ANIMATING FACIAL IMAGES WITH DRAWINGS

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER
ENGINEERING AND INFORMATION SCIENCE
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BİLKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Gamze Dilek Tunalı
July, 1996

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Bülent Özgüç (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Cevdet Aykanat

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Özgür Ulusoy

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Science

ABSTRACT

ANIMATING FACIAL IMAGES WITH DRAWINGS

Gamze Dilek Tunalı
M.S. in Computer Engineering and Information Science
Supervisor: Prof. Bülent Özgüç
July, 1996

The work presented here describes the power of 2D animation with texture mapping controlled by line drawings. Animation is specifically intended for facial animation and not restricted by the human face.

We initially have a sequence of facial images which are taken from a video sequence of the same face and an image of another face to be animated. The aim is to animate the face image with the same expressions as those of the given video sequence.

To realize the animation, a set of frames are taken from a video sequence. Key features of the first frame are rotoscoped and the other frames are automatically rotoscoped using the first frame. Similarly, the corresponding features of the image which will be animated are rotoscoped. The key features of the first frame of the sequence and the image to be animated are mapped and using cross-synthesis procedure, other drawings for the given image are produced. Using these animated line drawings and the original image, the corresponding frame sequence is produced by image warping. The resulting sequence has the same expressions as those of the video sequence.

This work encourages the reuse of animated motion by gathering facial motion sequences into a database. Furthermore, by using motion sequences of a human face, non-human characters can be animated realistically or complex characters can be animated by the help of motion sequences of simpler characters.

Key words: facial animation, facial expression, snakes, active contour models, multigrid relaxation, multilevel B-Spline interpolation.

// ÖZET

YÜZ GÖRÜNTÜLERİNİ ÇİZİMLER YARDIMIYLA CANLANDIRMA

Gamze Dilek Tunalı

Bilgisayar ve Enformatik Mühendisliği

Yüksek Lisans

Tez Yöneticisi: Prof. Bülent Özgüç

Temmuz, 1996

Bu çalışma, çizimlerin denetiminde doku kaplama yöntemini kullanarak, iki boyutlu animasyonların gerçekleştirilmesini içerir. Çalışma özellikle yüz animasyonunu amaçlamaktadır ve yüzler insan yüzü olmak zorunda değildir.

İlk aşamada, aynı yüze ait bir video dizgisinden alınmış yüz görüntüleri ve canlandıralacak başka bir yüze ait bir görüntü kullanılır. Amacımız, verilen tek kare yüz görüntüsünü, video dizgisindeki yüz ifadeleriyle canlandırmaktır.

Bu çalışma sayesinde, canlandırılmış sıralı yüz hareketlerı bir veri tabanında toplanarak, verilen herhangi bir yüz için istenilen hareket sırası seçilerek, canlandırma gerçekleştirilebilir. Daha da önemlisi, insan yüzüne ait sıralı yüz hareketleri kullanılarak, insana ait olmayan yüzler gerçeğe çok yakın bir biçimde canlandırılabilir. Aynı zamanda, karmaşık karakterler daha basit karakterlerin hareket sıraları yardımıyla canlandırılabilirler.

Anahtar sözcükler: yüz animasyonu, yüz ifadesi, *snakes*, aktif ayırıcı çizgi modelleri, çok katlı ızgara interpolasyonu, çok seviyeli *B-Spline* interpolasyonu.

To my family

ACKNOWLEDGEMENT

I am very grateful to my supervisor, Professor Bülent Özgüç for his invaluable guidance and motivating support during this study. I would like to thank Dr. Cevdet Aykanat and Dr. Özgür Ulusoy for their remarks and comments on the thesis.

I would like to thank Aydın Alatan and Tahsin M. Kurç for their valuable contributions to the image processing and some other theoretical aspects. I also thank Ferit Fındık for his efforts to help me.

I am grateful to Peter Litwinowitcz and Seung-Yong Lee because of their patience and interest in my work. I wish to thank especially Peter Litwinowitcz for his valuable guidance.

Thanks are due to Uğur Güdükbay and Veysi İşler for their efforts in providing me very valuable documents.

I am grateful to Uğur Çetintemel for his moral support and assistance in solving numerious problems. I am also grateful to A. Kurtuluş Yorulmaz and all my friends for their moral support.

I have a deep gratitude to my family for everything they did for my being where I am today and, for their moral support all throughout my life.

Contents

List of Figures

Chapter 1

INTRODUCTION

In this study, 2D facial animation is performed which is controlled by line drawings. By aligning curves, lines and points with features in an image, intuitive controls for image warping are constructed. Deformation of an image can be accomplished by applying the warp defined by the original drawing and any other drawing of the same features. Animation is done simply by animating drawings and applying the image warp at each frame. The ability of mapping animation from one image and a set of features to another gives a power to animated sequences and enables the reuse of animated motion for any single face image.

Initially, we have frames of a video sequence of the same face and an image of another face that is to be animated. The goal of our work is to animate a given face image with the same expressions as those of the given video frames. As the first step, we outline some features which are of interest by hand on the first frame of the sequence, and then carry them to their real places by the help of snakes. Features are mouth, eyes, nose, eyebrows, etc. The face need not be human face for both images. For each feature on the first frame, we specify a corresponding feature on the image to be animated. Features can be considered as a collection of sequenced points on the image plane. For each feature correspondence, we have a set of point pairs. Using the cross-synthesis method explained in [14], we get an interpolation function which gives a pixel position on the image to be animated for each pixel on the first frame. Litwinowicz used multilevel surface reconstruction method to find the interpolation function, but it is slow. Furthermore, there will be no discontinuity in the function, therefore using this method is not advantegous. Instead, multilevel B-spline interpolation is used, which is faster and simpler than multilevel surface reconstruction method in this respect.

After the features are specified on the first frame, places of these features on the other frames in the sequence should be tracked. This process is performed automatically using two motion estimation techniques. For the end-points of the features, block matching technique and for the interior points, optical flow method are used to automatically find appropriate places of the features on the next frame.

Since we have acquired the interpolation function which gives pixel position correspondences of two initial images and tracks each feature on the video frames, a similar animated drawing sequence can be produced for the image to be animated. To produce the full set of animated images, we need a warp function for each animated drawing frame. For a number of known (x_k, y_k) positions in the image plane which are control points of the features, we have a set of known displacements $(\Delta x_k, \Delta y_k)$ as defined by the original and produced feature drawings. To warp the image according to the drawings, we need two interpolation functions $F_1(x_k, y_k) = \Delta x_k$ and $F_2(x_k, y_k) = \Delta y_k$. First function is a smooth interpolating function for the x-displacements for an entire image, and the second is similarly for the y-displacements. In this case, some discontinuities can be pointed out by the user for some features, therefore multilevel surface reconstruction method is more convenient. For each produced animated drawing frame of the sequence, a warp function is computed by using the original drawings. Then the warp functions F_1 and F_2 are applied to the still image for animation.

To find the mapping between corresponding feature drawings, *multilevel B-spline interpolation*; for image warping, *multilevel surface reconstruction methods* are used. Litwinowicz used multilevel surface reconstruction method to find the mapping between corresponding feature drawings. Instead, multilevel B-spline interpolation is used in our work, which is faster and simpler than the multilevel surface reconstruction method. To specify features on an image *snakes*, of Williams and Shah [24] are applied and these features are automatically tracked by applying *block matching* and *optical flow* methods.

This animation system is implemented in C, under Irix. User interface design is realized by using Motif combined with graphics library of SG Iris Indigo [8].

The organization of the text is as follows. Chapter 2 presents the most recent research that contribute to the study of facial animation. Some different techniques in animation studies are presented.

Chapter 3 explains the specifications of features on a face. Major studies in this area are compared, and the most advantegous one, Greedy algorithm is broadly explained.

In chapter 4, automatic feature tracking in a sequence of frames is explained. Block matching and optical flow methods are briefly presented.

In chapter 5, B-Spline interpolation is explained briefly.

Chapter 6 presents multigrid visual surface reconstruction method used in scattered data interpolation of warping step. After the mathematical basis is presented, discretization of the problem follows and finally multilevel equations are given.

Chapter 7 gives some example animation sequences produced. By using these examples, the important factors that effect our facial animation work are explained.

Chapter 8 gives a conclusion of the work.

Chapter 2

BACKGROUND

Traditional animation carries all action by drawings - points, lines and curves - defined at arbitrary instants. The work described here takes much of the motivation from the technique of traditional animation where interpolation defines the full sequence from the sparse keys. In our work, drawings on the objects or characters define the keys and an interpolation in space gives the full action of them on the image. The interpolated motion of a sequence of keyframe drawings define spatial deformations which may be applied to other images. Therefore, this technique encourages the reuse of animated motions. Feature-based deformations controlled by curves, lines and regions enable the animation of complex images and forms.

Animating drawings by their features is first studied by Litwinowicz et al. [11] by using a mesh of bilinear Coons patches [4]. Coons patches are inexpensive to evaluate, but they require manual division of the image into a mesh and all of the patch boundaries should be animated to control the motion. Therefore, it is time consuming and requires a substantial manual effort. Specifying and animating only the features of interest is more general and easier.

Deformations based on tensor-product splines [16, 3] permit an animated *skeleton* of linked line segments to drive the animation by polygonal tessellation of regions around the *bones* (line drawings) of the skeleton. An alternate parameterization has been described by Wolberg et al. [25] which is based on skeleton derived from the shape of the image region to be animated. The skeleton is obtained by successive thinning operations on the original shape. The works of Sederberg, Farin and Wolberg [16, 3, 25] are based on the skeletons and bones of them. But there is no guarantee that bones will align with the features the animator is interested in controlling directly.

More recent skeleton animation research has tried to use smoother interpolation functions. Van Overvald et al. [23] has improved the skeleton animation technique. He developed a physical simulation which is calculated for a simple skeleton and then applied to a more complex model by a distance-weighted *force field*.

The interpolation function used is equivalent to Shepard's interpolation that has been developed for terrain surfaces [17]. Beier and Neely [2] have also used Shepard's interpolation for image warping but they have also added line segments as control primitives to the animation. Since the lines can be aligned with edges in the image, the metamorphism was termed *feature based*. Since a line can do the work of dozens of points, it offers a natural and intuitive means of interpolating local orientations.

Harder and Desmarais introduced *thin-plate spline surfaces* to computer aided geometric design [5]. By the use of finite-element methods, smooth surfaces have been computed over the scattered data, for CAGD purposes by Pilcher et al. [15]. Smooth scattered data interpolation is analogous to physical surfaces and widely used in vision and image reconstruction. Fast numerical methods [22] supplied the demands of rapid processing for practical vision systems. Our animation system utilizes multigrid finite-difference evaluation of a thin-plate spline for animated deformations. This approach extends the feature primitives to curves and solid regions.

The most recent related work has been carried out by Litwinowicz et al. [14]. In that case, an actor's facial expressions are captured from video by the help of fluorescent spots on the actor's face. By these spots, *motion control points* are tracked. The acquired motion control points are spatially mapped to the synthetic face, giving new control points which are used to animate the synthetic face.

Chapter 3

FEATURE SPECIFICATION

After deciding on the features that are important for our animation sequence, they should be localized on the image accurately. Initially they are outlined by hand and then *snakes* are applied to carry them to their real places. To effectively use snakes to track edges on an image, a preprocessing operation should be performed. For this reason, Sobel [13] or another edge enhancement filter is run over the image. This produces a binary image which is white where the filter has detected an edge, and black otherwise. Then, the gradient [13] of the enhanced image is calculated to determine in which direction is the nearest edge.

A snake is an energy minimizing-spline guided by external constraint forces and influenced by image forces that pull it toward features such as lines and edges. Snakes are active contour models. They lock onto nearby edges, localizing them accurately.

