EXPRESSION MORPHING BETWEEN DIFFERENT ORIENTATIONS

by

TAO FU
B.S. Hohai University, 1993

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2004

ABSTRACT

How to generate new views based on given reference images has been an important and interesting topic in the area of image-based rendering. Two important algorithms that can be used are field morphing and view morphing. Field morphing, which is an algorithm of image morphing, generates new views based on two reference images which were taken at the same viewpoint. The most successful result of field morphing is morphing from one person's face to the other one's face. View morphing, which is an algorithm of view synthesis, generates in-between views based on two reference views which were taken at different viewpoints for the same object. The result of view morphing is often an animation of moving one object from the viewpoint of one reference image to the viewpoint of the other one.

In this thesis, we proposed a new framework that integrates field morphing and view morphing to solve the problem of expression morphing. Based on four reference images, we successfully generate the morphing from one viewpoint with one expression to another viewpoint with a different expression. We also proposed a new approach to eliminate artifacts that frequently occur in view morphing due to occlusions and in field morphing due to some unforeseen combination of feature lines. We solve these problems by relaxing the monotonicity assumption to piece-wise monotonicity along the epipolar lines. Our experimental results demonstrate the efficiency of this approach in handling occlusions for more realistic synthesis of novel views.

ACKNOWLEDGMENTS

I would like to thank, first and foremost, my advisor Dr. Hassan Foroosh, for his guidance during the completion of this thesis. His expertise, understanding, and patience have been a tremendous influence and an invaluable resource for me. I appreciate his advice in helping me doing the research and his assistance in writing the thesis.

Also, I would like to thank the other members of my committee, Dr. Christian Bauer, Dr. Harold Klee, and Dr. Charles Hughes for taking time out from their busy schedules and giving me comments and suggestions.

Special thanks to my family for the support they provided me and in particular, my wife Yilin. Without her love and encourage, all my work could not have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

Image morphing has been widely used in the entertainment industry to achieve powerful visual effects for many years. One of the most unforgettable results is from Michael Jackson's famous MTV "black and white", in which one face transformed into the other smoothly. Generally speaking, image morphing is a coupling of image warping with color interpolation [5]. After specifying corresponding features on reference images (such as points, lines, and mesh nodes), the pixel mapping function can be calculated between the novel image and reference images. Then color interpolations, mostly cross-dissolve, can be applied to generate in-between images. The most compelling results of image morphing are those involving transformations from one person to another, and morphing between different expressions of the same person. Two important requirements that make image morphing strikingly realistic are: reference images from the same viewpoint and objects look alike.

One of the earliest methodologies used for image morphing was mesh warping [6], which is based on using corresponding mesh nodes as image features. The in-between mesh nodes are the linear interpolations of mesh nodes between reference images. After image warping, cross-dissolving can be applied to generate in-between images.

Field morphing, which was developed by Beier and Neely [1], simplified the user interface to handle correspondences by means of line pairs. After specifying corresponding feature lines on reference images, pixel mapping functions can be calculated based on these feature lines.

The warping functions of image morphing have also been widely studied. Since feature lines and curves can be point sampled, it is possible to consider the features on an image to be specified by a set of points. Based on scattered data interpolation, Ruprecht and Muller [7] investigated several warping functions and showed that radial basis functions, particularly multi-quadrics, are suitable for application to image warping. Using a small number of anchor points, Arad et al. [8] demonstrated that radial basis functions provided a power mechanism for processing facial expressions based on reference images.

Lee et al. [9] developed a two-dimensional deformation technique to calculate warp functions. The resulting warp is C1-continuous and one-to-one and reflects the feature correspondences between the images. Lee et al. [10] also introduced snakes into image morphing, which can reduce the burden of feature specification. The multilevel free-form deformation (MFFD) used by Lee et al. [10] can achieve C2-continuous and one-to-one warps among feature point pairs.

View synthesis, which also generates novel views based on given reference views, has been very popular for many years in both computer vision and computer graphics. View synthesis methods can be divided into two categories: one is reconstructing the scene using given views, and then rendering the novel view using reprojection and texture mapping. This 2D-3D-2D approach can generate novel views from different viewpoints if the reconstruction is accurate enough. However, because of the complexity of the real world, accumulated errors during 2D to 3D and 3D to 2D cannot be avoided, which also make this approach difficult in practice. The other approach, which is a 2D view synthesis technique, is referred to as image based rendering, and

relies on generating novel views by rearranging pixels from reference images directly without reconstruction of a 3D model. Because this approach bypasses the 3D reconstruction, the complexity and accumulated errors can be reduced dramatically.

Among the early works of 2D view synthesis, Chen and Williams [11] described how to interpolate new views of a scene based on images taken from closely spaced viewpoints. Their result showed image-based rendering could speed up the rendering time. Their observation led to the development of some later hardware systems to achieve real-time rendering rates for synthetic 3D scenes [12-14].

QuickTime VR [15,16] is the system developed by Chen in Apple Computer Inc. It can provide panoramic visualization of scenes using cylindrical image mosaics. Walking in a space is accomplished by "hopping" to different panoramic point. The success of QuickTime VR and other systems like Surround video [17], IPIX [18], Smooth-Move [19] and RealVR [20] brought image-based scene visualization into mainstream.

McMillan and Bishop [21] also developed an image-based rendering approach using Adelson and Bergen's plenoptic modeling concept [22]. They propose a technique for sampling, reconstruction, and resampling of the plenoptic function, which is similar to the QuickTime VR. A panoramic image is generated in this way using a set of uncalibrated images, with new views reprojected using the plenoptic function.

Debevec et al. [23] proposed a photogrammetric approach to construct 3D models of architectural scenes. Their work demonstrated that user-interaction could also play an important role in view synthesis and in constructing high-quality 3D models.

Levoy [24] and Gortler et al. [25] both developed novel ray-based methods for view synthesis: i.e. light field and Lumigraph. Their approaches use a four-dimensional ray space to represent any visible scene. Although new views can be reconstructed very fast without knowing the pixel correspondences, both methods require extensive data acquisition.

View morphing [2], proposed by Seitz and Dyer, solves the problem of synthesizing novel views based on the reference views taken from different viewpoints for the same object. By exploiting the epipolar geometry associated with a stereo pair, physically-valid in-between novel views can be generated without knowing the camera parameters. The limitation of this approach is that the viewpoints of novel views have to lie on the straight line connecting the reference viewpoints.

Recently, image-based rendering using the plenoptic function has been investigated further by several researchers. Shum and He [26] presented a 3D plenoptic function called concentric mosaics, in which the camera motion is restricted to planar concentric circles, and concentric mosaics [27] are created by composing slit images taken from different locations. Novel views can be rendered by combining the appropriate captured rays. Takahashi et al [28] created a 3D plenoptic function for reconstructing novel views in large-scale scenes. They use an omni-directional camera [29] to capture mosaicing panoramic images along a straight line and

recording the capturing position of each image using a GPS system. Plenoptic stitching, proposed by Aliaga and Carlbom [30], is another technique based on plenoptic function. It parameterizes a 4D plenoptic function and supports walkthrough applications in large, arbitrarily shaped environment.

Dynamic view morphing, which interpolates views for dynamic scenes, extends the concept of view morphing. Manning and Dyer [31] assumed that each object in the original scene underwent a series of rigid translations and generated one possible physically valid scene transformation. Xiao et al [32] applied view morphing to non-rigid objects that contain both rotation and translation. They assume that a non-rigid object can be separated into several rigid parts. For each part, the least distortion method [33] is used to determine its moving path. Our work focuses on two main issues:

1. Expression morphing: the emphasis is on developing tools that would allow for synthesizing new views of human faces with expression change. In our case, a non-rigid object (i.e. a human face) contains both rotation and translation, and facial deformations. Moreover, this object cannot be separated into rigid parts. The approach that we propose combines field morphing and view morphing in a single framework. It takes advantage of field morphing's ability to morph one expression to the other at the same viewpoint, and view morphing's ability to morph same expression from different viewpoints. Based on four reference images we successfully generate the morphing from one viewpoint with one expression to another viewpoint with a different expression.

2. The second issue that we address in this thesis is the elimination of the artifacts known as the "ghosting effects", which are frequently caused in view morphing due to occlusions. Because reference images are typically taken from different viewpoints, some scene points may not be visible in both images and some parts of the object may be more visible in one image than the other. As a result, the assumption of monotonicity and order invariance of feature points along the epipolar lines is often violated, and hence during the course of morph transition, the unmatched points appear to slowly fade in or out in the occluded areas. These artifacts can be very noticeable and disturbing when reference images are taken from distant viewpoints and directions. Our reference images are just in this case, and hence simple cross-dissolve of reference images will inevitably introduce many such artifacts. We propose to solve this problem by relaxing the monotonicity assumption to piece-wise monotonicity along the epipolar lines. For this purpose, we segment the object into several areas and divide it into labeled regions. According to the label of each region, the pixels of that region can be mapped from one of the reference images or from cross-dissolve of both images. Our experimental results demonstrate the efficiency of this approach in handling occlusions for more realistic synthesis of novel views.

Since our approach integrates the two methodologies of field morphing and view morphing in a single framework, we have devoted the next two chapters of this thesis to a quick overview of the technical details of these two approaches.

CHAPTER 2. FIELD MORPHING

Field morphing is an image morphing algorithm originally proposed by Beier and Neely [1]. After specifying feature points and feature lines in reference images, the position of feature lines in the in-between images can be linearly interpolated. A mapping function is then used to map the pixels from the in-between image to reference images. This is done based on all the feature lines. The final pixel values in the in-between image are the results of cross-dissolving from reference images.

