to increase processing speed the number of frames has been down sampled by a factor of 4. Otherwise, unnecessary processing time is wasted to find redundant motion parameters. It has been assumed that there is no motion of the object in the y direction (smooth road without ditches and bumps) and the only rotation possible is about the y-axis. There is a global rotation due to the tilt of the camera (approximately 10 degrees), due to which a portion of the roof of vehicles is visible. For tracking, this is not a problem but for template matching, it has to be roll compensated to show the side face only.

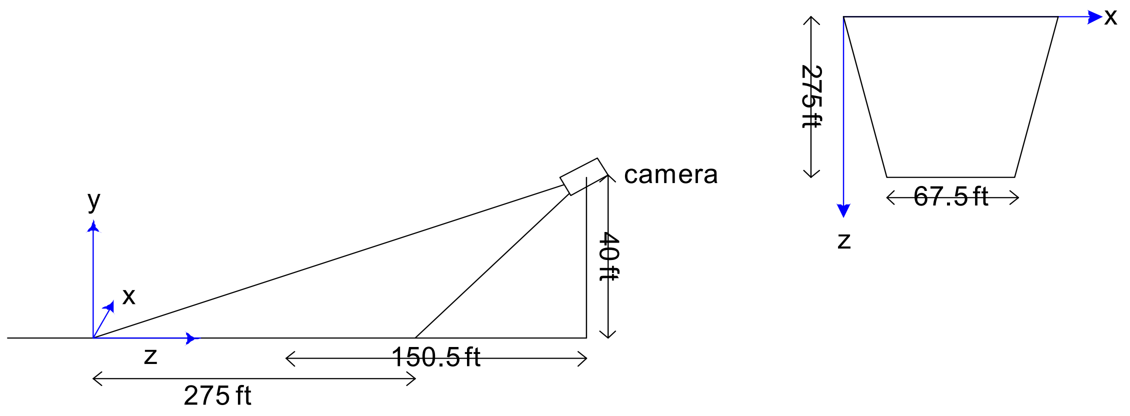


Figure 15. Schematic diagram of experimental setup.

The results from tracking are stored in a file with field's time, x-coordinate, y-coordinate, z-coordinate and rotation about y-axis. A sample data file and corresponding vehicle positions are shown in Figure 16. The results from template matching identify the closest existing template that can be used to recreate the motion in the VR world with the motion information extracted during tracking. A VRML file is generated with the extracted object information from the library and the motion information. The other objects in the scene are static objects, which can be modeled offline and inserted in the VRML file to populate the scene. Multiple viewpoints cane be created in the VRML file with one that is attached to the extracted object. This enables the user to be present in or on the moving object allowing the person to see the events within the 3D environment.

A small demonstration can be seen at the website: http://ie.tamu.edu/tatp99/RightTurn.avi and http://ie.tamu.edu/tatp99/Demo.htm. The first link contains a video clip that was used for the experiment. The pickup truck that makes a right turn in the video clip is the object of interest, for which the user creates a mask similar to the mask shown in Figure 1. The tracking model determines the speed and direction of motion, which is used to create the vrml file that can be seen using the second link. It shows the 3D reconstruction of the object from the library and the resultant motion. The embedded vrml file within the html file needs a vrml plugin for the browser. It has been tested using WorldView 2.1 and Cortona vrml plugins for Internet Explorer 6 and Netscape Navigator 4. The 3D world has not been populated with the static objects in the scene. The truck motion has been scaled down to every fourth frame, starting with the 81st frame of the video clip for a duration of 3 seconds. The scaled down approach improves performance at the cost of appearance of slightly jagged motion. This limitation can be easily overcome by using a faster processor with more memory. The motion playback has been embedded in a loop to enable viewing the model from any viewpoint for as long as the user needs.

```
Frame   x        y       z       y-rot (radians)
-------------------------------------------------------------
  01  51.5000    6.0000  93.9583    2.0336
  27  58.0000    6.0000  95.6771    2.0336
  41  62.5000    6.0000  99.1146    2.0336
  50  67.0000    6.0000 100.2604    2.0336
  57  73.5000    6.0000 103.1250    2.0336
  65  80.0000    6.0000 107.1354    2.0336
  74  88.0000    6.0000 111.1458    2.0336
  81  96.5000    6.0000 112.8646    2.0336
  85 100.5000    6.0000 114.0104    2.0336
  89 106.5000    6.0000 117.0104    2.0336
  93 109.5000    6.0000 118.5937    2.0336
  97 115.5000    6.0000 119.5937    2.0336
 101 121.0000    6.0000 120.8854    2.0336
 105 127.0000    6.0000 123.8854    2.0336
 109 132.5000    6.0000 124.8958    2.0336
 113 138.5000    6.0000 127.8958    2.0336
 117 145.5000    6.0000 129.4792    2.1023
 121 152.5000    6.0000 131.4792    2.1222
 125 160.5000    6.0000 133.4896    2.1512
 129 169.5000    6.0000 135.4896    2.1941
 133 180.0000    6.0000 136.9271    2.2432
 137 189.0000    6.0000 138.9271    2.3011
 141 199.0000    6.0000 140.3646    2.3721
 145 210.0000    6.0000 142.3646    2.4545
 149 222.0000    6.0000 145.5208    2.5433
 153 234.0000    6.0000 147.5208    2.6423
 157 247.5000    6.0000 147.8125    2.7540
 161 258.5000    6.0000 147.9125    2.8700
 165 269.0000    6.0000 151.2500    3.0020
 169 280.0000    6.0000 151.2500    3.1416
 173 293.0000    6.0000 151.2500    3.1416
 177 305.0000    6.0000 151.2500    3.1416
 181 317.0000    6.0000 151.2500    3.1416
```