An energy function, whose local minima comprise a set of alternative solutions, should be designed. Selection of the answer from this set is accomplished by the addition of some energy terms which push the model towards the desired solution. The solution is an active model that falls into the desired solution when placed near it. A snake always minimizes its energy functional until reaching the desired solution, so it exhibits a dynamic behaviour. Since it slithers like a snake while minimizing its energy, it is called as snake.

A snake has internal contour forces, external forces and image forces which are composed according to the desired behaviour of the contour. Internal spline forces serve to control the smoothness of the contour while external and image forces push the snake towards salient image features. Image energy term can include three different energy functionals which attract a snake to lines, edges and terminals. The total image energy can be stated as a weighted combination of these functionals according to the nature of the desired features. External forces can be exerted by the user to give direction to the contour in the desired way.

Kass, Witkin and Terzopoulos [7] have developed the snakes (Active Contour Models). The problems of the method of Kass, Witkin and Terzopoulos are numerical instability and a tendency for points to bunch up on strong portions of an edge contour. Amini et al. [1] has pointed out these problems and proposed a new algorithm using dynamic programming. This method is more stable and allows the inclusion of hard constraints but it is slow and having the complexity $O(nm^3)$ where n is the number of control points and m is the size of the neighborhood in which a point can move during a single iteration. Another method, a greedy algorithm for active contours was proposed by Williams and Shah [24]. This new method retains the improvements of older methods and also brings a new improvement, lower complexity. The complexity of the algorithm is $O(nm)$. The control points are more evenly spaced, so the estimation of curvature is more accurate.

The greedy algorithm is stable, flexible and allows hard constraints and runs much faster than the dynamic programming method. In addition to the other methods' internal spline energy and external energy terms, it includes a continuity term and a curvature term in the total energy to be minimized.

3.1 Minimum Energy Contours

3.1.1 Snakes of Kass, Witkin and Terzopoulos

Kass, Witkin and Terzopoulos [7] has developed an active contour model called snakes. In this method, controlled continuity spline can be operated upon internal control forces, image forces and some external forces applied by an interactive user or a higher level process.

In their work, a contour was represented by a vector $v(s) = (x(s), y(s))$ where s is the parameter denoting the arc length. They defined an energy functional and described a method to find the local minima of the functional as the solution. The functional is:

$$E_{snake} = \int_0^1 E_{snake}(v(s))ds = \int_0^1 E_{int}(v(s)) + E_{image}(v(s)) + E_{con}(v(s))ds \quad (1)$$

E_{int} represents the internal energy of the snake due to the bending or discontinuities. E_{image} is the image forces applied by some features like edges, lines and terminals in the image. E_{con} is the force applied by the external constraints.

Internal energy E_{int} is written as:

$$E_{int} = (\alpha(s)|v_s(s)|^2 + \beta(s)|v_{ss}(s)|^2)/2 \quad (2)$$

Equation 2 contains a first order term which will have large values when there is a gap in the curvature and a second order term which will be larger where the curve is bending rapidly. The relative sizes of α and β can be chosen to control the influence of the corresponding constraints. The minimum energy contour was determined by the variational calculus techniques.

In this method, forces can travel large distances along the contour, allowing faster convergence. On the other hand, image forces and constraints should be differentiable in order to guarantee the convergence. So it is not possible to include hard constraints such as minimum distance between points. As another drawback, intermediate results are not meaningful. The contour does not smoothly approach the minimum value.

3.1.2 The Solution of Amini

Amini et al. has pointed out some problems of snakes and proposed a new method which uses dynamic programming. This work introduced *hard constraints* that cannot be violated besides the continuity constraints inherent to the problem which are called *soft constraints*.

This method is numerically stable but slow, being $O(nm^3)$ and memory requirements are large, being $O(nm^2)$ where n is the number of points and m is the number of possible locations to which a point may move in a single iteration.

3.1.3 Advantages and Disadvantages Related to Both Methods

Besides the advantages and disadvantages specific to the methods themselves which are mentioned in the previous sections, there are some advantages and disadvantages related to both methods. Advantages of snakes of Kass and Amini are:

- A closed contour which is placed around an object outlines the entire object, rather than following texture edges on the surface of the object.
- Higher level processes can determine the values of external constraint term and the values of α and β. For example, corners can be allowed at certain points on the contours.

Disadvantages are as follows:

- α and β are used in both methods but there is no information about the values of them. It is apparent that their values are critical and must be chosen carefully to obtain meaningful results.
- If β is constant, corners will be not well defined. If points are far apart and a corner falls between these, there will be a problem on the contour.

- $|v_s(s)|^2$ in equation 2 is approximated as

$$|v_s(s)|^2 \approx (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2$$

  This is equivalent to minimizing the distance between the points and causes the contour to shrink.

- According to the minimization algorithm, points can move along the contour as well as perpendicular to it. This allows the points to bunch up in segments of the contour where the image forces are higher. Amini et al. used hard constraints to overcome this problem.

3.2 Greedy Algorithm

A greedy algorithm is presented by Williams and Shah [24] which allows the inclusion of hard constraints as described by Amini et al. [1] but much faster than their $O(nm^3)$ algorithm, being $O(nm)$. This algorithm allows a contour with controlled first and second order continuity to converge on an area of high image energy, in this case edges.

The algorithm is not guaranteed to give a global minimum but the experimental results produced by Williams and Shah were comparable to other methods.

The energy functional which will be minimized is:

$$E = \int (\alpha(s)E_{cont} + \beta(s)E_{curv} + \gamma(s)E_{image})ds \tag{3}$$

First and second terms correspond to E_{int} in equation 2. The last term measures some image quantity such as edge strength or intensity.

This method, as the methods of Kass and Amini, is iterative. At each iteration, points in the neighborhood of the current point are examined and the value of the energy function is computed at each of them. Then, one of the points in the neighborhood, giving the smallest energy value, is chosen as the new location of the current point. For example in figure 3.1, the neighborhood of point v_2 consists of 9 points (pixels) including itself. If the value of the energy function is smallest at v_2', then new location of the point at v_2 is chosen as the point v_2'.

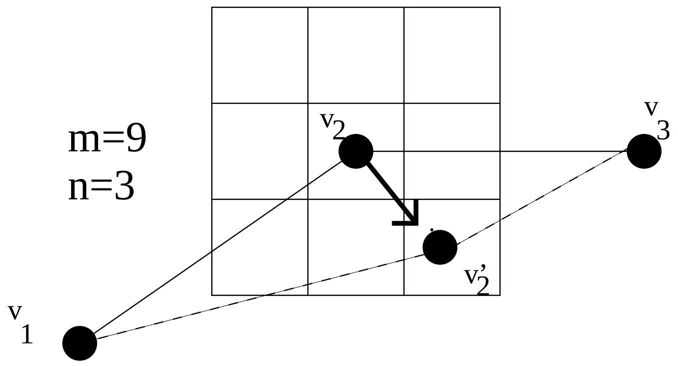

Figure 3.1: New location of a point in an iteration

The values of α and γ are considered as 1 and 1.2 in the study, so the image gradient will have slightly more importance than the contunity term to determine where the points on the contour move. β will be 0 or 1 depending upon whether a corner is assumed at that location.

Determining the first term E_{cont} of equation 3 presents some difficulties. If we use $|v_i - v_{i-1}|^2$ as Kass and Amini, contour tends to shrink while minimizing the distance between the points. It also contributes to the problem of points bunching up on strong portions of the contour. A term encouraging even spacing will reflect the desired behaviour of the contours. In this case, the original goal, first order continuity is still satisfied. So the algorithm uses the difference between $\bar{d}$, average distance between points, and $|v_i - v_{i-1}|$, the distance between the points: $\bar{d} - |v_i - v_{i-1}|$. By this formula, points having distance near the average will have the minimum value. The value is normalized by dividing by the largest value in the neighborhood to which the point may move, having a value in $[0, 1]$. At the end of each iteration, a new value of $\bar{d}$ is computed.

The second term E_{curv} in equation 3 is curvature. Since the continuity term causes the points to be relatively evenly spaced, $|v_{i-1} - 2v_i + v_{i+1}|^2$ is a reasonable estimate of curvature. This formulation has also given good results in the work of Kass and Amini. Like the continuity term, curvature term is also normalized by dividing the largest value in the neighborhood, giving a value in $[0, 1]$.

The third term E_{image} is the image force which is the gradient magnitude. Gradient magnitude is computed as eight bit integer with values $0-255$. There is a significant difference between 240 and 255 as gradient magnitudes. So normalizing the value by 255 will not reflect the differences. Thus given the magnitude (mag) at a point and the maximum (max) and minimum (min) gradient in each neighborhood, normalized edge strength term is computed as $(min - mag)/(max - min)$. This term is negative so points with large gradient will have small values. If the magnitude of a gradient at a point is high, it means that, the point is probably on an edge of the image. If $(max - min) < 5$ then min is given the value $(max - 5)$. This prevents large differences in the value of this term from occurring in areas where the gradient magnitude is nearly uniform.

At the end of each iteration, the curvature at each point is determined and if the value is a curvature maximum, then β is set to 0, otherwise it remains 1. This step is a primitive high level process giving feedback to the energy minimization process. Curvature is computed as $|\frac{\vec{u}_i}{|\vec{u}_i|} - \frac{\vec{u}_i}{|\vec{u}_i|}|^2$ where $\vec{u}_i = (x_i - x_{i-1}, y_i - y_{i-1})$ and $\vec{u}_{i+1} = (x_{i+1} - x_i, y_{i+1} - y_i)$. Then, nonmaxima suppression is performed on curvature values along the contour and curvature maxima points having curvature above a threshold are considered as corner points for the next iteration. A further consideration is that the gradient magnitude must be above some minimum value. This prevents corners from forming until the corner is near an edge.

Pseudo-code for the greedy algorithm is as follows:

```
Initialize α_i and β_i to 1 and γ_i to 1.2 for all i.
do
  /* loop to move points to new locations */
  for i = 0 to n
        E_min = BIG
        for j = 0 to m − 1
              E_j = α_i E_cont,j + β_i E_curv,j + γ_i E_image,j
              if E_j < E_min then
                    E_min = E_j
                    jmin = j
        Move point v_i to location jmin
        if jmin not current location, ptsmoved+ = 1
  /*process determines where to allow corners in the next iteration */
  for i = 0 to n − 1
        c_i = |u_i/|u_i| − u_{i+1}/|u_{i+1}||^2
  for i = 0 to n − 1
        if (c_i > c_{i−1} and c_i > c_{i+1}      /*if curvature is larger than neighbors */
          and c_i > threshold1                  /* curvature is larger than threshold */
          and mag(v_i) > threshold2             /* edge strength is above threshold*/
            then β_i = 0
until ptsmoved < threshold3
```

The threshold for setting $\beta = 0$ was 0.25, the threshold for the minimum gradient magnitude before a corner would be marked was 100 and the final threshold which is the number of points move to determine the convergence was a small nonzero value $(2 - 5)$. These values have given quite well results.

Chapter 4

AUTOMATIC FEATURE TRACKING

Some robot vision, animation and medical applications require feature tracking in video sequences. In our work, we focus on tracking edges in video sequences corresponding to facial features for animation purposes. By tracking an actor's facial expression, various computer animated characters can be driven.We have a sequence of facial images, so the motion is traced in 2D by the method of [12] and the animations are morphing of 2D images, but with sequences produced by two or more cameras, the motion can be tracked in 3D.

The features which are important for the animation purposes are outlined in the first frame by hand. Then, by using *active contours method* [7] mentioned in chapter 3, they are carried to their exact place on the image. For the other frames of the sequence, automatic edge finding process is applied to track the edges specified on the first frame.

During the edge finding process for each frame, the endpoints of snakes generally tend to move away from the corresponding features in the first frame. According to the motion of the features, they can slide back and forth along an edge. So, snakes that have a length preserving constraint are of little use for our work. Furthermore, if a feature moves far enough from one frame to another, a snake may switch edges. For instance, when you are viewing the video of a talking person, you can see that the lower edge of the upper lip visually replace the upper edge of the lower lip from one frame to another. Without motion prediction, a snake trying to track upper lip will suddenly find itself tracking the lower lip. Because of these problems, intensive user interaction may be necessary to extract motion from video sequences.