2.1 Field Morphing Steps

The field morphing algorithm can be divided into the following steps:

1. Specify a series of corresponding feature lines $\mathbf{l_0}$ and $\mathbf{l_1}$ from given images I_0 and I_1, respectively.
2. For each pair of lines $\mathbf{l_0}$ and $\mathbf{l_1}$, interpolate the corresponding endpoints and get a new line $\mathbf{l_s}$, which is the feature line corresponding to $\mathbf{l_0}$ and $\mathbf{l_1}$ in the in-between image I_s.
3. Calculate the inverse mapping $\mathbf{m_0}$ based on line pair $\mathbf{l_0}$ and $\mathbf{l_s}$, which maps each pixel from I_s to its corresponding pixel in I_0.
4. Calculate the inverse mapping $\mathbf{m_1}$ based on line pair $\mathbf{l_1}$ and $\mathbf{l_s}$, which maps each pixel from I_s to its corresponding pixel in I_1.
5. For each point $\boldsymbol{p_s}$ in I_s, cross-dissolve $\mathbf{m_0}(\boldsymbol{p_s})$ and $\mathbf{m_1}(\boldsymbol{p_s})$ to get the pixel value for $\boldsymbol{p_s}$.

2.2 Calculating the Mapping Functions

The most important step of field morphing is how to calculate the mapping functions $\mathbf{m_0}$ and $\mathbf{m_1}$ based on specified feature lines. Actually, there are two type of mapping functions: forward mapping and reverse mapping. Let's define the reference image as the source image and the in-between image as the destination image. Forward mapping scans each pixel in the source image and calculates its corresponding position in the destination image; whereas reverse mapping scans each pixel in the destination image and samples the corresponding pixel from the source image. Because reverse mapping has the advantage that each pixel in the destination image can be calculated and get appropriate value, it is often the preferred approach in field morphing.

2.2.1 Mapping with One Pair of Feature Lines

In field morphing, each pair of corresponding feature lines in the source and destination images can define a mapping from one image to the other. Suppose **X** is an image pixel in the destination image and we want to map **X** to its corresponding pixel **X'** in the source image. Also, suppose **PQ** and **P'Q'** are one pair of corresponding feature lines in the destination image and the source image, respectively.

Let

$$\mathbf{u} = \frac{(\mathbf{X} - \mathbf{P}) \cdot (\mathbf{Q} - \mathbf{P})}{\| \mathbf{Q} - \mathbf{P} \|^2} \qquad (2.1)$$

$$\mathbf{v} = \frac{(\mathbf{X} - \mathbf{P}) \cdot \text{Perpendicular}(\mathbf{Q} - \mathbf{P})}{\| \mathbf{Q} - \mathbf{P} \|} \tag{2.2}$$

then

$$\mathbf{X'} = \mathbf{P'} + \mathbf{u} \cdot (\mathbf{Q'} - \mathbf{P'}) + \frac{\mathbf{v} \cdot \text{Perpendicular}(\mathbf{Q'} - \mathbf{P'})}{\| \mathbf{Q'} - \mathbf{P'} \|} \tag{2.3}$$

where Perpendicular() returns the vector perpendicular to, and the same length as, the input vector. (There are two perpendicular vectors; either the left or the right one can be used, as long as it is consistently used throughout, ***u*** specifies the position along the line, which takes values in the range 0 to 1 as the pixel moves from **P** to **Q**, and is less than 0 or greater than 1 outside that range, and **v** is the perpendicular distance in pixels from the line.

So, the mapping procedure using one pair of feature lines is as follows:

For each pixel **X** in the destination image

calculate the corresponding **u, v**

calculate the **X'** in the source image for using **u, v** ;

destination image(**X**) = source image(**X'**);

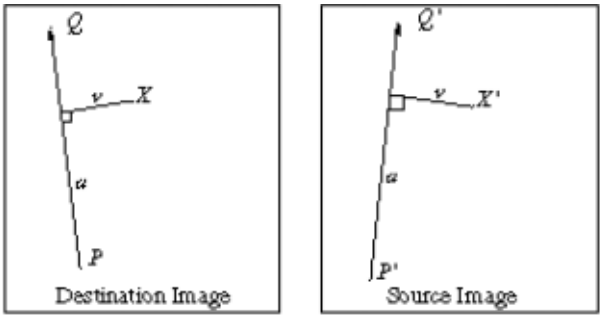


Figure 1: Calculate Mapping Pixel Using One Pair Of Feature Lines

2.2.2 Mapping with Multiple Line Pairs

In the case where there are multiple pairs of feature lines, field morphing introduces different weighting for the coordinates calculated using different pairs of lines. The final solution is the weighted result of all feature lines.

Let $\mathbf{X_i'}$ ($\mathbf{i} = 1 \ldots n$, n is the number of total feature line pairs) be the result pixel in the source image calculated by the **i**th pair of feature lines. Suppose displacement $\mathbf{D_i} = \mathbf{X_i'} - \mathbf{X}$ is the difference between the pixel location in the source and destination images. One weight will be calculated for each displacement $\mathbf{D_i}$. As shown in Figure 2, the weighted average of the displacements is also calculated and added to the current pixel location **X** to determine the position **X'** in the source image.

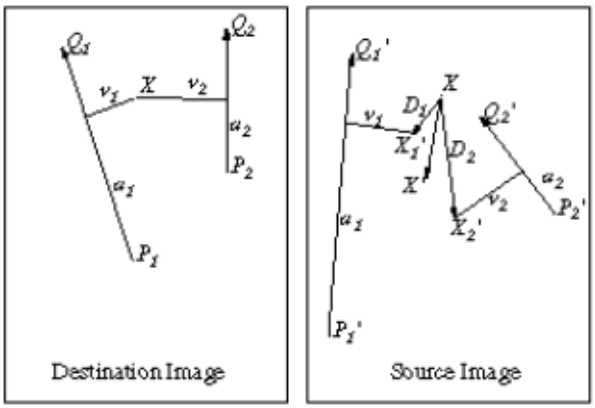


Figure 2: Pixel Mapping Using Multiple Feature Line Pairs

The weight for each displacement is calculated using the following equation:

$$weight = \left(\frac{length^{p}}{(a + dist)} \right)^{b} \tag{2.4}$$

where *length* is the length of the line, *dist* is the distance from the pixel to the line, and ***a*****,** *b* and *p* are constants that can be used to change the relative effect of the lines.

The mapping procedure using multiple line pairs is as follows:

For each pixel **X** in the destination

DSUM = (0,0)

weightsum = 0

For each line $\mathbf{P_iQ_i}$ (i = 1…n, n is the number of total feature line pairs)

calculate **u**, **v** based on $\mathbf{P_i}$**,** $\mathbf{Q_i}$

calculate $\mathbf{X_i'}$ based on **u, v,** $\mathbf{P_i'}$**,** $\mathbf{Q_i'}$

calculate displacement $\mathbf{D_i} = \mathbf{X_i'} - \mathbf{X_i}$ for this line

dist = shortest distance from **X** to $\mathbf{P_iQ_i}$

weight = $(length^{p} / (a + dist))^{b}$

DSUM = **DSUM** + $\mathbf{D_i}$*weight

weightsum = weightsum + weight

X' = **X** + **DSUM** / weightsum

destination image(**X**) = source image(**X'**)

2.3 Parameters in Field Morphing

There are three parameters in field morphing: *a, b,* and *p*. In order to get a better morphing result, these parameters should be specified carefully.

- If *a* is a positive number very close to zero, the weight will be nearly infinity when the distance from the line to the pixel is zero. This will make the pixel on the line go exactly where it should. A larger value will yield a more smooth warping, but with less precise control.
- The variable *b* determines how the relative strength of different lines falls off with distance. If *b* is large, only feature lines near the pixel will affect it; If *b* is zero, every line will affect each pixel equally. The range of *b* is usually [0.5, 2].
- The value of *p* is typically in the range [0, 1]; if it is zero, then all lines have the same weight, if it is one, then longer lines have a greater relative weight than shorter lines.

2.4 Problems with Field Morphing

The main problem with field morphing is that it is a shape-distorting transformation. It tends to bend straight lines, yielding quit unintuitive image transitions. In particular, the projective mapping of a planar surface between two different views has the following form:

$$\mathbf{H}(x,y) = \left(\frac{ax+by+c}{gx+hy+i}, \frac{dx+ey+f}{gx+hy+i} \right) \tag{2.5}$$

Such, projective mappings are not preserved under 2D linear interpolation because the sum of such expressions is in general a ratio of quadratics and therefore not a projective mapping.

CHAPTER 3. VIEW MORPHING

View morphing was proposed by Seitz and Dyer [2]. As a 2-D view synthesis algorithm, it can be used to generate shape-preserving novel views based on given reference views without knowing the camera parameters. The image centers of novel views are located along the line $\mathbf{C_0C_1}$, i.e. the line joining the camera perspective centers for the reference images. In order to explain this approach, we first provide a brief description of a pinhole camera model.

3.1 Pinhole Camera Model

As shown in Figure 3, a pinhole camera projects points in the 3D space into an images plane. We consider a 3D coordinate system attached to the camera (i.e. the canonical coordinate frame): In this coordinate system the origin be is at the camera center **C**, and the x and y axes are parallel to the image axes, with the z-axis determined by the right hand rule and intersects the image plane at the so called principal point. The image plane (also called focal plane) is given by the plane $z = f$.

Under the pinhole camera model, a point **P** in the Euclidean space is mapped to a point **p** in the image plane, which is given by the intersection of the line PC with the image plane.

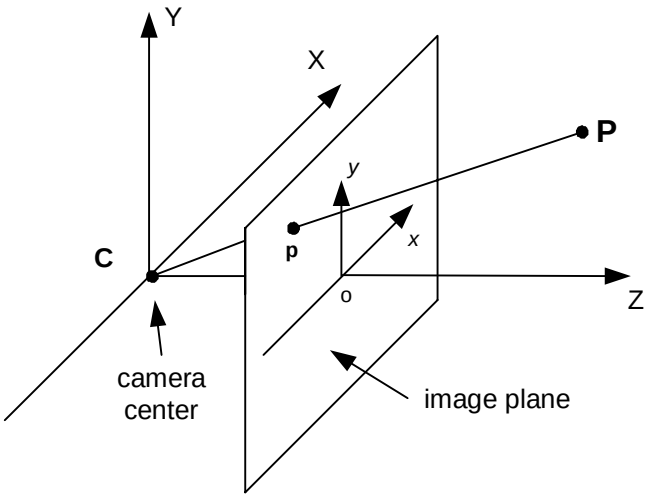


Figure 3: Pinhole Camera Geometry. **C** is the camera center and *o* is the principal point.