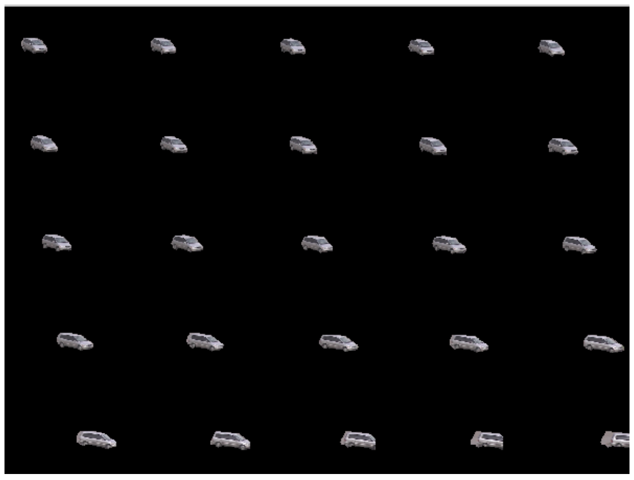


Figure 16. Motion information and masks from a left turn sequence.

4.2 DISCUSSIONS AND FUTURE WORK

This method demonstrates the reconstruction of 3D environment from perspective 2D images using a single camera. The location and orientation of camera is fixed, which provides information about the region that is being captured. This information is pre-processed to calibrate the system, and model the static objects in the environment. In the case where the user needs to track multiple moving objects, the algorithm needs to be run multiple times, with each run masking and tracking a different object. The stored information can then be combined in the VRML file to model the motion of the different objects.

The algorithm for semi-automatic object segmentation and tracking was initially implemented using MATLAB R12. This version performed well in batch mode, but had performance problems while running real-time. This necessitated development of a C++ version on a SGI platform. The immersive virtual environment in our laboratory is driven by SGI, which made it easier to interface the data. This immersive environment enables the creation of a more realistic 3D environment for the user to participate in the events. The library of masks was created and stored using VRML.

Emphasis was given on practical aspects of the problem. A Sony Digital Video camera was used using NTSC capture mode at 29.97 frames per second. The frame size was 320 x 240 pixels. Refining the user defined mask was given more attention as we realized that it is not realistic to expect the user to outline the perfect mask. A higher level of user interaction was introduced to ensure that the user is satisfied with the masks that are created. It is possible to automate the process with predefined threshold levels for the masks. This can be used for automatic control and detection of shape abnormalities.

The method has a number of applications in industrial engineering. It can be used as a real-time feedback method to detect and avoid congestions in a conveyor network caused due to odd shaped or bulky items. Another application is the post processing (reconstruction) of an incident (e.g. on a shop floor, or a traffic situation) from multiple perspectives in 3D, based on the events captured on video leading up to the incident.

The work in the future will be towards developing faster search algorithms and better motion models(both Gaussian and Non-gaussian/ higher order Markov processes), making the algorithm suitable for segmentation of non-rigid bodies, bodies with appendages, handling multiple cameras and moving cameras, tracking multiple objects in one pass, face composition and decomposition, automatic growing (or learning) of the library of masks, dealing with uncalibrated setup and distributed operations over a network.

REFERENCES

1. A. Bovik, (Ed.) *Handbook of Image and Video Processing.* San Diego, CA: Academic Press, 2000.

2. F. Marques, M. Pardas and P. Salembier, *Coding –Oriented Segmentation of Video Sequences in Video Coding.* San Diego, CA: Academic Pub, 1996.

3. J. Shi, J. Malik, T. Leung and S. Belongie, "Image and video segmentation: the normalized cut framework," *ICIP'98*, vol.1, pp. 943-947, 1998.

4. P.J. Burt, J.R. Bergen, R. Hingorani, R.J. Kolczynski, W.A. Lee, A. Leung, J. Lubin and J. Shvaytser, "Object tracking with a moving camera," *MOTION89*, vol. 2, pp. 2-12, 1989.