To track and position the endpoints of a snake, Litwinowitcz et al. [12] introduced the use of block matching technique for the first time. After block matching technique, the endpoints of a snake are held in place and non end-points are moved by optical flow method and then energy minimization process takes place. This technique avoids the sliding of a snake back and forth between frames.

As to the second problem, a snake can find an incorrect edge due to the large motion between frames. Litwinowitcz et al. [12] proposed the optical flow technique for the first time. Optical flow techniques generally do not produce perfect results for the motion of edges. However, after optical flow method is applied, energy minimization method can find the correct place as a last step. Thus, optical estimation is used to push a snake near to its desired edge.

4.1 Motion Estimation Techniques

4.1.1 Block Matching

Block matching is commonly used in motion analysis to find the correspondences among local image patterns in a sequence of images. The first step in our tracking process is to find the new locations of feature end-points based on their positions in the previous frame. The basic idea is to try to find the rectangular block, which is centered around the feature in the first frame, in the second frame.

The algorithm can be summarized by two figures. Figure 4.1(a) shows the displacement of an object from one frame to another. In figure 4.1(b), the cross indicates a feature end-point, the inner rectangle indicates block of best match around the feature point and the outer rectangle indicates a search area. The basic idea is to search the block of best match on the next frame within the specified search area.

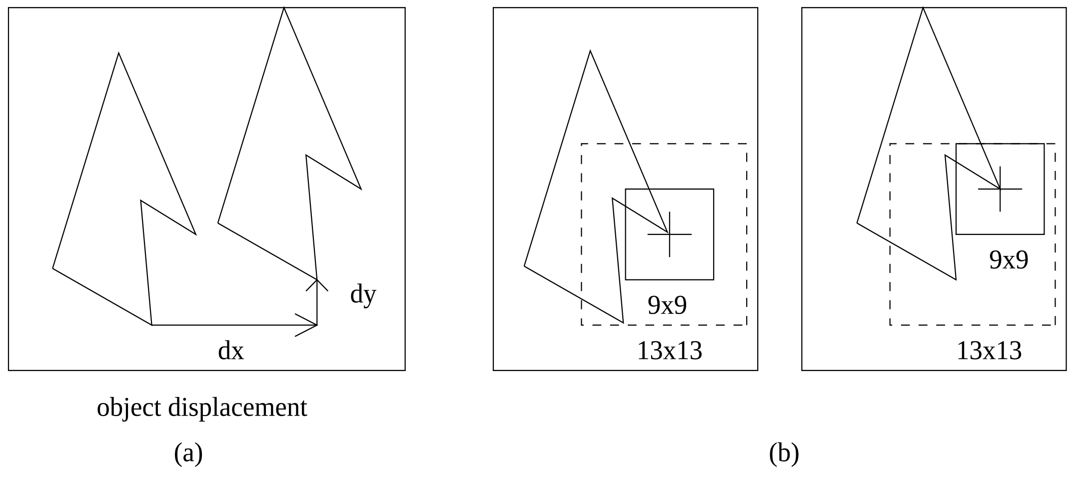

Figure 4.1: Basic algorithm of Block Matching

By experimentation, 13x13 block size and 9x9 search area size are found optimal by [12]. These sizes worked well for their video sequences.

In order to find the best match between blocks within the search area, a similarity measure is needed:

$$C(i,\delta) = \sum_{m} W_m F[f(i+m)] * F[h(i+m+\delta)] \tag{1}$$

The general correlation formula C is computed between a pattern in f centered at point i and a pattern h centered at $(i+\delta)$. The size of the pattern is determined by a window function W. A preprocessing operator F is applied to both reference frames. The comparison operator $*$ can be any operator to find the similarity between two reference frames.

Known comparison operators that measure similarity can be classified as absolute differences and squared differences. These tend to identify only identical images and changes of reflectance and illumination highly effect the measure. Correlation methods are used to measure similarity on the basis of pattern characteristics that are invariant over motion. Simple correlation measures are:

- *Direct Correlation* which uses simple multiplication operator as the comparison operator.

$$C(i,\delta) = \sum_m W_m f(i+m) h(i+m+\delta) \tag{2}$$

  Gives a high peak if patterns are identical but gives wrong results if mean values between blocks differ and therefore the maximum value may seldom be the point of exact match.

- *Mean Normalized Correlation* which eliminates the principal source of errors of direct correlation by subtracting the mean value of the block being considered ($\bar{f}$ and $\bar{h}$ respectively).

$$C_M(i,\delta) = \sum_m W_m \left(f(i+m) - \bar{f}(i)\right)\left(h(i+m+\delta) - \bar{h}(i+\delta)\right) \tag{3}$$

  The mean value of an MxN block b is:

$$\bar{b} = \frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N} E(k,l)$$

  where $E(k,l)$ is the intensity value at point (k,l) on the frame.

- *Variance Normalized Correlation* looks like equation 3 but in this case variance (Var) of the pattern is taken into consideration. It is very costly to compute but, it can be considered as an optimum measure since it gives 1 if exact match exists, otherwise gives a value between 0 and 1.

$$C_V(i,\delta) = \frac{\sum_m W_m \left(f(i+m) - \bar{f}(i)\right)\left(h(i+m+\delta) - \bar{h}(i+\delta)\right)}{\sqrt{Var_f(i)}\sqrt{Var_h(i+\delta)}} \tag{4}$$

  The variance of an MxN block b is:

$$Var_b = \frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N}\left(E(k,l) - \bar{b}\right)^2$$

Variance Normalized Correlation method is used in this work since it produces more correct results.

4.1.2 Optical Flow

The next step in the tracking process is to automatically push non end-points of a snake to wherever the corresponding image edge is moved by using optical flow technique. Optical flow technique was first developed by Horn and Schunck [6]. In feature tracking, the usage of optical flow method is first proposed by Litwinowicz et al. [12]. It was very convenient for feature tracking process, because it is independent from the number of snakes and the total number of snakes' control points. Initially, block matching was considered by Litwinowicz for all control points but accuracy could not be guaranteed and was much more time consuming. Optical flow technique is based on the assumption that illumination is constant and occlusion can be ignored, that is the observed grey-level changes are only due to the motion of underlying objects. In this case, it is evident that:

$$E(x, y, t) = E(x + \Delta x, y + \Delta y, t + \Delta t)$$

where E is the image brightness at point (x, y) in the image plane at time t.

When the pattern moves, the brightness of a particular point in the pattern is constant, so that

$$\frac{dE}{dt} = 0$$

Using the chain rule for differentiation,

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0$$

If we let $u = dx/dt$ and $v = dy/dt$ as velocities in the x and y direction, then we have a single linear equation with two unknowns u and v:

$$E_x u + E_y v + E_t = 0$$

The flow velocity (u,v) cannot be determined by one equation. The second constraint will be utilized is smoothness constraint. This constraint is necessary because if every point of the brightness pattern can move independently, we cannot recover the velocities. One way of expressing the smoothness constraint is to minimize the square of magnitude of the gradient of the optical velocity:

$$(\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 \ \ and \ \ (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2$$

Another measure of the smoothness of optical flow field is the sum of the squares of the Laplacians of the x and y components of the flow. Laplacians of u and v are defined as:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \ \ and \ \ \nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}$$

We have used the square of the magnitude of the gradient as the smoothness measure in our work.

Derivatives of brightness should be estimated from the discrete set of available image brightness measurements. Horn and Schunck proposed an estimate of E_x, E_y, E_t at a point in the center of a cube shown in figure 4.2 formed by eight measurements. Each of the estimates is the average of four first differences taken over adjacent measurements in the cube.

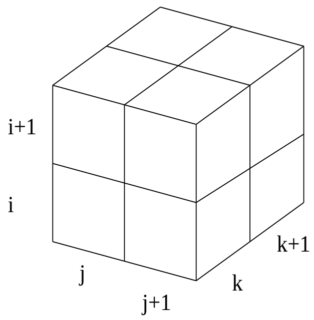

Figure 4.2: Horn and Schunck estimate of E_x, E_y and E_t.

$$\begin{aligned}
E_x \approx \tfrac{1}{4}\{ & E_{i,j+1,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i+1,j,k} \\
& + E_{i,j+1,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i+1,j,k+1}\}, \\
E_y \approx \tfrac{1}{4}\{ & E_{i+1,j,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i,j+1,k} \\
& + E_{i+1,j,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i,j+1,k+1}\}, \\
E_t \approx \tfrac{1}{4}\{ & E_{i,j,k+1} - E_{i,j,k} + E_{i+1,j,k+1} - E_{i+1,j,k} \\
& + E_{i,j+1,k+1} - E_{i,j+1,k} + E_{i+1,j+1,k+1} - E_{i+1,j+1,k}\},
\end{aligned}$$

Here the unit of length is the grid spacing interval in each frame and the unit of time is the image frame sampling period.

Also, Laplacians of u and v are needed to approximate:

$$\nabla^2 u \approx \kappa(\bar{u}_{i,j,k} - u_{i,j,k}) \;\; and \;\; \nabla^2 v \approx \kappa(\bar{v}_{i,j,k} - v_{i,j,k})$$

where the local averages $\bar{u}$ and $\bar{v}$ are defined as follows:

$$\begin{aligned}\bar{u}_{i,j,k} = \tfrac{1}{6}\{u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k}\} \\ +\tfrac{1}{12}\{u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}\}, \\ \bar{v}_{i,j,k} = \tfrac{1}{6}\{v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k}\} \\ +\tfrac{1}{12}\{v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}\},\end{aligned}$$

1/12	1/6	1/12
1/6	-1	1/6
1/12	1/6	2

Figure 4.3: Mask shows the suitable weights.

The proportionality factor κ is 3 with these neighboring weights and the assignment of weights to neighboring points are shown in figure 4.3.

Now, the problem is to minimize the sum of the errors in the equation for the rate of change of image brightness,

$$\mathcal{E}_b = E_x u + E_y v + E_t \tag{5}$$

and the measure of departure from smoothness in the velocity flow,

$$\mathcal{E}_c = (\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2 \tag{6}$$

Because of the possible quantization error and noise, we can not expect $\mathcal{E}_b$ to be identically zero. This quantity will tend to have a magnitude that is proportional to the noise in the measurement. The factor, will be denoted by α^2, determines the relative weight of $\mathcal{E}_b$ and $\mathcal{E}_c$. The total error to be minimized is:

$$\mathcal{E}^2 = \int\int (\alpha^2 \mathcal{E}_c^2 + \mathcal{E}_b^2)\, dx\, dy \tag{7}$$

The minimization is to be accomplished by finding suitable values for the optical flow velocity (u, v). Using the calculus of variation, we obtain

$$E_x^2 u + E_x E_y v = \alpha^2 \nabla^2 u - E_x Et,$$

$$E_x E_y u + E_y^2 v = \alpha^2 \nabla^2 v - E_y Et$$

Using the approximation to the Laplacian,

$$(\alpha^2 + Ex^2)u + E_x E_y v = (\alpha^2 \bar{u} - E_x E_t),$$

$$E_x E_y u + (\alpha^2 + Ey^2 v = (\alpha^2 \bar{v} - E_y E_t)$$

When we allow α^2 to tend to zero we obtain the solution to a constrained minimization problem.