Suppose the coordinates of **P** are $(X, Y, Z)^T$ in the Euclidean 3D-space and **p** is $(x, y)^T$ in the Euclidean 2D-space (image plane), then

$$x = fX/Z \tag{3.1}$$

$$y = fY/Z \tag{3.2}$$

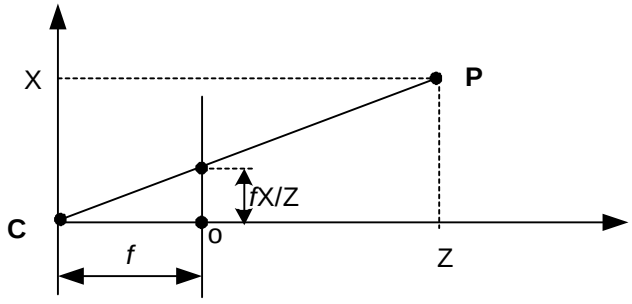


Figure 4: Mapping of Point **P** Into the Image Plane In the x Direction

If we use homogeneous coordinates to represent **P** and **p**, i.e. $\mathbf{P} = (X, Y, Z, 1)^T$ and $\mathbf{p} = (x, y, 1)^T$, we get:

$$\mathbf{p} = \begin{pmatrix} fX/Z \\ fY/Z \\ 1 \end{pmatrix} = \frac{1}{Z}\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} \tag{3.3}$$

Or alternatively

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.4}$$

we thus get

$$\mathbf{p} = \frac{1}{Z}\prod \mathbf{P} \tag{3.5}$$

where

$$\prod = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.6}$$

However, in practice, the origin of the camera is not located at the principal point. Therefore as shown in Figure 5, there is a coordinate system offset.

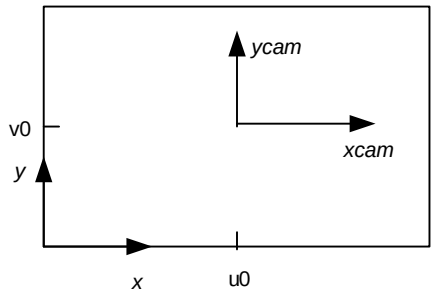


Figure 5: Image (x, y) and Camera (x_{cam}, y_{cam}) Coordinate System

So, after the projection, the point **P** is mapped to the point $\mathbf{p} = (x, y, 1)^T$ via

$$x = fX/Z + u_0 \tag{3.7}$$

$$y = fY/Z + v_0 \tag{3.8}$$

Or

$$\begin{pmatrix} fX + Zu_0 \\ fY + Zv_0 \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.9}$$

which can be written in matrix notation as

$$\mathbf{p} = \begin{pmatrix} fX/Z + u_0 \\ fY/Z + v_0 \\ 1 \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} fX + Zu_0 \\ fY + Zv_0 \\ Z \end{pmatrix} = \frac{1}{Z} \prod \mathbf{P} \tag{3.10}$$

where

$$\prod = \begin{bmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.11}$$

and

$$P = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.12}$$

Let

$$\mathbf{K} = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

which is the camera intrinsic (calibration) matrix.

Then

$$\prod = \mathbf{K[I \mid 0]} \tag{3.14}$$

$$\mathbf{p} = \frac{1}{Z}\prod \mathbf{P} = \frac{1}{Z}\mathbf{K[I \mid 0]P} \tag{3.15}$$

where [**I** | **0**] represents a matrix divided up into a 3×3 block(the identity matrix) plus a column zero vector.

The above discussion assumes that the origin of the 3D Euclidian space is located at the camera projection center. We call this coordinate system the camera coordinate frame. In general, the points in space are expressed in terms of the world coordinate frame. And these two frames are related via a rotation and a translation. Let $\mathbf{X}$ be the inhomogeneous 3-vector that represents the coordinate of a point in the world coordinate frame and $\mathbf{X}_{\mathbf{cam}}$ be the coordinate of the same point in the camera coordinate frame. As shown in Figure3.4, we have $\mathbf{X}_{\mathbf{cam}} = \mathbf{R}(\mathbf{X} - \mathbf{C})$, where $\mathbf{C}$ represents the coordinates of the camera center in the world coordinate frame, and $\mathbf{R}$ is a 3×3 rotation matrix representing the orientation of the camera coordinate frame. This equation may be written in homogeneous form as

$$\mathbf{P}_{\mathbf{cam}} = \begin{bmatrix} \mathbf{R} & -\mathbf{RC} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{RC} \\ 0 & 1 \end{bmatrix} \mathbf{P}_{\mathbf{w}} \tag{3.16}$$

Substituting (3.16) into (3.15), we get

$$\mathbf{p} = \frac{1}{Z}\mathbf{K}[\mathbf{I} \mid \mathbf{0}]\mathbf{P}_{\mathbf{cam}} = \frac{1}{Z}\mathbf{KR}[\mathbf{I} \mid -\mathbf{C}]\mathbf{P}_{\mathbf{w}} \tag{3.17}$$

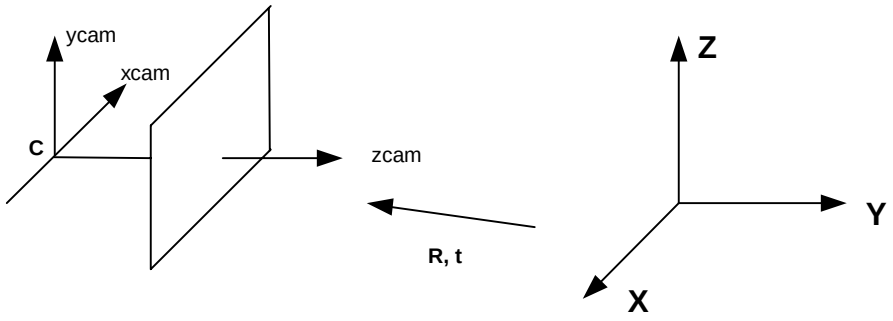


Figure 6: Euclidean Transformation Between World and Camera Coordinate Frames

3.2 View Morphing for Parallel Views

When reference images are two parallel views, linear interpolation of both views can generate shape-preserving in-between images.

Supposed I_0 and I_1 are two parallel images of the same object as shown in Figure 7. The focal length of I_0 and I_1 are f_0 and f_1 respectively. Also suppose the origin of the world coordinate frame is located in the camera center of I_0. And the camera center of I_1 is located in (Cx, Cy, 0) in the world coordinate system.

Based on equation (3.17): $\mathbf{p} = \frac{\mathbf{1}}{Z}\mathbf{KR[I \mid -C]P_w}$, we can map the point **P** from world coordinate frame into the image plane as follows:

For image plane I_0, since its camera center is located in the origin of the world coordinate frame, we get $\mathbf{C}_0 = \mathbf{0}$ and $\mathbf{R} = \mathbf{I}$. So,

$$\mathbf{p_0} = \frac{1}{Z}\mathbf{KI[I \mid -0]P} = \frac{1}{Z}\prod_0 \mathbf{P} \tag{3.18}$$

where

$$\prod_0 = \begin{bmatrix} f_0 & 0 & u_0 & 0 \\ 0 & f_0 & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.19}$$

For image plane I_1, we have $\mathbf{C}_1 = (C_x, C_y, 0)$ and $\mathbf{R} = \mathbf{I}$. So,

$$\mathbf{p_1} = \frac{1}{Z}\mathbf{KI[I \mid -C]P} = \frac{1}{Z}\prod_1 \mathbf{P} \tag{3.20}$$

where

$$\prod_1 = \begin{bmatrix} f_1 & 0 & u_0 & -f_1 C_x \\ 0 & f_1 & v_0 & -f_1 C_y \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Let $p_0 \in I_0$ and $p_1 \in I_1$ be projections of the scene point $\mathbf{P} = [X\ Y\ Z\ 1]^T$. Linear interpolation of $\mathbf{p}0$ and $\mathbf{p}1$ yields

$$(1-s)\mathbf{p_0} + s\mathbf{p_1} = (1-s)\frac{1}{Z}\prod_0 \mathbf{P} + s\frac{1}{Z}\prod_1 \mathbf{P}$$

$$= \frac{1}{Z} \prod_s \mathbf{P} \tag{3.21}$$

where

$$\prod_s = (1-s)\prod_0 + s\prod_1$$

$$= \begin{bmatrix} f_s & 0 & u_0 & -f_s \alpha_s C_X \\ 0 & f_s & v_0 & -f_s \alpha_s C_Y \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.22}$$

$$f_s = (1-s) f_0 + s f_1 \tag{3.23}$$

$$\alpha_s = \frac{s f_1}{(1-s) f_0 + s f_1} \tag{3.24}$$

From equation (3.21), we conclude that for parallel cameras shown in Figure 7, image interpolation produces a new view whose projection matrix Π_s is a linear interpolation of Π_0 and Π_1, representing a camera with focal length f_s and the perspective center Cs given by:

$$C_s = (\alpha_s C_x, \alpha_s C_{Y,} 0) \tag{3.25}$$

In other words, interpolating images from parallel cameras produces images that correspond to moving a camera on the line C_0C_1 between the two camera centers and zooming continuously. Because a new view for the same object can be produced, this interpolation can be seen as shape-preserving.