5. M. Schultz and T.E. Ebrahimi, "Matching error based criterion of region merging for joint motion estimation and segmentation techniques," *ICIP'96*, vol.2, pp. 509-512, 1996.

6. A.M. Tekalp, *Digital Video Processing*. Upper Saddle River, NJ: Prentice-Hall, 1995.

7. MPEG Video and SNHC Groups. (Oct. 1997) Committee draft of MPEG-4, part 2, 14496-2, Technical Report, ISO/IEC JTC/SC29/WG11/N1902, ISO/IEC, Fribourg, Switzerland, October 1997.

8. R. Castagno, T.E. Ebrahimi, and M. Kunt, "Video segmentation based on Multiple Features for Interactive Multimedia Applications," *IEEE Transactions on Circuits and Systems for Video Tech*nology, vol. 8, pp. 562-571, 1998.

9. P. Salembier, F. Marques, M. Pardas, J.R. Morros, I. Corset, S. Jeannin, L. Bouchard,., F. Meyer and B. Marcotegui, "Segmentation-based video coding

system allowing the manipulation of objects," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 60-74, 1997

10. P. Correia and F. Pereira, "User interaction in content-based video coding and indexing," *EUSIPCO-98*, Rhodes, Greece, 1998.
11. F. Meyer, "Morphological multiscale and interactive segmentation," *IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing*, Antalya, Turkey, 1999.
12. B. Marcotegui, F. Zanoguera, P. Correia, R. Rosa, F. Marques, R. Mech and M. Wollborn, "A Video Object Generation Tool Allowing Friendly User Interaction," *ACTS-AC098MoMuSys*, 1998.
13. Z. Zhang, R. Deriche, O. Faugeras, and Q. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," Technical Report 2273, The French National Institute for Research in Computer Science and Control (INRIA), Sophia-Antipolis, France, 1994.
14. G. Sparr, "An algebraic-analytic method for reconstruction from image correspondences," *Proceedings of the 7th Scandinavian Conference on Image Analysis*, 274-281, 1991.
15. D. Zetu, P. Banerjee and P. Schneider, "Data input model for virtual reality-aided facility layout design," *IIE Transactions*, vol. 30, pp. 597-620, 1998.
16. T. Kanade, P.J. Narayanan, P.W. Rander, "Virtualized reality: Concepts and early results," *IEEE Workshop on Representation of Visual Scenes,* Boston. citeseer.nj.nec.com/kanade95virtualized.html, 1995.
17. T. Kanade, P. Rander, "Virtualized Reality: Constructing virtual worlds from real scenes," *IEEE publication*, vol. 2, pp.34-47, 1997.

18. T. Kanade, P.J. Narayanan, P.W. Rander, "Recovery of dynamic scene structure from multiple image sequences," *International Conference on Multisensor Fusion and Integration of Intelligent Systems,* Washington D.C., pp. 305-312, 1996.

19. Wang Chih-Ming, Lee Chia-Wen, Chang Yao-Jen, Chen Yung-Chang, "Realtime object extraction and tracking with an atctive camera using image mosaics," *ICME*, www.cs.ccu.edu.tw/~cwlin/pub/icme02object.pdf, 2001.

20. R. Fablet, P. Bouthemy, "Extraction of regions of interest based on motion activity for video retrieval and partial query," *IPMU 2002 Special Session for "Intelligent systems for Video processing".* www.cs.brown.edu/people/rfablet/Papers/ipmu2002.pdf, 2002.

21. J. Pan, C.W. Lin, C. Gu and M.T. Sun, "A robust video object segmentation scheme with pre-stored background information," *ISCAS.* www.cs.ccu.edu.tw/~cwlin/pub/iscas02seg.pdf, 2002.

22. A. Blake and M. Isard, *Active Contours,* Cambridge, England: Springer, 1998.

23. D.J. Fleet, *Measurement of Image Velocity*, The Kluwer International Series in Engineering and Computer Science, 169, Boston, MA: Kluwer Academic Publishers, 1992.

24. Image Processing Toolbox, *MATLAB User's Guide*, The Math Works Inc., 1997.

25. J.B. Kuipers, *Quaternions and Rotation Sequences*, Princeton, NJ: Princeton University Press, 1999.

VITA

Sumantra Dasgupta was born on December 31, 1976 in India. He completed his Bachelor of Engineering in electrical and electronics engineering from Birla Institute of Technology, MESRA, India in August 2000. His research interests are computer vision, image processing, stereo vision, video processing, soft computing and imaging/vision hardware implementations. His permanent address is 8B, Burnpur Road, 1st Floor, PO-Chelidanga, Asansol South, Dist: Burdwan, W.B., India 713304, ph: 011913463261551, email: rishi123102@yahoo.com.