Iterative Method:

We have now a pair of equations for each point on the image. It would be very costly to solve these equations simultaneously by one of the standard methods such as Gauss-Jordan elimination. The corresponding matrix is very large and sparse, so iterative methods such as Gauss-Seidel method, suggests themselves. At each iteration, (u^{n+1}, v^{n+1}) will be estimated by using the estimated derivatives and the average of the previous velocity estimates (u^n, v^n) by

$$u^{n+1} = \bar{u}^n - E_x\{E_x \bar{u}^n + E_y \bar{v}^n + E_t\}/(\alpha^2 + E_x^2 + E_y^2)$$

$$v^{n+1} = \bar{v}^n - E_y\{E_x \bar{u}^n + E_y \bar{v}^n + E_t\}/(\alpha^2 + E_x^2 + E_y^2)$$

The initial values of u and v for each point can be assigned to zero.

Chapter 5

MULTILEVEL B-SPLINE INTERPOLATION

After the feature correspondence between the two faces is set by the animator, a scattered data interpolation should be applied to find the correspondence between all the pixels of the two images. Uniform cubic B-spline surfaces are a good choice because they offer nice properties such as continuity and local control. B-spline method is much simpler and faster than the energy minimization method [9].

5.1 Manipulation of B-spline surfaces

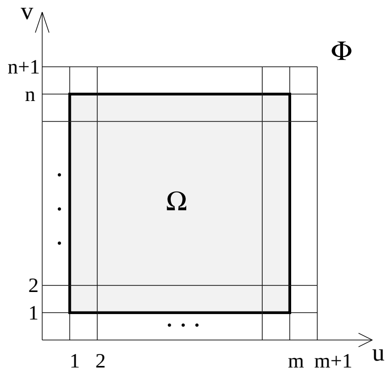

Figure 5.1: Lattice of control points on the uv plane

Let Ω be a rectangular region in the uv-plane which contains points $P = (u, v)$ such that $1 \leq u \leq m$ and $1 \leq v \leq n$. Let Φ be a $(m + 2) \times (n + 2)$ lattice of control points overlaid on the region Ω. It is shown in figure 5.1.

Initially, the control point ij is on lattice Φ lies on the point (i,j) in the uv-plane. If the control points on lattice Φ are displaced only in the direction perpendicular to the uv-plane (z direction), the resulting B-spline surface can be represented by a real valued function f. For all points $p = (u, v)$ on Ω, the function value f(p) implies that point p is placed at the position (u,v,f(p)) on the surface when the surface is generated.

Let ϕ_{ij} be the height of the control point (i, j) from the uv-plane. The function f can be stated as:

$$f(u, v) = \sum_{k=0}^{3} \sum_{l=0}^{3} B_k(s) B_l(t) \phi_{(i+k)(j+l)} \tag{1}$$

where $i = \lfloor u \rfloor - 1$, $j = \lfloor v \rfloor - 1$, $s = u - \lfloor u \rfloor$ and $t = v - \lfloor v \rfloor$. $B_k(s)$ and $B_l(t)$ are uniform cubic B-spline basis functions evaluated at s and t. Uniform cubic B-spline basis functions are as follows:

$$\begin{aligned}
B_0(t) &= (1 - t)^3/6 \\
B_1(t) &= (3t^3 - 6t^2 + 4)/6 \\
B_2(t) &= (-3t^3 + 3t^2 + 3t + 1)/6 \\
B_3(t) &= t^3/6
\end{aligned}$$

From the equation 1, we know that the function value of a point p depends on sixteen control points in its neighborhood. So the height of the ijth control point on lattice Φ is computed by using the set of points $P' = (u_c, v_c) \in P$ such that $i - 2 \leq u_c < i + 2$ and $j - 2 \leq v_c < j + 2$ (the point ij is the initial position of ϕ as in figure 5.2).

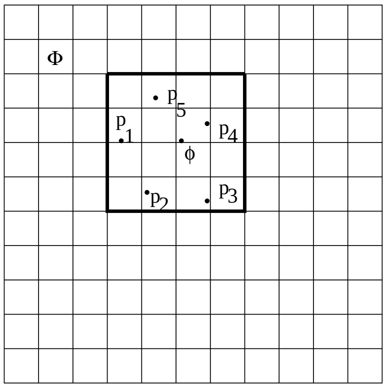

Figure 5.2: Sixteen neighbors of ϕ

When we displace the ϕ, displacement of all the points in P' are influenced. For each point p_c in P', the displacement ϕ_c of control point ϕ required for moving p_c to the specified point (u_c, v_c, t_c) is given by the equation:

$$\Delta\phi_c = \frac{w_{kl} t_c}{\sum_{a=0}^{3} \sum_{b=0}^{3} w_{ab}^2} \tag{2}$$

where $k = i + 1 - \lfloor u_c \rfloor$, $l = j + 1 - \lfloor v_c \rfloor$, $s = u_c - \lfloor u_c \rfloor$, $t = v_c - \lfloor v_c \rfloor$ and $w_{ab} = B_a(s)B_b(t)$

Since $\Delta\phi_c$ may be different from point to point in P', displacement $\Delta\phi$ of control point ϕ is chosen to minimize the error:

$$\sum_c (w_c\Delta\phi - w_c\Delta\phi_c)^2 \tag{3}$$

In the error, $w_c\Delta\phi$ is the displacement of point p_c due to the displacement $\Delta\phi$ of ϕ and $w_c\Delta\phi_c$ represents the contribution of control point ϕ to move p_c to its specified position (u_c, v_c, t_c). To minimize the error, differentiating the equation 3 with respect to $\Delta\phi$ and then equating to zero, $\Delta\phi$ is found as:

$$\Delta\phi = \frac{\sum_c w_c^2 \Delta\phi_c}{\sum_c w_c^2} \tag{4}$$

5.2 Multilevel B-spline interpolation

Let P be a set of points (u_c, v_c, t_c) where each (u_c, v_c) is in the region Ω. A function is required to interpolate all the points in P. By using the equation

1, we may not necessarily interpolate the points in P. The solution to the problem can be to use sufficiently fine control lattice so that every point in P can be interpolated without inferring with other points. But in that case, the surface shows sharp local deformations near the points in P [9]. Thus, multilevel B-spline interpolation is introduced to overcome this drawback.

In multilevel B-spline interpolation, there are m control lattices $\Phi_0, \Phi_1, \ldots, \Phi_m$ which are overlaid over the region Ω to derive the functions $f_0, f_1, \ldots, f_m$. h_i is defined as the spacing between control points of lattice Φ_i such that $h_i = 2h_{i+1}$. We assume that h_0 and h_m are given. The coarsest spacing h_0 determines the effect of an interpolated point on the resulting surface and the finest spacing h_m controls the precision to which the resulting surface interpolates the given points.

Interpolation process starts from the coarsest level. First, the heights of the control points on Φ_0 are derived and then the surface f_0 which interpolates the points in P is generated. The surface f_0 may only pass near the points (u_c, v_c, t_c) in P leaving the deviation $\Delta^0 t_c = t_c - f_0(u_c, v_c)$. Then the next finer control lattice Φ_1 is used to obtain the surface f_1 which interpolates the points $(u_c, v_c, \Delta^0 t_c)$. Generally, the method is to derive the heights of the control points on lattice Φ_k and then generating the surface f_k which interpolates $(u_c, v_c, \Delta^{k-1} t_c)$ where $\Delta^{k-1} t_c = t_c - \sum_{i=0}^{k-1} f_i(u_c, v_c)$. This process continues to the finest level Φ_m until the maximum difference between the points in P and the final surface f falls below a given threshold. The final surface is defined as the sum of functions f_i, that is, $\sum_i f_i(w)$ for each point w on Ω.

Chapter 6

MULTIGRID VISUAL SURFACE RECONSTRUCTION

A control primitive's original and final shape defines a set of displacements. Namely, for a number of known (x_k, y_k) positions on the image plane, there are known displacements $(\Delta x_k, \Delta y_k)$ as defined by the original and final drawings. We should construct interpolating functions $F_1(x_k, y_k) = \Delta x_k$ and $F_2(x_k, y_k) = \Delta y_k$ to apply the image warp at each frame. Since the points (x_k, y_k) are arbitrarily spaced on the image domain, the term scattered [10] is used.

The visual surface reconstruction stage should assimilate the scattered information provided by the various processes and *fill in the gaps* in a way that the constructed surface is a unique, smooth and most consistent with the scattered information. The thin-plate spline is one solution to our goals. Effect of a particular primitive is global but the area most affected is between primitive and its nearest neighbors. The thin-plate spline is C^1 continuous, certainly smoother than a piecewise planar triangulated surface, and not so cuspy as a Shepard's interpolant [10].

The solution of the thin-plate spline requires computation on each point and solving a linear system. It is extremely expensive when the number of points increases. Discretizing the problem, the solution time is dependent on the strain energy in the plate and not on the number of the data points (beyond a small initialization cost) [10]. The grid sizes are on the order of the image size in pixel. To get the function value at each pixel, we make sure that at least one grid element corresponds to each pixel. So the size of the grid is large and we will use coarse to fine multiresolution method to calculate our intepolants efficiently.

6.1 The Thin Plate Model

The thin plate model provides an intuitive interpretation of the surface reconstruction problem. The model consist of a bounded planar region Ω, an elastic surface and a number of pins and springs. On the planar region Ω (assume on the xy plane), there are some pins in the z direction which resemble the depth constraints and heights of them are proportional to the corresponding depth constraint values. Since some of the measurements may be erroneous, an ideal spring which pulls the plate's surface toward it is attached to the tip of the each pin as shown in figure 6.1. The springs provide that the thin-plate surface passes near the constraints (pins) by leaving a small amount of deviation.

The reconstructed surface is then deflection function $u(x, y)$ defined over Ω that represents the plate's surface in its equilibrium position.

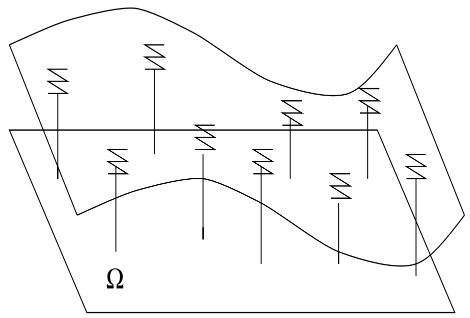

Figure 6.1: The thin-plate model

6.2 Mathematical Basis of Visible Surface Reconstruction

Let the distance $z = Z(x, y)$ (function of the image coordinates) be the distance from the xy plane to the surface. Low level visual processes generate a set of noise corrupted surface shape estimates (i.e. constraints) c_i which can be expressed as:

$$c_i = \mathcal{L}_i(x, y) + \epsilon_i \tag{1}$$

where $\mathcal{L}_i$ is the measurement functional and ϵ_i is the associated measurement error [22]. In the light of immediate definitions, visible surface reconstruction can be stated as: *reconstruct from available constraints c_i, the depth function $Z(x, y)$ along with an explicit representation of its discontinuities over the visual field.*

Let κ be a linear space of admissible functions. Let $\mathcal{S}(v)$ be a stabilizing functional which measures the (lack of) smoothness of a function $v \in \kappa$. Let $\mathcal{P}(v)$ be a penalty functional which measures the discrepancy between v and the given constraint. The energy functional is:

$$\mathcal{E}(v) = \mathcal{S}(v) + \mathcal{P}(v) \tag{2}$$

The solution $u(x, y)$ to the problem which minimizes the energy functional, characterizes the best reconstruction of the function $Z(x, y)$ as the smoothest admissible function $v \in \kappa$ which is most compatible with the available constraints. $u(x, y)$ should satisfy the Euler-Lagrange equation which is necessary condition to get the minimum energy functional value by taking first variational derivative δ_u of $\mathcal{E}(u)$ and equating to zero:

$$\delta_u \mathcal{E}(u) = \delta_u \mathcal{S}(u) + \delta_u \mathcal{P}(u) = 0 \tag{3}$$

6.2.1 Controlled-continuity Stabilizers

Controlled-continuity stabilizer provides local control over the continuity of the solution while preserving discontinuities. Controlled-continuity stabilizer of order 2 in two dimensions suffices in constructing C^1 continuous surfaces. C^1 continuous surface has continuously varying surface normal. The formula of the stabilizer is:

$$\mathcal{S}(v) = \frac{1}{2}\int\int_{\Omega} \rho(x,y)\{\tau(x,y)(v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2) + [1-\tau(x,y)](v_x^2 + v_y^2)\}dxdy \quad (4)$$

where $\rho(x,y)$ and $\tau(x,y)$ are real-valued continuity control functions which get a value in $[0,1]$. ρ and τ constitutes an explicit representation of depth and orientation discontinuities respectively over the visual field Ω. In our work, there is no orientation constraint and orientation discontinuity. Because we have not an information about the orientation of the surface, we have only a number of Δx_k and Δy_k values as depth constraints at each (x_k, y_k) on the Ω.