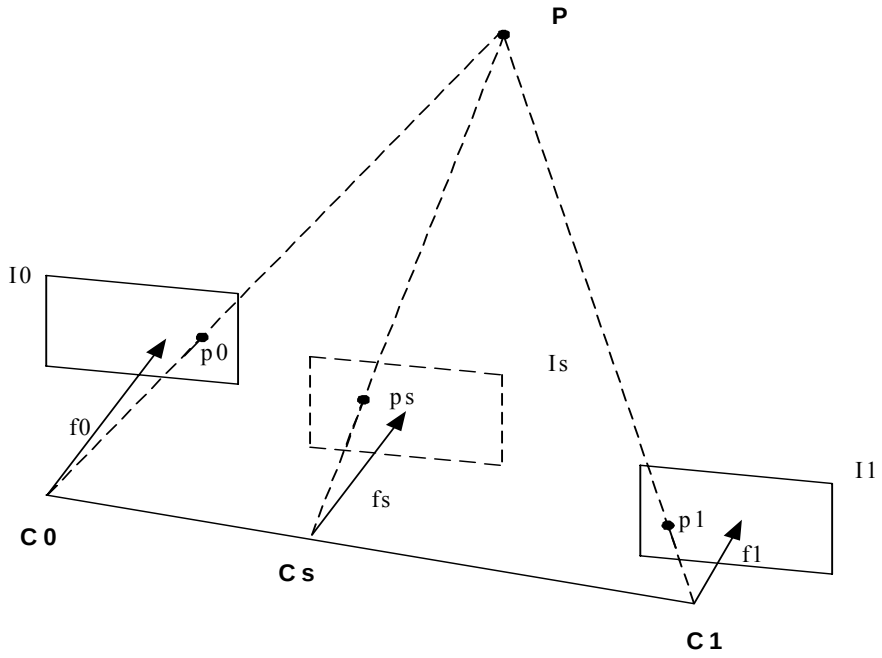


Figure 7: Linear Interpolations of Corresponding Pixels In Parallel Views. Image planes I_0 and I_1 creates image I_s, which represents another parallel view of the same scene.

3.3 View Morphing for Non-parallel Views

As discussed above in Chapter 2, direct application of view morphing to non-parallel views will lead to shape-distortions. The solution to this problem is therefore to first transform the two non-parallel views into two parallel ones, and then apply the morphing process.

Let I_0 and I_1 be two perspective views of the same object **P**. Let also the camera centers be located in $\mathbf{C_0}$ and $\mathbf{C_1}$, respectively. Based on equation (3.17): $\mathbf{p} = \frac{1}{Z}\mathbf{KR[I \mid -C]P_w}$, and we can map the point **P** from world to the image planes I_0 and I_1 using:

$$\mathbf{p_0} = \frac{1}{Z}\mathbf{K}\mathbf{R_0}[\mathbf{I} \mid -\mathbf{C_0}]\mathbf{P_w} \tag{3.26}$$

$$\mathbf{p_1} = \frac{1}{Z}\mathbf{K}\mathbf{R_1}[\mathbf{I} \mid -\mathbf{C_1}]\mathbf{P_w} \tag{3.27}$$

Let $\mathbf{H_0} = \mathbf{KR_0}$ and $\mathbf{H_1} = \mathbf{KR_1}$. The projection matrices $\mathbf{\Pi_0}$ and $\mathbf{\Pi_1}$ can be written as:

$$\mathbf{\Pi_0} = \mathbf{KR_0}[\mathbf{I} \mid -\mathbf{C_0}] = [\mathbf{H_0} \mid -\mathbf{H_0C_0}] \tag{3.28}$$

$$\mathbf{\Pi_1} = \mathbf{KR_1}[\mathbf{I} \mid -\mathbf{C_1}] = [\mathbf{H_1} \mid -\mathbf{H_1C_1}] \tag{3.29}$$

As shown in Figure 8, view morphing uses the following 3-step procedure to generate in-between shape-preserving images I_s with camera center $\mathbf{C_s}$ located on the line joining the camera centers for the reference images, i.e. on $\mathbf{C_0C_1}$.

1. Prewarp: Apply projective transform $\mathbf{H_0^{-1}}$ to I_0 and $\mathbf{H_1^{-1}}$ to I_1, to generate parallel views I_{0w} and I_{1w}.
2. Morph: Form I_{sw} by linearly interpolating positions and colors of corresponding points in I_{0w} and I_{1w}
3. Postwarp: Apply $\mathbf{H_s}$ to I_{sw}, to get the in-between image I_s

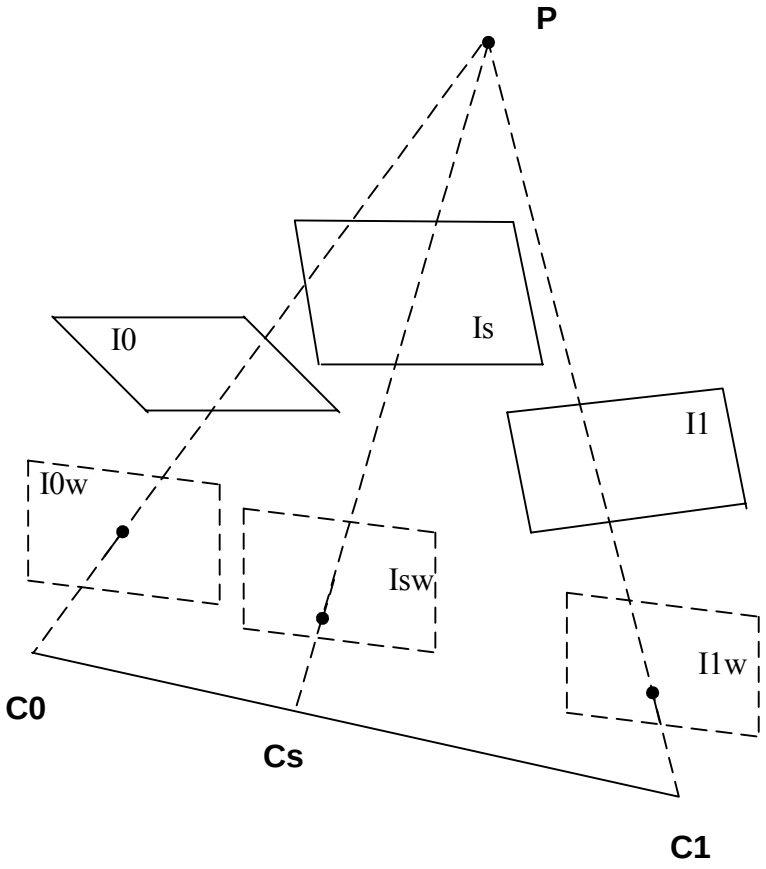


Figure 8: View Morphing for Non-parallel Views.

After the prewarping, the projection matrices of the prewarped images I_{0w} and I_{1w} have the following formats:

$$\mathbf{\Pi_{0w}} = \mathbf{H_0}^{-1} [\mathbf{H_0} \mid -\mathbf{H_0 C_0}] = [\mathbf{I} \mid -\mathbf{C_0}] \tag{3.30}$$

$$\mathbf{\Pi_{1w}} = \mathbf{H_1}^{-1} [\mathbf{H_1} \mid -\mathbf{H_1 C_1}] = [\mathbf{I} \mid -\mathbf{C_1}] \tag{3.31}$$

This rectifies the two reference images so that the corresponding points in the two images appear along the same scan line.

The projection matrix of I_s can be written as $\mathbf{\Pi_s} = [\mathbf{H_s} \mid -\mathbf{H_s C_s}]$, where $\mathbf{C_s}$ can be calculated using equation (3.25).

Generally speaking, prewarping brings the image planes into alignment without changing the optical centers of the two cameras; morphing the prewarped images moves the optical canter to $\mathbf{C_s}$; and finally postwarping transforms the image plane of the new view to its desired position and orientation.

3.4 View Morphing for Noncalibrated Views

When reference images are uncalibrated, it is still possible to use the 3-step algorithm described above to generate in-between images. The following sections describe the details.

3.4.1 Prewarping Uncalibrated Images

The purpose of prewapring of uncalibrated images is as before to make two reference images parallel to each other so that the corresponding points appear along the same scanlines.

In the uncalibrated case we need to find two 2D projective transformations $\mathbf{H_0}^{-1}$ and $\mathbf{H_1}^{-1}$ that would rectify I_0 and I_1, respectively. Because the prewarped images are rectified, it can be shown that the fundamental matrix for I_{0w} and I_{1w} is given by [4]:

$$\mathbf{F_w} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.32}$$

Based on the epipolar geometry, a point $\mathbf{p_{0w}}$ in I_{0w} and its corresponding point $\mathbf{p_{1w}}$ in I_{1w} have the following relationship:

$$\mathbf{p_{1w}}^{\mathbf{T}}\, \mathbf{F_w}\, \mathbf{p_{0w}} = 0 \tag{3.33}$$

Substituting $\mathbf{p_{0w}}$ with $\mathbf{H_0}^{-1}\mathbf{p_0}$ and $\mathbf{p_{1w}}$ with $\mathbf{H_1}^{-1}\mathbf{p_1}$, we get:

$$\mathbf{p_1}^{\mathbf{T}}\mathbf{H_1}^{-\mathbf{T}}\mathbf{F_w}\mathbf{H_0}^{-1}\mathbf{p_0} = 0$$

Since $\mathbf{p_1}^{\mathbf{T}}\mathbf{F}\mathbf{p_0} = 0$,

$$\mathbf{H_1}^{-\mathbf{T}}\mathbf{F_w}\mathbf{H_0}^{-1} = \mathbf{F} \tag{3.34}$$

i.e.,

$$\mathbf{H_1}^{\mathbf{T}}\mathbf{F}\mathbf{H_0} = \mathbf{F_w} \tag{3.35}$$

So, the prewarping can be solved if we can find a pair of homographies $\mathbf{H_0}$ and $\mathbf{H_1}$ that satisfy Equation (3.35). The following procedure can be used to find these two homographies, which basically rotate the image planes to obtain parallel views and then apply 2D affine transformations to align corresponding scanlines.