The formulas of controlled-continuity stabilizer and penalty functional will be given by considering both depth constraints and orientation constraints for completion of the mathematical basis but note that only depth constraints will be considered as only constraint when we discretize the problem. For more information about orientation constraints please refer to [22].

The variational derivative of x in the interior of Ω is given by:

$$\delta_u \mathcal{S}(v) = \frac{\partial^2}{\partial x^2}(\mu v_{xx}) + \frac{2\partial^2}{\partial x \partial y}(\mu v_{xy}) + \frac{\partial^2}{\partial y^2}(\mu v_{yy}) - \frac{\partial}{\partial x}(\eta v_x) - \frac{\partial}{\partial y}(\eta v_y) \quad (5)$$

where $\mu(x,y) = \rho(x,y)\tau(x,y)$ and $\eta(x,y) = \rho(x,y)[1-\tau(x,y)]$.

Since ρ and τ determine the local continuity of $u(x,y)$ at any point (x,y) in Ω,

$\lim_{\tau(x,y)\to 0} \mathcal{S}_{\rho\tau}(v)$ locally characterizes membrane spline, which is C^0 surface which needs only be continuous,

$\lim_{\tau(x,y)\to 1} \mathcal{S}_{\rho\tau}(v)$ locally characterizes thin-plate spline, which is C^1 surface which is continuous and has continuous first derivative,

$\lim_{\rho(x,y)\to 0} \mathcal{S}_{\rho\tau}(v)$ characterizes locally discontinuous surface.

Intermediate values of ρ and τ locally characterize a hybrid C^1 *thin-plate spline under tension* where $\rho(x, y)$ is a spatially varying surface *cohesion* and $[1 - \tau(x, y)]$ is the spatially varying surface *tension* [22].

6.2.2 Penalty Functional

Penalty functional is the total deformation energy of a set of ideal springs attached to the constraints. Scattered depth constraints determine the shape of the elastic surface at equilibrium. The springs let the $u(x, y)$ value to deviate from the constraint at the point (x, y), to supply the equilibrium of elastic surface.

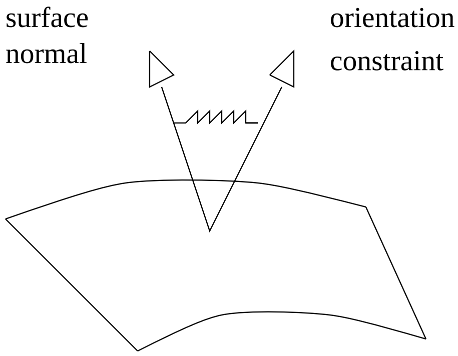

Figure 6.2: Local influence of an orientation constraint

Let us enumerate the constraints by i. If there is a depth constraint at (x_i, y_i),

$$d_{(x_i,y_i)} = v(x_i, y_i) + \epsilon_i$$

is the function value at that point and $i \in D$. Otherwise, (x_i, y_i) is an orientation constraint and $p_{(x_i,y_i)} = v(x_i, y_i) + \epsilon_i$ is the x component of the surface normal and

$$q_{(x_i,y_i)} = v(x_i, y_i) + \epsilon_i$$

is the y component of the surface normal at that point. If x component exists then $i \in P$ and if y component exists then i also an element of Q.

The penalty function can be written as:

$$\begin{aligned}\mathcal{P}(v) = &\frac{1}{2}\sum_{i\in D}\alpha_{d_i}[v(x_i,y_i) - d_{(x_i,y_i)}]^2 \\ &+\frac{1}{2}\sum_{i\in P}\alpha_{p_i}[v_x(x_i,y_i) - p_{(x_i,y_i)}]^2 \\ &+\frac{1}{2}\sum_{i\in Q}\alpha_{q_i}[v_x(x_i,y_i) - q_{(x_i,y_i)}]^2 \qquad (6)\end{aligned}$$

where α_{d_i} is the stiffness of the springs which control the depth constraints and α_{p_i} and α_{q_i} are the stiffness of the springs coercing the surface normal as shown in figure 6.2.

6.3 The Discrete Surface Reconstruction Problem

A closed form solution to the variational principle for visible surface reconstruction is infeasible due to the irregular occurrence of constraints and discontinuities [22]. So, by using finite element model, local approximations can be performed and the problem can be discretized.

6.3.1 Discretizing the Domain

The domain of the surface could be discretized by irregularly shaped elements, but discretizing will follow a Cartesian sampling pattern typical of images.

The domain Ω is teselled into square element subdomains with sides of length h. Nodes are located at corners of subdomains and the elements are interconnected at the nodes. The nodal variables ($v^h \in S^h$) are displacements of the plate at nodes. The element size h is adjustable so, one-to-one mapping can be achieved between nodes and pixels on the image. The nodes are indexed by (i, j) for $i = 1, \ldots, N_x^h$ and for $j = 1, \ldots, N_y^h$. A superscript h of a variable indicates that this variable defined over the grid where the element size is h. It is a convenient notation for multilevel structure. The total number of nodes, hence the number of nodal variables on a level will be $N^h = N_x^h \times N_y^h$.

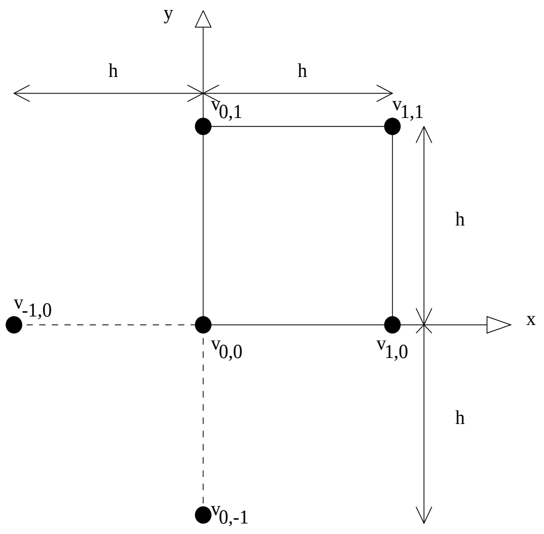

Figure 6.3: Unisolvent nodes for nonconforming element

A polynomial p^E is required within the element domain. The completeness condition, which must be satisfied, states that p^E be at least a general full-second degree polynomial [19]. When p^E is chosen to be six-degree-of freedom, full-quadratic polynomial, the requirement is satisfied. $p^E : E \rightarrow \Re$ is:

$$p^E(x, y) = ax^2 + by^2 + cxy + dx + ey + f$$

The six parameters a to f are determined uniquely in terms of the element node displacements at a p^E-unisolvent set of nodes which are shown in figure 6.3.

In figure 6.3, $v_{i,j} \in \Re$ denotes the node displacement and the parameters of p^E are as follows:

$$a = \frac{1}{2h^2}(v_{1,0} - 2v_{0,0} + v_{-1,0})$$

$$b = \frac{1}{2h^2}(v_{0,1} - 2v_{0,0} + v_{0,-1})$$

$$c = \frac{1}{h^2}(v_{1,1} - v_{0,1} - v_{1,0} + v_{0,0})$$

$$d = \frac{1}{2h}(v_{1,0} - v_{-1,0})$$

$$e = \frac{1}{2h}(v_{0,1} - v_{0,-1})$$

$$f = v_{0,0}$$

Six degrees of freedom is insufficient to enforce C^1 continuity of v^h across interelement boundaries. But, since the square elements pass the patch test [19], unique discrete solutions will converge to exact solution of the continuous problem as the discretizing is made increasingly finer.

6.3.2 The Discrete Equations

Since discretizing is realized, the functionals defined for continuous problem should be discretized in terms of nodal displacements. Partial derivatives at node (i,j) are:

$$v^h_{xx}|_E = p^E_{xx} = 2a = \frac{1}{h^2}(v^h_{i+1,j} - 2v^h_{i,j} + v^h_{i-1,j})$$

$$v^h_{yy}|_E = p^E_{yy} = 2b = \frac{1}{h^2}(v^h_{i,j+1} - 2v^h_{i,j} + v^h_{i,j-1})$$

$$v^h_{xy}|_E = p^E_{yy} = 2c = \frac{1}{h^2}(v^h_{i+1,j+1} - v^h_{i,j+1} - v^h_{i+1,j} + v^h_{i,j})$$

$$v^h_x = \frac{1}{h}(v^h_{i+1,j} - v^h_{i,j})$$

$$v^h_y = \frac{1}{h}(v^h_{i,j+1} - v^h_{i,j})$$

By substituting these partial derivatives in equation 4, we can write the discrete controlled-continuity stabilizer as:

$$\mathcal{S}^h_{\rho\tau}(v^h) = \frac{1}{2}\sum_{i,j} \rho^h_{i,j}\{\frac{\tau^h_{i,j}}{h^2}[(v^h_{i+1,j} - 2v^h_{i,j} + v^h_{i-1,j})^2$$

$$+2(v^h_{i+1,j+1} - v^h_{i,j+1} - v^h_{i+1,j} + v^h_{i,j})^2$$

$$+(v^h_{i,j+1} - 2v^h_{i,j} + v^h_{i,j-1})$$

$$+[1 - \tau^h_{i,j}][(v^h_{i+1,j} - v^h_{i,j})^2$$

$$+(v^h_{i,j+1} - v^h_{i,j})^2]\}$$

Assuming a one-to-one mapping between nodes and image pixels, a constraint or discontinuity may coincide with a node on the grid, but not all nodes be constrained or defined as a discontinuity.

Penalty functional will be given in the case of depth constraint only. For the complete expressions, see [22]. The discrete form of equation 6 becomes:

$$\mathcal{P}^h(v^h) = \frac{1}{2}\sum_{(i,j)\in D} \alpha^h_{d_{i,j}}(v^h_{i,j} - d^h_{i,j})^2 \tag{7}$$

The gradient of the discrete energy functional should be minimized to find the surface u^h at equilibrium,

$$\nabla\mathcal{E}^h_{\rho\tau}(u^h) = \nabla\mathcal{S}^h_{\rho\tau}(u^h) + \nabla\mathcal{P}^h(u^h) \tag{8}$$

This formula is generally a nonlinear system of equations. For fixed $\rho_{i,j}$ and $\tau_{i,j}$ (preset discontinuities), the system reduces to a linear system of equations, because $\mathcal{E}^h_{\rho\tau}(u^h)$ is a quadratic form in the $u^h_{i,j}$ [22]. To find u^h, a linear equation for each node (i,j) should be solved simultaneously. The nodal equation at an arbitrary node (i,j) is given by $(\frac{\partial\mathcal{E}^h_{\rho\tau}(u^h)}{\partial u^h_{i,j}} + \frac{\partial\mathcal{P}^h(u^h)}{\partial u^h_{i,j}}) = 0$.