We therefore need to first compute the fundamental matrix **F** using for instance the well-known 8-point algorithm [3]:

Once the Fundamental matrix is known, one can find the epipoles of the two images using the following equations [4]:

$$\mathbf{F}\mathbf{e_0} = \mathbf{0} \tag{3.36}$$

$$\mathbf{F}^{\mathbf{T}}\mathbf{e_1} = \mathbf{0} \tag{3.37}$$

Given the epipoles the two images can then be made parallel using the following approach. Let **E** be a plane parallel to $\mathbf{C_0C_1}$, suppose **E** intersects the image plane I_i at $\mathbf{d_i}$, the rotation of I_i about $\mathbf{d_i}$ will make the two image planes parallel. Alternatively rotating I_i about any line parallel to $\mathbf{d_i}$ will also make image planes parallel to each other.

Suppose **E** intersects I_0 at $\mathbf{d_0}$, that passes through the image center of I_0: $\mathbf{d_0} = [-d_0^y \; d_0^x \; 0]^{\mathrm{T}}$. Point **p** on $\mathbf{d_0}$ has the form $\mathbf{p} = [sd_0^x \; sd_0^y \; 0]^{\mathrm{T}}$ and satisfy the equation

$$\mathbf{d_0}^{\mathbf{T}}\mathbf{p} = \mathbf{0} \tag{3.38}$$

Because the epipoles of the image planes after rectification are located at infinity, the new epipole for I_0 after rotating about $\mathbf{d_0}$ has the form

$$\mathbf{e_{0N}} = \mathbf{R}_{\theta_0}^{d_0}\mathbf{e_0} \tag{3.39}$$

$$= [e_0^x, e_0^y, 0]^{\mathrm{T}} \tag{3.40}$$

Where the rotation matrix is given by

$$\mathbf{R}_{\theta_0}^{d_0} = \begin{bmatrix} (d_0^x)^2 + (1-(d_0^x)^2)\cos\theta_0 & d_0^x d_0^y (1-\cos\theta_0) & d_0^y \sin\theta_0 \\ d_0^x d_0^y (1-\cos\theta_0) & (d_0^y)^2 + (1-(d_0^y)^2)\cos\theta_0 & -d_0^x \sin\theta_0 \\ -d_0^y \sin\theta_0 & d_0^x \sin\theta_0 & \cos\theta_0 \end{bmatrix} \quad (3.41)$$

Substituting (3.40) and (3.41) into (3.39), we get:

$$\theta_0 = tan^{-1}\left(\frac{e_0^z}{d_0^y e_0^x - d_0^x e_0^y}\right) \quad (3.42)$$

In order to minimize the rotation angle θ_0, we can chose $d_0^x = \alpha e_0^y$, $d_0^y = -\alpha e_0^x$, where $\alpha = \frac{1}{\sqrt{(e_0^x)^2 + (e_0^y)^2}}$. So the line $\mathbf{d_0} = [-d_0^y \; d_0^x \; 0]^T = \alpha[e_0^x \; e_0^y \; 0]$.

Let **E'** be an epipolar plane parallel to **E**. **E'** intersects I_i in an epipolar line $\mathbf{l_i}$ parallel to $\mathbf{d_i}$. Because they are parallel, $\mathbf{l_0}$ and $\mathbf{d_0}$ intersect at the ideal point $\mathbf{i_0} = [\, d_0^x \quad d_0^y \quad 0\,]^T$. Since $\mathbf{i_0}$ is on the epipolar line $\mathbf{l_0}$, we can get the epipolar line $\mathbf{l_1}$ using the following equation:

$$\mathbf{l_1} = \mathbf{F i_0} \quad (3.43)$$

Let $\mathbf{d_1}$ be the line passing through the image origin of I_1 and parallel to $\mathbf{l_1}$. A rotation of I_1 about $\mathbf{d_1}$, which makes I_1 parallel to **E'** will also make it parallel to **E**. Accordingly, if $[x \; y \; z]^T = \mathbf{F i_0} =$

$\mathbf{F}[\, d_0^x \quad d_0^y \quad 0\,]^T$, then the rotation axis is $\mathbf{d_1} = \alpha[x \;\; y \;\; 0]^T$, i.e. $d_1^x = \alpha y$ and $d_1^y = -\alpha x$, where $\alpha = \frac{1}{\sqrt{x^2 + y^2}}$.

After aligning the image planes, the two image planes are parallel to each other with the new epipole $\mathbf{e_{0N}} = \mathbf{R}_{\theta_0}^{d_0}\,\mathbf{e_0}$ of the form $\mathbf{e_{0N}} = [e_0^x \;\; e_0^y \;\; 0]^T$. The next step is to rotate the images about the z-axis so that the epipolar lines become horizontal. i.e. the epipole will be of the form $\mathbf{e_{0N}} = \alpha\,[1 \;\; 0 \;\; 0]^T$. The rotations are given by

$$\phi_i = -tan^{-1}(e_0^y / e_0^x) \tag{3.44}$$

$$\mathbf{R_{\phi i}} = \begin{bmatrix} \cos\phi_i & -\sin\phi_i & 0 \\ \sin\phi_i & \cos\phi_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.45}$$

After applying these image plane rotations, the fundamental matrix will have the form:

$$\mathbf{F_w} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & a \\ 0 & 1 & b \end{bmatrix} \tag{3.46}$$

In order to make $\mathbf{F_w}$ of the form $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$, the second image should be vertically scaled and translated by the matrix

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -a & -b \\ 0 & 0 & 1 \end{bmatrix} \tag{3.47}$$

Therefore the two prewarp transforms $\mathbf{H_0}^{-1}$ and $\mathbf{H_1}^{-1}$ are given by

$$\mathbf{H_0}^{-1} = \mathbf{R}_{\phi 0}\, \mathbf{R}_{\theta_0}^{d_0} \tag{3.49}$$

$$\mathbf{H_1}^{-1} = \mathbf{T}\, \mathbf{R}_{\phi 1}\, \mathbf{R}_{\theta_1}^{d_1} \tag{3.50}$$

3.4.2 Morph the Prewarped Images

Since the images are now parallel and corresponding points appear in the same scanline, the morphing process is simply achieved by applying a linear interpolation.

3.4.3 Specifying Postwarps

Postwarping transforms the image plane to its desired position and orientation. From this viewpoint, postwarping is a projective transformation $\mathbf{H_s}$ that transforms I_{sw} to I_s. Let

$$\mathbf{H_s} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{3.51}$$

and suppose $\mathbf{p_w} = [x_w\ y_w\ 1]^T$ and $\mathbf{p_s} = [x_s\ y_s\ 1]^T$ are one pair of corresponding points in I_{sw} to I_s respectively. We have $c\mathbf{p_s} = \mathbf{H_s p_w}$, where c is a scale factor. Eliminating c yields two linear equation for one pair of $\mathbf{p_w}$ and $\mathbf{p_s}$.

$$x_s(h_{31}x_w + h_{32}y_w + h_{33}) - (h_{11}x_w + h_{12}y_w + h_{13}) = 0 \qquad (3.52)$$

$$y_s(h_{31}x_w + h_{32}y_w + h_{33}) - (h_{21}x_w + h_{22}y_w + h_{23}) = 0 \qquad (3.53)$$

Using 4 pair of such corresponding points, we can get 8 linear equations in terms of the components of the matrix $\mathbf{H_s}$. These equations can be written in homogeneous form as $\mathbf{Ah} = 0$, where $\mathbf{A}$ is the coefficient matrix and $\mathbf{h} = [h_{11}\ h_{12}\ h_{13}\ h_{21}\ h_{22}\ h_{23}\ h_{31}\ h_{32}\ h_{33}]^T$. The solution is then given by the unit eigenvector of $\mathbf{A^T A}$ corresponding to the smallest eigenvalue. By adding the constraint $\mathbf{h_{33}} = 1$, we can get the final solution for $\mathbf{H_s}$.

Therefore the postwarping can be done as follows: we first specify the paths of at least four image points through the entire morph transition; for each in-between image I_s, we specify the position of these control points in I_s; we then find the corresponding position of the control points from the morphed image $I_{sw;}$ the positions of the control points in I_s and I_{sw} specify a homogeneous linear system of equations whose solution yields $\mathbf{H_s}$; Apply $\mathbf{H_s}$ to I_{sw} will yield the in-between image I_s.

3.5 Issues in View Morphing

The prewarping transformation relies heavily on the fundamental matrix. In order to get a stable solution for the fundamental matrix, it is important to choose a reliable subset of feature points. Degeneration of feature points (e.g., coplanar features) should be avoided. In addition, better results are obtained when feature points are well distributed throughout the pair of images.

During the prewarping and postwarping, images are transformed by applying transformation matrices. During this process, some pixels in the destination may not get painted. One approach to solve this problem it is to use reverse mapping, which goes through the destination image pixel-by-pixel, and samples the correct pixel from the source image. The most important feature of reverse mapping is that every pixel in the destination image gets an appropriate value.

3.6 Experimental Result and Analysis

Compared to field morphing, view morphing can generate a shape-preserving in-between image using two reference images, taken from different viewpoints from the same object. The most impressive part of view morphing is the prewarping procedure. Using epipolar geometry, a plane that is parallel to the line joining the camera centers can be found. Then the reference images are rotated so that they are made parallel to this plane. This also makes them parallel to each other. Since linear interpolation of two parallel views can generate shape-preserving in-between images, new high-quality images can be artificially synthesized.

Prewarpimg and postwarping introduce more image resampling operations than field morphing, which may lead to noticeable blurring in the in-between images. This is particularly true when low-resolution images are used as reference images. The example in Figure 9 demonstrates these effects: $I_{0.5}$ is the in-between image of I_0 and I_1 generated using view morphing. The blurs can be found in each line in $I_{0.5}$. And the letters" Pentium 4 Processor", which are clear in both I_0 and I_1, become blurred in $I_{0.5}$.

I_0 $I_{0.5}$ I_1

Figure 9: View Morphing of A Box: left and right: reference images, middle: the synthesized image.