Letting $\mu^h_{i,j} = \rho^h_{i,j}\tau^h_{i,j}/h^2$ and $\eta^h_{i,j} = \rho^h_{i,j}(1-\tau^h_{i,j})$, the partial derivatives are:

$$
\begin{aligned}
\frac{\partial \mathcal{E}^h_{\rho\tau}(u^h)}{\partial u^h_{i,j}} = \{ & (u^h_{i,j} - 2u^h_{i-1,j} + u^h_{i-2,j})\mu^h_{i-1,j} \\
& +(-2u^h_{i+1,j} + 4u^h_{i,j} - 2u^h_{i-1,j})\mu^h_{i,j} \\
& +(u^h_{i+2,j} - 2u^h_{i+1,j} + u^h_{i,j})\mu^h_{i+1,j} \\
& +(2u^h_{i,j} - 2u^h_{i-1,j} - 2u^h_{i,j-1} + 2u^h_{i-1,j-1})\mu^h_{i-1,j-1} \\
& +(-2u^h_{i+1,j} + 2u^h_{i,j} + 2u^h_{i+1,j-1} - 2u^h_{i,j-1})\mu^h_{i,j-1} \\
& +(-2u^h_{i,j+1} + 2u^h_{i-1,j+1} + 2u^h_{i,j} - 2u^h_{i-1,j})\mu^h_{i-1,j} \\
& +(2u^h_{i+1,j+1} - 2u^h_{i,j-1} - 2u^h_{i+1,j} + 2u^h_{i,j})\mu^h_{i,j} \\
& +(u^h_{i,j} - 2u^h_{i,j-1} + u^h_{i,j-2})\mu^h_{i,j-1} \\
& +(-2u^h_{i,j+1} + 4u^h_{i,j} - 2u^h_{i,j-1})\mu^h_{i,j} \\
& +(u^h_{i,j+2} - 2u^h_{i,j+1} + u^h_{i,j})\mu^h_{i,j+1}\} \\
& +\{(u^h_{i,j} - u^h_{i-1,j})\eta^h_{i-1,j} \\
& +(u^h_{i,j} - u^h_{i+1,j})\eta^h_{i,j} \\
& +(u^h_{i,j} - u^h_{i,j-1})\eta^h_{i,j-1} \\
& +(u^h_{i,j} - u^h_{i,j+1})\eta^h_{i,j}\} \qquad (9)
\end{aligned}
$$

$$
\frac{\partial \mathcal{P}^h(u^h)}{\partial u^h_{i,j}} = (\beta u^h_{i,j} - \beta d^h_{i,j}) \qquad (10)
$$

We can express $\nabla\mathcal{E}^h_{\rho\tau}(u^h)$ as:

$$\nabla\mathcal{E}^h_{\rho\tau}(u^h) = A^h u^h - f^h = 0 \tag{11}$$

where $A^h \in \Re^{NN}$ is a matrix of coefficients and $f^h \in \Re^N$ and $u^h \in \Re^N$ are column vectors. The linear term is $f^h = \beta c^h$, where those entries of $c^h \in \Re^N$ which are associated with constrained displacements are the constrained values $c^h_{i,j}$ and the rest are 0. N^h is the number of nodes on the grid where element size is h, namely $N^h = N^h_x \times N^h_y$. A^h is a positive-definite, symmetric and a sparse matrix. So the solution of equation 11 is easier than an ordinary linear equation. To find a linear equation for each node (i,j), the coefficients of $u^h_{i,j}$'s should be found by using the discrete controlled-continuity stabilizer and penalty functions. But it can be tedious for nodes at the boundary of Ω and nodes near the discontinuities. So we use computation molecules to make life easier.

6.3.3 Computational Molecules

The values of $\rho^h_{i,j}$ and $\tau^h_{i,j}$ in the discrete case are assumed in the range $[0,1]$. But if we permit $\rho^h_{i,j}$ and $\tau^h_{i,j}$ to get only the values in $\{0,1\}$ indicating {presence, absence} of discontinuities, suggests the following graphical interpretation of nodal equations.

Each term of equation 9 and 10 in the parenthesis may be visualized as a basic computational molecule [22]. Molecules consists of atoms, indicated by circles, arranged in the spatial grid pattern and containing coefficients of the associated nodal variables. Figure 6.4(a) illustrates ten plate molecules obtained from first component of equation 9 while 6.4(b) shows four membrane molecules obtained from second component of equation 9. The depth constraint molecule consists of one basic atom shown by figure 6.4(c). A double circle indicates the node (i,j), central to the nodal equation.

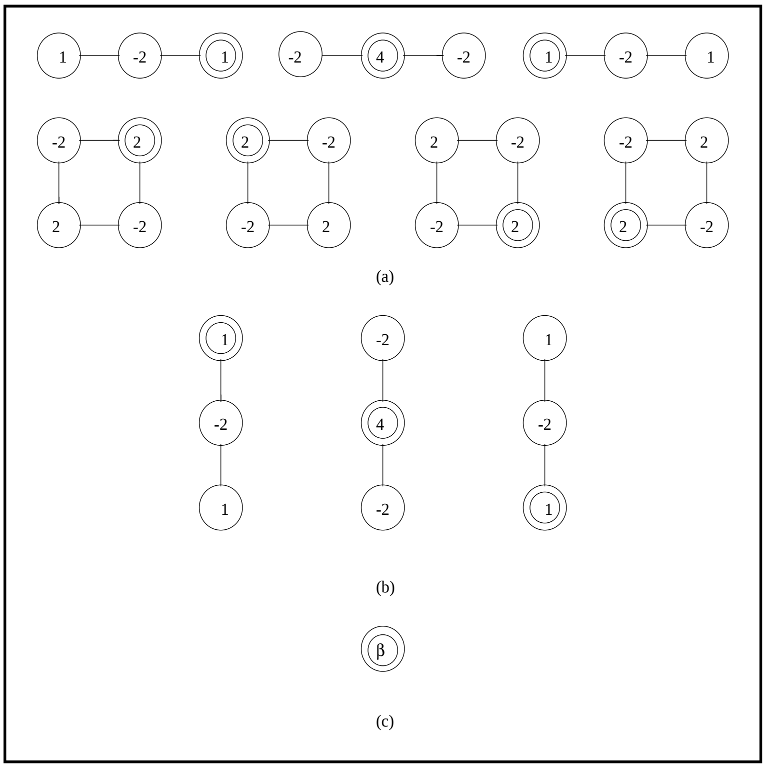

Figure 6.4: Basic molecules (a) Plate molecules (b) Membrane molecules (c) Depth constraint molecule

We have fifteen basic molecules at hand and then we will construct a computational molecule for each (i, j) on the grid by molecular summation. Discontinuities require molecular inhibition. If there exists a discontinuity at node (i, j) then $\mu^h_{i,j}$ or $\eta^h_{i,j}$ or both are zero, which inhibits the summation of certain molecules. Specifically, for a node (i, j), the corresponding computational molecule is found by summing basic molecules, which have not an atom corresponds to a discontinuity node on the grid or remain out of boundary, at the central atom. If depth constraint exists at the node (i, j) then depth constraint molecule is also added.

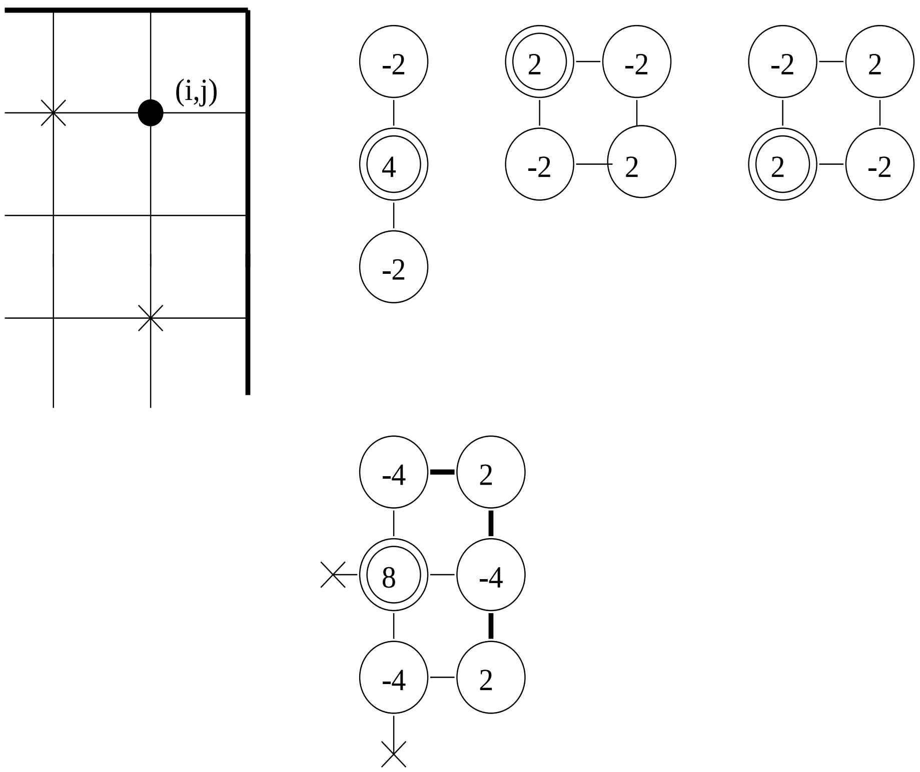

Figure 6.5: Constructing the computational molecule for the node (i,j)

Figure 6.5 shows some basic molecules to be added and resulting computational molecules as examples.

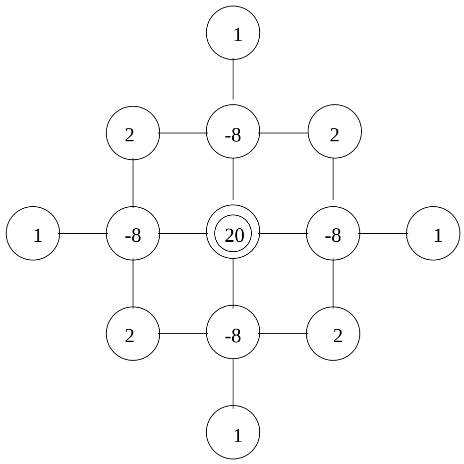

Figure 6.6: Biggest computational molecule

For example, the equation for the displacement at node (i, j) in the interior of Ω and away from discontinuities where computation molecule is shown by figure 6.6 is:

$$\frac{-8}{h^2}(v^h_{i-1,j} + v^h_{i+1,j} + v^h_{i,j-1} + v^h_{i,j+1})+$$

$$\frac{2}{h^2}(v^h_{i-1,j-1} + v^h_{i+1,j-1} + v^h_{i-1,j-1} + v^h_{i+1,j+1})+$$

$$\frac{1}{h^2}(v^h_{i-2,j} + v^h_{i+2,j} + v^h_{i,j-2} + v^h_{i,j+2})+$$

$$\frac{20}{h^2}v^h_{i,j} + \beta v^h_{i,j} = \beta c_{i,j}.$$

The terms involving β exist only if the node (i, j) has a depth constraint. β is taken as $\frac{2.0}{h^2}$ in the equations.

6.3.4 Solution of the Linear System of Equations

To find the reconstructed surface u^h as the solution of interpolating operation, the linear system of equations in the form of $A^h u^h = f^h$ should be solved.

The solution can be found as $f^h(A^h)^{-1}$ with a direct method. But the time to find the inverse of a large matrix is very long. Another method is LL decomposition. We will use a recursive method to find u^h. The sparseness, bandness and symmetrical structure of the matrix A is very convenient for a recursive method. Gauss-Seidel relaxation, Jacobi relaxation and successive overrelaxation method as well as gradient methods can be used to solve the equation. In this work, first Gauss-Seidel relaxation method was used, but this is very slow and not very prominent. Instead, Conjugate Gradient Method [18] is applied. It is the most prominent iterative method for solving sparse systems of linear equations.

6.4 Multilevel Equations

To introduce multilevel relaxation theory, consider the large, sparse system of linear equations $A^h u^h = f^h$ where A^h is nonsingular. The typical iterative solution technique is used to obtain a sequence of increasingly better approximations to the exact solution u^h by applying a long series of relaxation operations. This approach is inefficient because takes too much time.

One approach to increase the efficiency is to introduce $L-1$ similar problems on increasingly coarser levels. Discretizing can be done in the usual way by introducing a sequence of finite element spaces $S^{h_1}, \ldots, S^{h_L}$ over the rectangular domain Ω where L is the number of levels and $h_1 > \ldots > h_L$ are fundamental sizes of the elements at each level. The hierarchy of problems is then given by the sequence of L linear systems of the form

$$A^{h_k} u^{h_k} = f^{h_k}$$

where $1 \leq k \leq L$.

Element sizes are chosen as $h_k = 2h_{k+1}$. The finer level is L^{th} level and the coarsest level is 1^{st} level. Fixing the element size of finest level, the element sizes can be found on the coarser levels. Figure 6.7 shows a three level multigrid structure.

The problem at the coarsest level $A^{h_1} u^{h_1} = f^{h_1}$ can be solved quickly and the solution u^{h_1} can be used as an initial approximation to the next level. Proceeding in this way to the finest level L, a single accurate solution is obtained at the finest level. In the work of [21], interlevel operations are performed between levels to correct coarser levels' solutions. This is necessary to obtain solutions at each level as accurate as the finest level. But in our work, we do not need the solutions at the levels 1 to $L-1$. So multilevel cycle algorithm in [22, 19, 20, 21] is not used in our multilevel surface reconstruction problem.

6.4.1 The Multilevel Relaxation Algorithm

- Step1- Solve the coarsest-grid equation

Compute by relaxation an approximate solution u^{h_1} to the coarsest-grid equation $A^{h_1}u^{h_1} = f^{h_1}$. Set $l \leftarrow 1$.

- Step2- Set a new finest level

  If $l = L$ stop, otherwise increment l and set the finest approximation on the new level to be $\left(u^{h_l}\right)^0 = I_{h_{l-1} \Rightarrow h_l} u^{h_{l-1}}$.

- Step3- Perform relaxation

  Perform a relaxation iteration. If the norm of the current u^{h_l} is smaller than a precipecified value goto Step2, else goto Step3.

The operation $I_{h_{l-1} \Rightarrow h_l} u^{h_{l-1}}$ is an interpolation operation from level $l-1$ to level l. Bilinear interpolation [3] is used for this operation in our work.

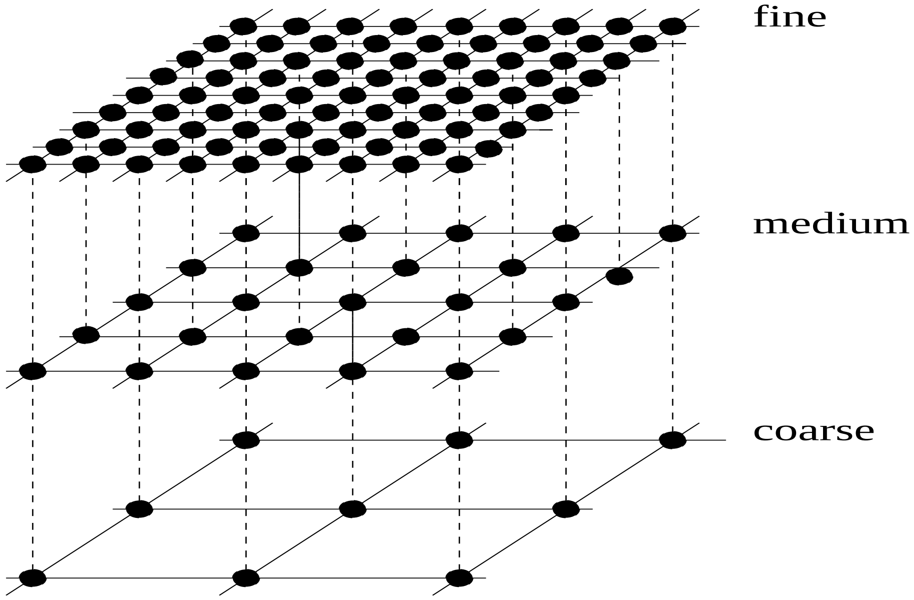

Figure 6.7: Three level multigrid structure

Chapter 7

SAMPLE ANIMATED FRAMES

7.1 Indefinite Feature Borders

In facial animation, finding features and tracking them correctly is the most important part. In the study of Litwinowitcz et al. [10], actors in the video sequences have a make up on their faces to highlight the important details [12]. In this study, we do not have a video sequence recorded in a similar way. Therefore, snakes find and track edges only according to the intensities and lighting highly effects the intensities. In this work, generally synthetic facial image sequences are used as the given sequence and their features are not definite adequately. But the implemented second and third parts of the thesis, namely finding the corresponding drawings and warping the image according to these drawings work well. This is demonstrated in all of the example figures. Animated drawings for the given image are consistent with the tracked drawings of the given sequence. Similarly, animated images are warped according to the changes of drawings from frame to frame.

In example 7.1, given sequence is obtained from a face maker program by changing some parameters of the eyes and mouth. Features of the face are not very definite, also some are visualized as incomplete. Because of the light source located on the right, right half of the lower lip is indefinite, so our snake algorithm passes this part reaching through a nearest edge. Although feature drawings of the sequence are not very correct, animated drawings are highly consistent with them. Similarly, produced (animated) images show the effects of deflections of animated drawings from the original drawings. Since only the outer border of the mouth is selected, open mouth is not realized by the animation.

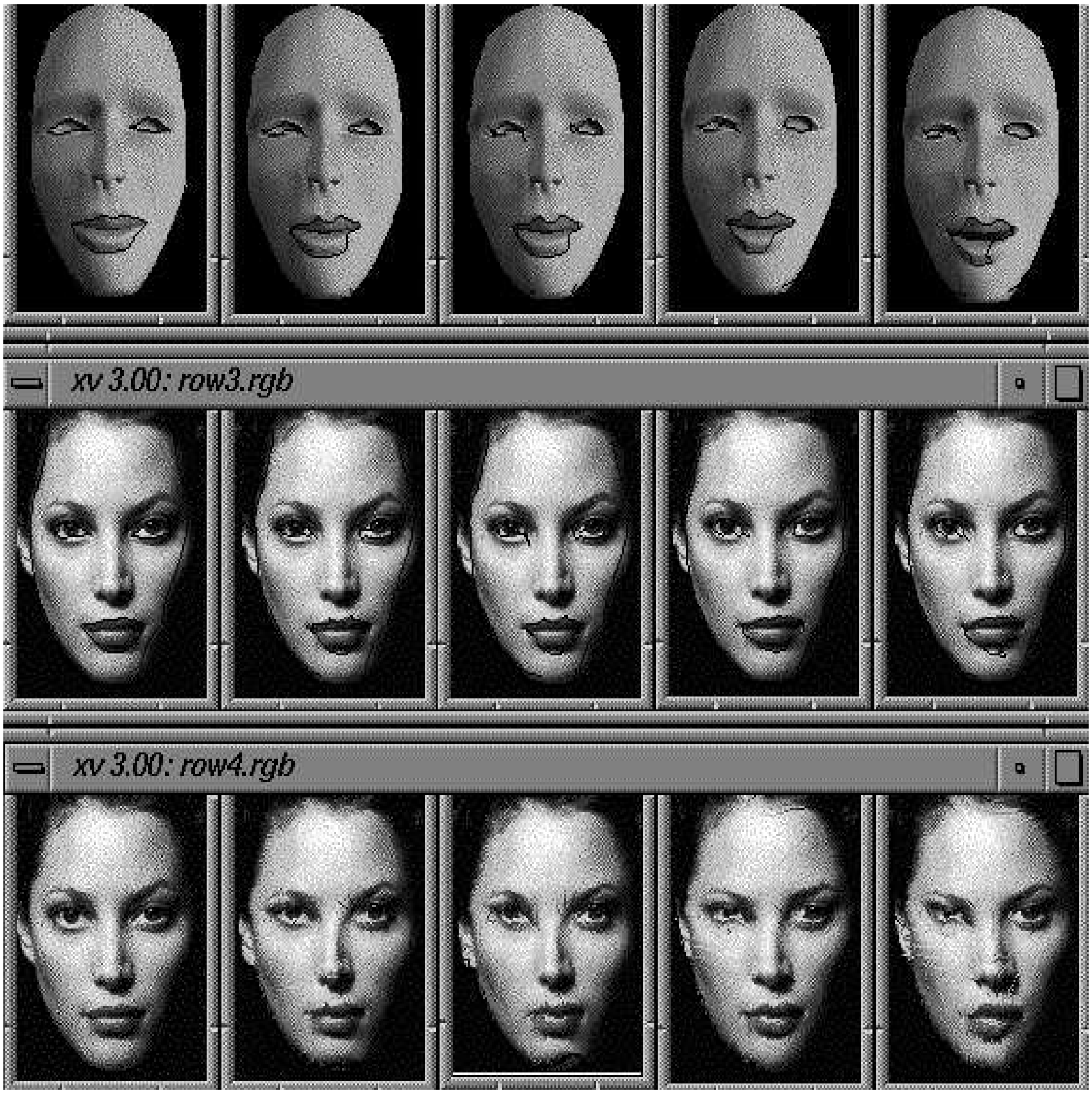

Figure 7.1: Tracked features, generated drawings on the original image and the animated image sequence

7.2 Which Features Should be Outlined?

To realize an animation most realistically, deciding the features to be specified is highly important. It is shown by the following example. In figure 7.2, we only want to map the change of mouth, and the corresponding mouths are selected on both images. Animated image shows a deformation mostly on the lower half of it. This part shrinks since the mouth does the same. If we could outline the borders of the faces, it would give a better result.

The same problem arises also in example 7.3. The eyes, the mouth and the left and right sides of the head are specified on the first frame and the corresponding features are specified on the face of the woman. The first row of figure 7.3 shows the frames of the given sequence. Left eyelid closes very slowly from first frame to third and mouth shrinks as kissing. Since motion should be smooth between frames for optical flow motion estimation technique, big changes between frames are not allowed.

While the features correspond to left eye and the mouth, they effect the nearby regions on the face. There is a shadow on the left eye visualized as an eyebrow but it is only a shadow on the face. While eyelid slightly closes, this shadow remains on the same place. But, while corresponding eyelid is closing, the eyebrow on it comes down by the effect of this eyelid. The same effect is shown around the mouth, especially on the nose.

To overcome these unwanted effects, two curves, one for the left eyebrow and the other for the right side of the nose is added to the existing features. This is shown in figure 7.4. Because of the reasons mentioned in the previous section, these newly added features are not tracked properly on the second frame. But the improvements on the eyebrow and nose are visualized clearly. In spite of the closing eyelid, eyebrow remained in its place. Nose is more close to its original appearance. If the left side of the nose could be specified properly, then more improvements could be achieved. Unfortunately, this part on the given sequence is too bright to be caught by the snakes.

7.3 Opening Mouth

In figure 7.5, an opening mouth is tracked with the feature specification of eyes, mouth, left and right sides of the head. Inner edges of the lips are only outlined to see the effect on the lips. First row shows the tracked head. First frame is displayed twice the size to effectively draw the edges. Other two frames are tracked automatically. Second row shows the animated drawings on the original image. Third row shows the animated frames. The lower ends of the side drawings of the head in the first row moved a little bit from the original drawings. Therefore, corresponding side drawings of the given image reflect this movements by deviating from the original lower ends. Since the warping functions account for these changes, produced images show a deformation on the neck. As a result, some incorrect movements of the features of the given sequence produce some deformations on the animated frames. The mouth is opened and lips moved accordingly in spite of determining only the inner boundary of the lips. This is the natural consequence of using thin-plate splines.