Sometimes the prewarping procedure may not work, i.e., the two images can not be made parallel. One situation is when the optical center of one camera is within the field of view of the other. In the parallel configuration, each camera's optical center is out of the field of view of the other. Since the image reprojection does not change a camera's field of view, it can not make the optical center out of the field of view of the other if it is already located inside in the reference image.

The other case where prewarping would fail is when the epipole of the image is located inside the image. In prewarping (rectification) stage the epipole, is projected to infinity. Therefore, since the size of the prewarped image can't be infinite, this point can not be visible in the prewarped image. The points outside the epipole will also become invisible. So only part of the image will be seen in the prewarped image. The best results are obtained when visibility is nearly constant, i.e., when most scene points are visible in both reference views. Occlusion is another source of problem, which can cause ghosting effects, due to the cross-dissolve, i.e. unmatched points will appear at fractional-intensity in in-between views.

CHAPTER 4. MORPHING BETWEEN DIFFERENT EXPRESSIONS

4.1 Problem Description

Suppose we are given four reference images: one pair from left and right with the mouth closed (I_{lc}, I_{rc}), and another pair from approximately the same positions, but with the mouth open ($I_{lo,}$ I_{ro}). Is it possible to generate in-between images for I_{lc} and I_{ro} or alternatively in-between images for I_{rc} and I_{lo}, i.e., is it possible to create a morphing animation that this person turns his head from left to the right and moves his mouth gradually at the same time?

4.2 Problem Analysis

Let's discuss one case of the problem, i.e. generating morphing images between I_{lc} and I_{ro}. Unlike the problem of view morphing, in which only the camera moves from left to right, the object in this case also moves a lot: the mouth from the closed position in I_{lc} moves to the open position in I_{ro}. Moreover, the moving of the object is not rigid: the moving quantities are different for different parts of the object. The hair of the person is almost at the same position in both images while the mouth and the chin move a lot. So this problem is out of the range of view morphing.

Applying field morphing can't solve this problem either. The cameras in both images are far apart from each other. As discussed above, field morphing is suitable for morphing the person's

expression changes in the same viewpoint. Changing orientation of the camera will cause distortion effects for field morphing in the in-between images.

4.3 Our Algorithm

Our algorithm is developed by combining field morphing and view morphing: it takes advantage of field morphing's ability to morph one expression to the other in the same viewpoint and view morphing's ability to morph the same expression from different viewpoints. The detailed steps are as follows:

4.3.1 Prewarp I_{lc} and I_{rc}

In I_{lc} and I_{rc}, the camera is facing the person from different positions and orientations, while the pose of the person is kept constant. So it is easy to apply the prewarping procedure of the view morphing to make the two images parallel and generate parallel images I_{lcw} and I_{rcw}.

4.3.2 Generate in-between images for I_{lcw} and I_{rcw}

We use field morphing to generate in-between images of I_{lcw} and I_{rcw}. However, occlusion is a major issue in this step: As the face turns around the left side may be visible in I_{lcw} but occluded in I_{rcw} and vice versa. . Occlusions cause a disturbing effect referred to as "ghosting", where occluded parts appear as fading in or out during rendering. We solve this problem by relaxing the

monotonicity assumption of the feature points along the epipolar lines to piece-wise monotonicity.

For this purpose, we specify the boundaries of left and right homogeneous regions as feature lines. This essentially leads to segmented reference images where each segment is assumed to preserve monotonicity of the feature points along the epipolar lines. Therefore interpolation can be performed without error in each segmented using only the boundaries and the feature lines that lie inside the region. Segmented regions are labeled for book keeping, so that if a feature line is crossing over multiple regions, only the segments inside each region are used for morphing within that region.

In this thesis the segmentation was done by user interaction as follows:

- We segmented the reference images into three types of regions: 1st type of regions are the ones that have almost the same visibilities in both images, such as eyes in our experimentations; 2nd type of regions are the ones that have more visibility in one image than the other, e.g. certain parts of the face; 3rd type of regions are the ones that are only visible in one image, i.e. occluded in the other one.
- Regions that are visible only in one image are segmented in that image and then the boundaries of these regions are projected in the second image.

After segmentation, all feature lines are selected, some of which may overlap between several regions. Segmentation allows processing and interpolating each region individually based on

their visibilities. For the 1st type of region, since their visibilities in both images are almost the same, we can use the same method of field morphing to interpolate feature lines and get pixels of in-between image using cross-dissolving the pixels from both images. For the 2nd type of region, since the shape of the area change dramatically in both images (from big area to small area or vice versa), only pixels from the most visible image are used. For the 3rd type of the region, since the information is only available in one image, we only use the pixels from that image.

After specifying the feature lines, we can calculate their positions in the in-between images using field morphing. Depending on the position of in-between images I_{cw}(i) (i =0.1, 0.2,… , 0.9), the position of each feature line in I_{cw}(i) is readily computed by linear interpolation of their endpoints.

Then for each pixel $\mathbf{p_{cw}}$ in the in-between image I_{cw}(i) (i =0.1, 0.2,… , 0.9), we calculate it's mapping pixels $\mathbf{p_{lcw}}$ in I_{lcw} and $\mathbf{p_{rcw}}$ in I_{rcw} using field morphing. However, the morphing is done based on segmentation: a pixel $\mathbf{p_{cw}}$ is morphed based only on the feature lines inside the segmented area where the pixel resides..

Once the geometric interpolation is performed, pixel color is selectively assigned based again on segmentation, i.e. cross-dissolve if the region is the 1st type, and use only the color of visible (or more visible) region if the region is of the other two types.

The approach proposed above eliminates all ghosting effects. However, it introduces problems at the region boundaries: due to segmentation seam may occur at the boundaries. However, this problem is fairly easy to solve by using a blending technique, which would yield a more smooth transition between neighboring regions.

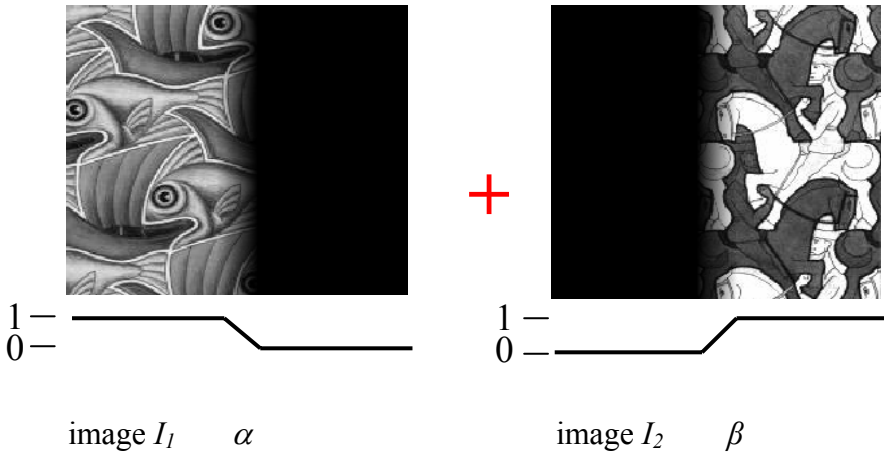


Figure 10: Using Feathering Method to Blend Two Images

The blending is done as follows. Suppose regions A and B are neighboring regions, such that the pixels of region A are from I_{lcw} and the pixels of region B are from I_{rcw}. We use feathering method to blend their boundary as shown in Figure 10. This is done by first, specifying a small

window (a small boundary band) as the transition area, where the colors are linearly interpolated between the two regions within the window.

4.3.3 Postwarp the In-between Images

For each in-between image I_{cw}(i) (i =0.1, 0.2,… , 0.9), we can get in-between images I_c(i) (i =0.1, 0.2,… , 0.9) by applying the postwarping procedure of view morphing.

4.3.4 Generate In-between Images I_o(i) (i =0.1, 0.2,… , 0.9) for I_{lo} and I_{ro}

Apply the same step of **4.3.1** to **4.3.3** on I_{lo} and I_{ro} and generate In-between Images I_o(i) (i =0.1, 0.2,… , 0.9) for them.

4.3.5 Generate In-between Images *I*(a) (a =0.1, 0.2,… , 0.9)

At this point, we have in-between close-mouth images I_c(i) (i =0.1, 0.2,… , 0.9) and in-between open-mouth images I_o(i) (i =0.1, 0.2,… , 0.9). We can take advantage of field morphing's ability to morph different expression in the same viewpoint. When I_c(i) and I_o(i) (i =0.1, 0.2,… , 0.9) are at the same position, it is possible to generate in-between images for them: for each pair of I_c(i) and I_o(i) (i =0.1, 0.2,… , 0.9), we generate their in-between images I(i) = I_a. This would allow us to include expression changes while the head is rotating from left to right..

4.4 Experimental Results

The reference images that we used are shown in Figure 11. I_{lc} and I_{lo} were taken approximately in the same orientation but with different expressions. Similarly, I_{rc} and I_{ro} were taken approximately from the same orientation with different expressions.

I_{lc} I_{rc} I_{lo} I_{ro}

Figure 11: Reference Images

The results after prewarping are shown in Figure 12. For prewarping, we manually chose 12 corresponding points for each pair of images in order to calculate the fundamental matrix and hence the homographies as described earlier.

I_{lcw} I_{rcw} I_{low} I_{row}

Figure 12: Prewarped Images

Figure 13: Segmentation of prewarped open mouth images

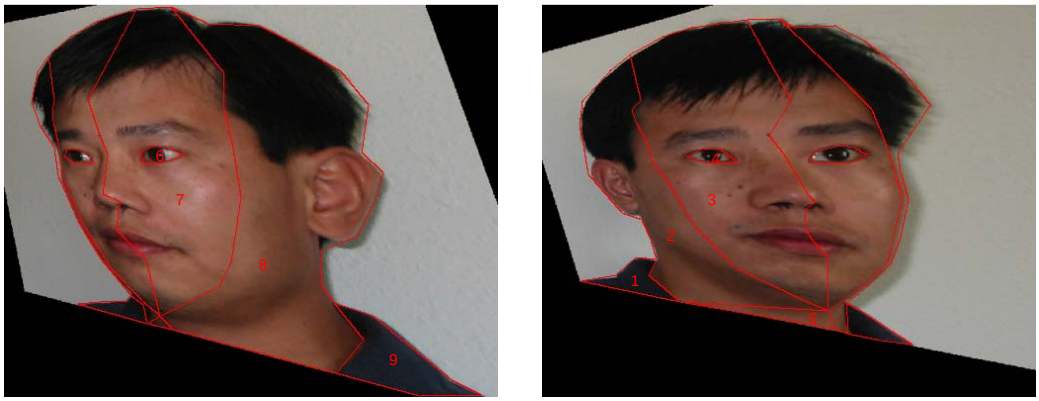


Figure 14: Segmentation of Prewarped Close Mouth Images

After prewarping the images are segmented into several regions. In this case the open mouth images were segmented into 12 regions. The segmented images are shown in Figure 13. Table 1 shows the reference image used for each region in the in-between images. The prewarped close mouth images were segmented into 10 regions as shown in Figure 14. Table 2 gives the reference image used for each part in the in-between images.

Table 1: Regions and Their Reference Image for the Prewarped Close Mouth Images

Region Number	Face Part	Reference Images
1	Right shoulder	Left, right
2	Right neck	Right
3	Right side face	Right
4	Right front face	Right
5	Right eye	Left, right
6	Middle neck	Right
7	Left eye	Left, right
8	Left front face	Left
9	Left side face	Left
10	Left neck	Left
11	Left shoulder	Left
12	Mouth	Left, right

Table 2: Regions and Their Reference Image for the Prewarped Open Mouth Images

Region Number	Face Part	Reference Images
1	Right should	Right
2	Right neck + right side face	Right
3	Right front face	Right
4	Right eye	Right, left
5	Middle neck	Right
6	Left eye	Left, right
7	Left front face	Left
8	Left side face + left neck	Left
9	Left shoulder	Left

The regions were then interpolated as described above. The results after the postwarping are shown in Figure 15. There is practically no ghosting effect in the images and area transitions are also very smooth, making the rendering appear very realistic. .

After generating in-between images for each pair of I_c(i) and I_o(i) (i =0.1, 0.2, … ,0.9), we can get the animation of the same person turn his head from right to the left while at the same time opening his mouth gradually. The results are shown in Figure 16.

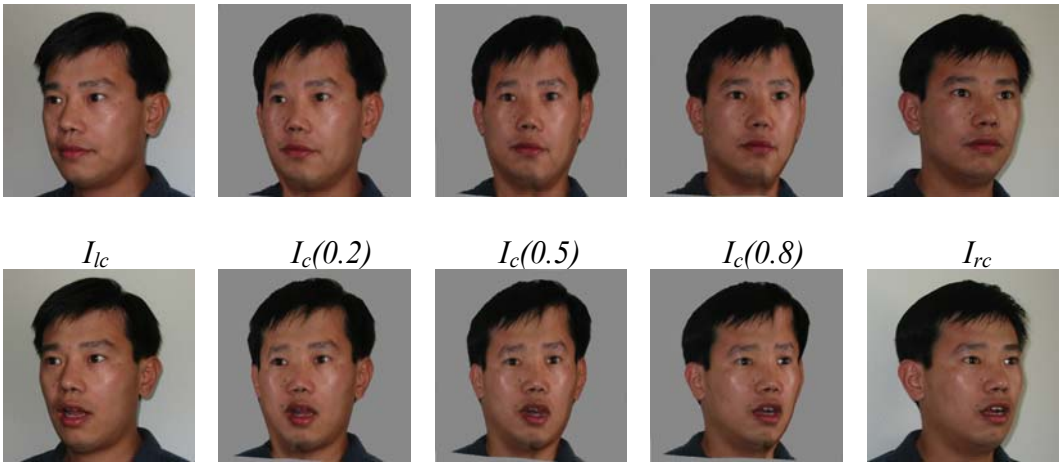


Figure 15: Synthesized In-between Images of Different Expressions. Top: in-between images for the close mouth images. Bottom: in-between images for the open mouth images.

Figure 16: Synthesized Finial In-between Images

4.5 Extensions to Other Scenarios

In addition to the above experimentations that led to excellent results, we considered other possible scenarios, and extensions of our algorithm. In particular we considered view changes around 180 degrees and also view extrapolation based on the reference images.

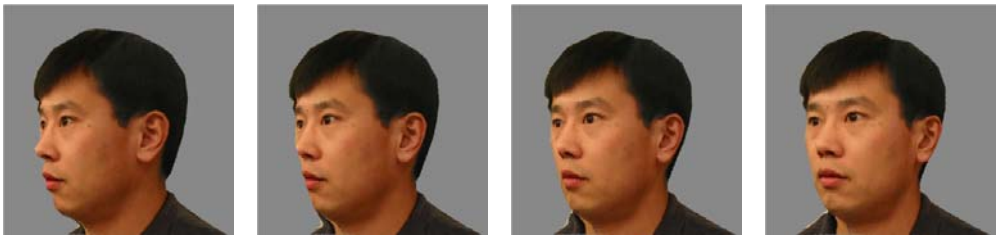
4.5.1 Morphing with Head Turning by 180 Degrees

This experimentation was aimed to extend the work described above and to determine how far segmentation can help to handle occlusions. The goal was to synthesize an animation where a person would be turning his/her head from left to right by 180 degrees. The reference images used for this experimentations are shown in Figure 17:

I_1 I_2 I_3 I_4

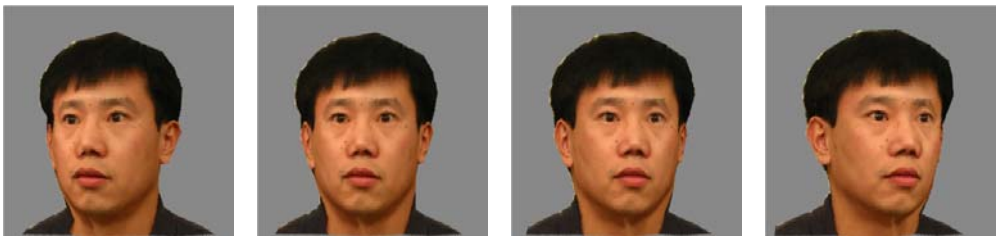
Figure 17: Reference Images For Morphing Head Turning Dy 180 Degrees

Some of the in-between images are shown in Figures 18, 19, and 20.



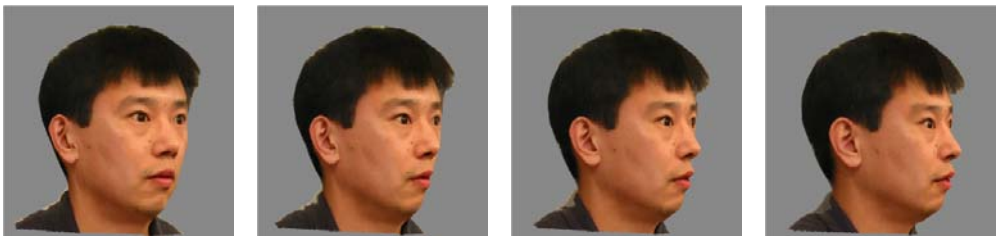
$I_{1.2}$ $I_{1.4}$ $I_{1.6}$ $I_{1.8}$

Figure 18: In-between Images For I_1 And I_2



$I_{2.2}$ $I_{2.4}$ $I_{2.6}$ $I_{2.8}$

Figure 19: In-between Images For I_2 And I_3



$I_{3.2}$ $I_{3.4}$ $I_{3.6}$ $I_{3.8}$

Figure 20: In-between images for I_3 and I_4

These images demonstrated the clear advantage of our approach in handling regions with occlusions. As shown in I_1 and I_2, most scene points of right face were only visible in I_2, which made view morphing very difficult to implement.

The animations we created showed smooth transitions from I_1 to I_2, I_2 to I_3, and I_3 to I_4. But when we connected these animations together and made a transition from I_1 to I_4 directly, we found jumps in the process. As shown in Figure 21, noticeable jumps can be found when $I_{1.98}$ goes to $I_{2.02}$ and $I_{2.98}$ goes to $I_{3.02}$.

$I_{1.98}$ $I_{2.02}$ $I_{2.98}$ $I_{3.02}$

Figure 21: Neighbor Images

The reason for these jumps is that the pixels of the same parts of the neighboring image frames ($I_{1.98}$ and $I_{2.02}$, $I_{2.98}$ and $I_{3.02}$) were from different reference images. Although, original images were taken in the same lighting situation using the same camera, the color difference of the left side face can also be noticed in I_1 and I_2, which led to a "jumping" effect from $I_{1.98}$ to $I_{2.02}$. We suggest that this problem can be solved by developing a temporal blending similar to spatial blending the removes seams.

4.5.2 Extrapolation

Here we tried to see if we could generate outside views from the given reference images. We used the same reference images shown in Figure 22. .

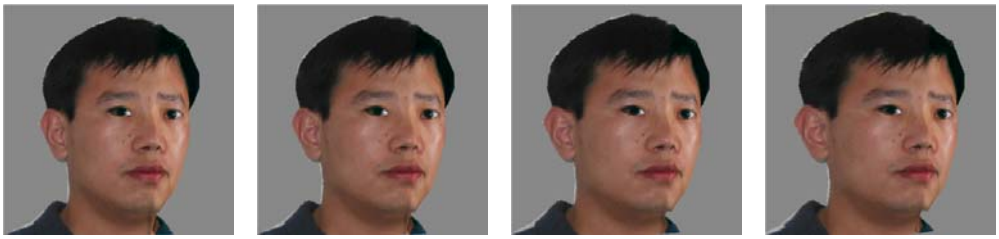
I_0 I_1

Figure 22: Reference Images For Extrapolation

When generating in-between images, the feature points were interpolated from starting positions in I_0 to ending positions in I_1 gradually. Suppose x coordinate of point $\boldsymbol{p}$ moves from x_0 in I_0 to x_1 in I_1, let $d = x_1 - x_0$, the x coordinate of $\boldsymbol{p}$ is just $x_0 + i*d$ in each in-between image I_i ($i = 0.1, 0.2, \ldots$). During the extrapolation, we let $i = -0.1, -0.2, \ldots$ or $i = 1.1, 1.2, \ldots$ so that generated images were located outside the range of I_0 and I_1. Figure 23 and Figure 24 are some of our results:

$I_{-0.05}$ $I_{-0.1}$ $I_{-0.15}$ $I_{-0.2}$

Figure 23: Extrapolated Images Outside I_0



$I_{1.05}$ $I_{1.1}$ $I_{1.15}$ $I_{1.2}$

Figure 24: Extrapolated Images Outside I_1

The results have considerable amount of distortions, especially when the generated images are far from the original ones. The reason is because the positions of feature lines in above images were extrapolated and could not reflect the real positions. For example, the right front face should be hidden gradually when images move toward outside I_0 in real circumstance. While from I_1 to I_0, this region only changes its size rather than having hiding effects.

This example shows the fact that morphing algorithms rely heavily on accurate boundaries. In particular, surface and texture discontinuities represent the strongest boundaries. Most of our feature lines were along these boundaries. Without knowing the accurate positions of them (like in this example), it's difficult to generate realistic morphing results.

4.6 Analysis of the Results

Occlusion has been the most challenging problem in both view morphing and field morphing. However, in addition to occlusions, the ghosting can also be caused by some unforeseen

combination of the specified line segments as shown in Figure 25. An important aspect of our algorithm is that it successfully eliminates these problems.

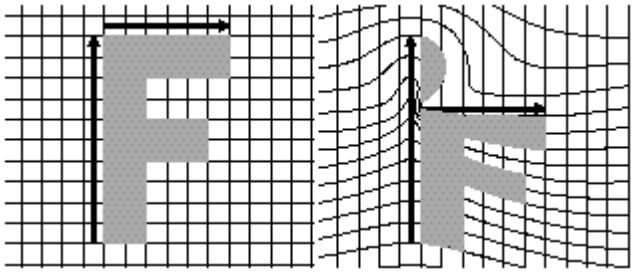


Figure 25: Moving the Horizontal Feature Line Down Creates Ghosting Above the Line

$I_{cw0.5}$ $I_{ow0.5}$

Figure 26: Morphing Results Without Segmentation. Applying view morphing and field morphing directly without segmentation. $I_{cw0.5}$: prewarped in-between image for close mouth images; $I_{ow0.5}$: prewarped in-between image for open mouth images.

Both types of problems can be seen in the Figure 26, which are in-between images generated using field morphing and view morphing without segmentation. Both problems lead to ghosting effects leading to unrealistic rendering of those parts of the image.

Figure 27 are in-between images using piece-wise processing of the 2D image features based on segmented images. However, each part in the in-between image is the result of cross-dissolving both images. Although the artifacts due to field morphing are eliminated, the fading of colors are still disturbing and unrealistic. In addition, to the ghosting effects on both sides of the face, the nose does not appear realistic due to partial occlusions.

$I_{cw0.5}$ $I_{ow0.5}$

Figure 27: Cross-dissolving After Segmentation. $I_{cw0.5}$: prewarped in-between image for close mouth images; $I_{ow0.5}$: prewarped in-between image for open mouth images.

The problem is readily solved by selectively interpolating pixel colors based on the three types of regions described above, i.e. based on the visibility of the segmented regions. Figure 28 shows the results of in-between images that use our algorithm without boundary blending. After boundary blending, we get the prewarped in-between images, which are shown in Figure 29.

$I_{cw0.5}$ $I_{ow0.5}$

Figure 28: Prewarped Views Before Boundary Blending

$I_{cw0.5}$ $I_{ow0.5}$

Figure 29: Finial Prewarped In-between Views.

CHAPTER 5. CONCLUSION

In this thesis we investigated the problem of expression morphing. The goal was to synthesize new views of human faces with expression changes based on reference images.

5.1 Contributions

One of the contributions of our work is proposing a new framework to solve expression morphing. This framework integrates field morphing and view morphing. It takes advantage of field morphing's ability to morph one expression to the other from the same viewpoint, and view morphing's ability to morph the same expression from different viewpoints. Based on four reference images we successfully generate the morphing from one viewpoint with one expression to another viewpoint with a different expression.

The other contributions of our work is proposing a new approach to eliminate artifacts that frequently occur in view morphing due to occlusions and in field morphing due to some unforeseen combination of feature lines. We propose to solve these problems by relaxing the monotonicity assumption to piece-wise monotonicity along the epipolar lines. For this purpose, we segment the object into several areas and divide it into labeled regions. According to the label of each region, the pixels of that region can be mapped from one of the reference images or from cross-dissolve of both images. Our experimental results demonstrate the efficiency of this approach in handling occlusions for more realistic synthesis of novel views.

5.2 Limitations and Future Work

The approach we proposed in this thesis combines both field morphing and view morphing, which has much more image resampling operations. These operations are very sensitive to image noise. In order to get better results, high quality images, which were taken in the same good lighting configuration, may be required. One of the future work might be considering the influence of illumination and modeling the surface reflectance in different viewpoints during view synthesis.

Like any other image morphing algorithms, both the starting and ending positions of feature boundaries should be known in advance to implement image interpolation in our approach. Although we have done some experimentation with extrapolation, results indicate that this problem is highly ill-posed. Some of the future work might be studying the movements of the feature points/lines and predicting reasonable feature positions outside the range of reference images. Temporal blending is also another issue that we would like to consider.

LIST OF REFERENCES

[1] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *Proc. SIGGRAPH 92*, 1992.

[2] Steven M. Seitz and Charles R. Dyer. View morphing. In *Proc. SIGGRAPH 96*, 1996.

[3] Richard I. Hartley. In defense of the 8-point algorithm. In *Proc. Fifth Int. Conf. On Computer Vision*, 1995

[4] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2000

[5] George Wolberg. Recent advances in image morphing. *Computer graphics international,* 1996

[6] George Wolberg. *Digital image warping*. IEEE Computer Society Press, 1990

[7] Detlef Ruprecht and Heinrich Muller. Image warping with scattered data interpolation. *IEEE Computer Graphics and Applications,* 1995

[8] Nur Arad, Nira Dyn, Daniel Reisfeld, Yehezkel Yeshurun. Image warping by radial basis functions: Applications to facial expressions. *CVGIP: Graphical Models and Image Procession.* 1994

[9] Seung-Yong Lee, Kyung-Yong Chwa, James Hahn, Sung Yong Shin. Image morphing using deformation techniques. *The Journal of Visualizatin and Computer Animatin,* 1996

[10] Seung-Yong Lee, Kyung-Yong Chwa, Sung Yong Shin, George Wolberg. Image metamorphosis using snakes and free-form deformations. *Computer Graphics*. 1995

[11] Shenchang Eric Chen, Lance Williams. View interpolation for image synthesis. *Proc. SIGGRAPH 93,* 1993

[12] Mattew Regan, Ronald Post. Priority rendering with a virtual reality address recalculation pipeline. *Proc. SIGGRAPH 94,* 1994

[13] Jay Torborg, James T. Kajiya. Talisman: Commodity realtime 3D graphics for the PC. *Proc. SIGGRAPH 96,* 1996

[14] Jed Lengyel, John Snyder. Rendering with coherent layers. *Proc. SIGGRAPH 97,* 1997

[15] Apple Computer Inc. QuicktimeVR

[16] Shenchang Eric Chen. Quicktime VR – An image-based approach to virtual environment navigation. *Proc. SIGGRAPH 95,* 1995

[17] Black Diamond Inc. Surround Video, 1997

[18] Interactive Picture Corporation Inc. IPIX, 1997

[19] Infinite Pictures Inc. SmoothMove, 1997

[20] Live Picture Inc. RealSpace Viewer, 1997

[21] Lenoard McMillan, Gray Bishop Plenoptic modeling: An image-based rendering system. *Proc. SIGGRAPH 95,* 1995

[22] Edward H. Adelson, James R. Bergen. The plenoptic function and the elements of early vision. *Computational Models of Visual Processing* The MIT Preess, Cambridge, MA 1991

[23] Paul E. Debevec, Camillo J. Taylor, Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Proc. SIGGRAPH 96,* 1996

[24] Marc Levoy, Pat Hanrahan. Light field rendering. *Proc. SIGGRAPH 96,*1996

[25] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, Michael F. Cohen. The lumigraph. *Proc. SIGGRAPH 96*, 1996

[26] Heung-Yeung Shum, Li-Wei He. Rendering with concentric mosaics. *Proc. SIGGRAPH 99*, 1999

[27] Shmuel Peleg and Joshua Herman. Panoramic mosaics by manifold projection. *Proc CVPR 97,* 1997

[28] Takuji Takahashi, Hiroshi Kawasaki, Katsushi Ikeuchi, Masao Sakauchi. Arbitrary view position and direction rendering for large-scale scenes. *IEEE Computer Vision and Patrern Recongnition (CVPR00)*, 2000

[29] Shree Nayar. Catadioptric omnidirectional camera. *IEEE Computer Vision and Pattern Recognition*, 1997

[30] Daniel G. Aliaga, Ingrid Carlbom. Plenoptic stitching: A scalable method for reconstructing 3D interactive walkthroughs. *Proc. SIGGRAPH 01*, 2001

[31] Russell A. Manning, Charles R. Dyer. Interpolating view and scene motion by dynamic view morphing. *Proc. Computer Vision and Pattern Recognition*, 1999

[32] Jiangjian Xiao, Cen Rao, Mubarak Shah. View interpolation for dynamic scenes. *EUROGRAPHICS*, 2002

[33] Mare Alexa, Daniel Cohen-Or, David Levin. As-rigid-as-possible shape interpolation. *Proc. SIGGRAPH 00*, 2000