7.4 Marylin Monroe Kisses

In figure 7.6, first row shows the given sequence with the tracked features. The eyes, the mouth, the eyebrow and the two sides of the head are selected as features. The first frame of the second row shows the corresponding features which are drawn by hand on the original image. Other three frames show the generated drawings which are produced according to the corresponding drawings of the first row on the original image. The changes in the drawings from one frame to another reflect the motion of given sequence. The left eye closes from one frame to the next slightly, the mouth shrinks as kissing and the eyebrow raises slightly. The side drawings of the head remain as they are. The third row shows the animated images. The first frame is the original image and the other three are the warped images. These images reflect the facial expressions of the given sequence. The left eyebrow raises, the left eye closes slightly and the mouth shrinks as kissing.

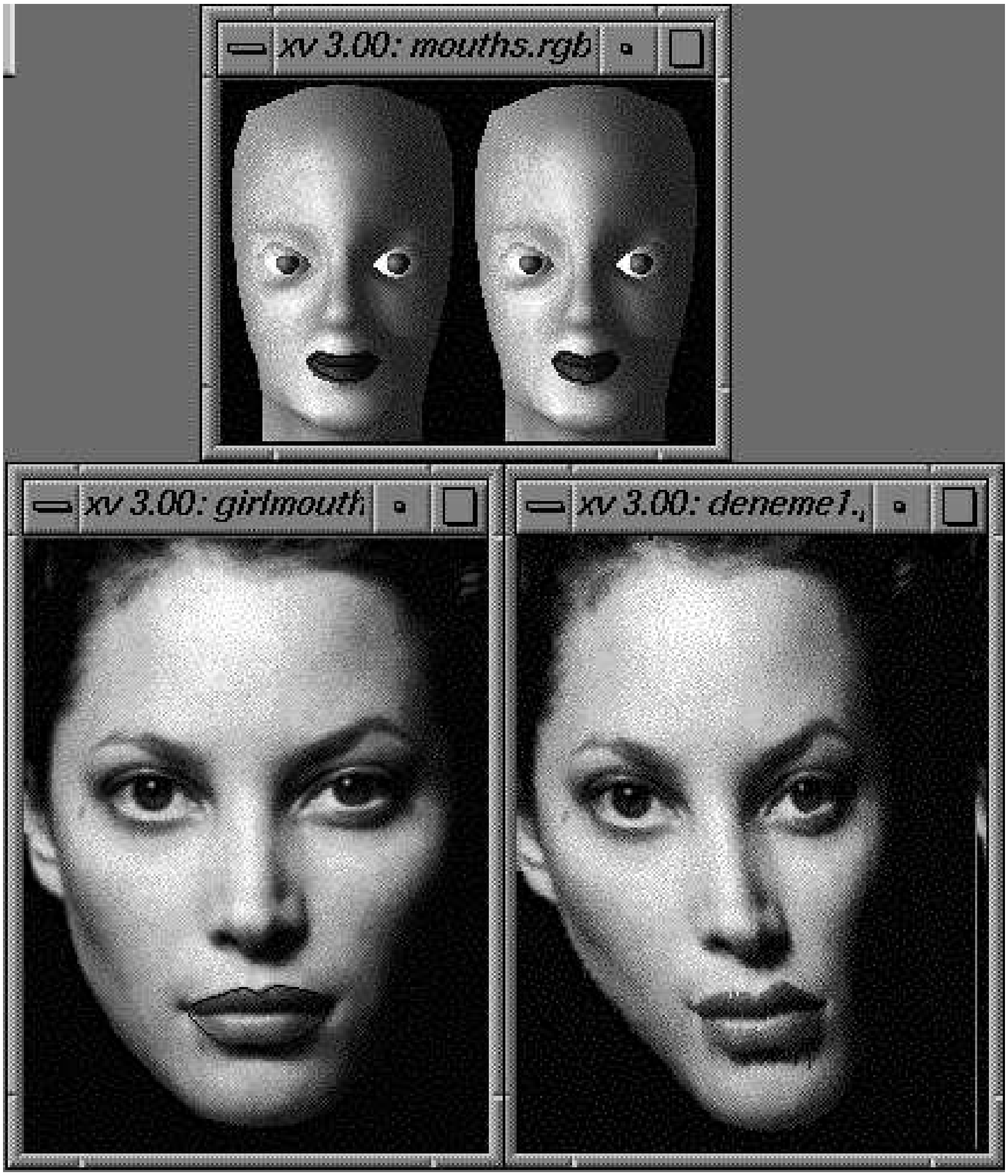

Figure 7.2: Generating a new frame by only specifying the mouths

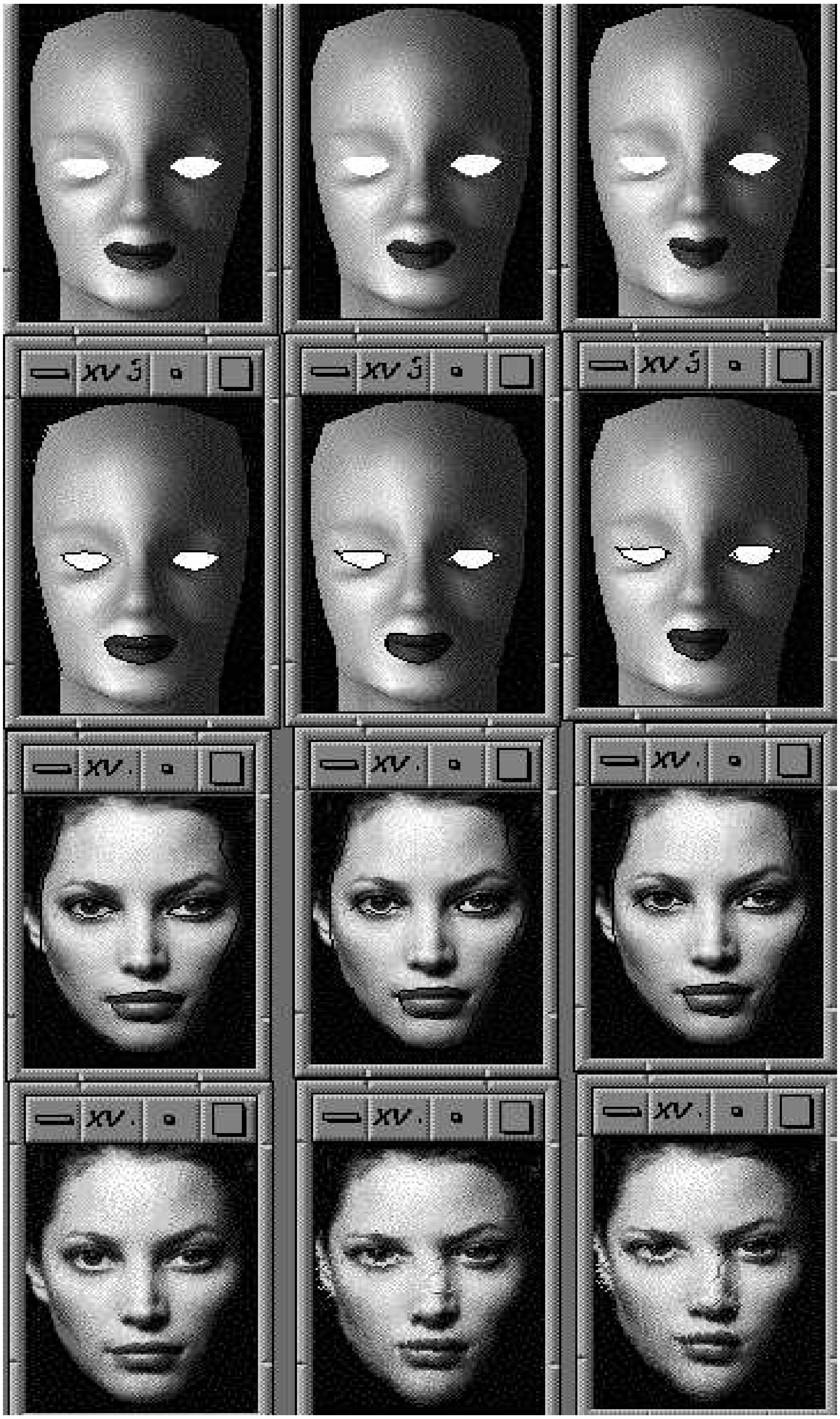

Figure 7.3: Change of the left eye and the mouth

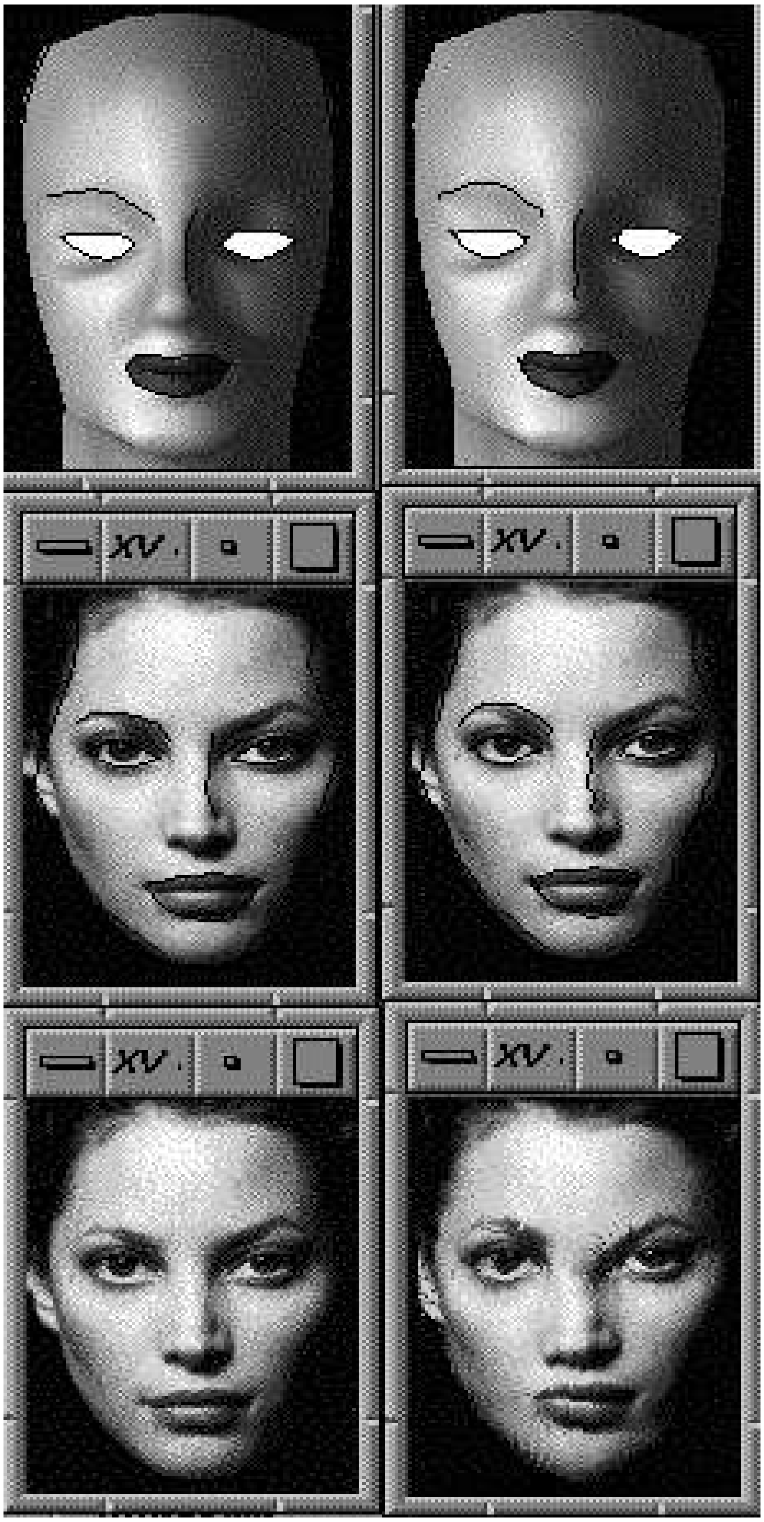

Figure 7.4: Adding two new features to the features of previous figure

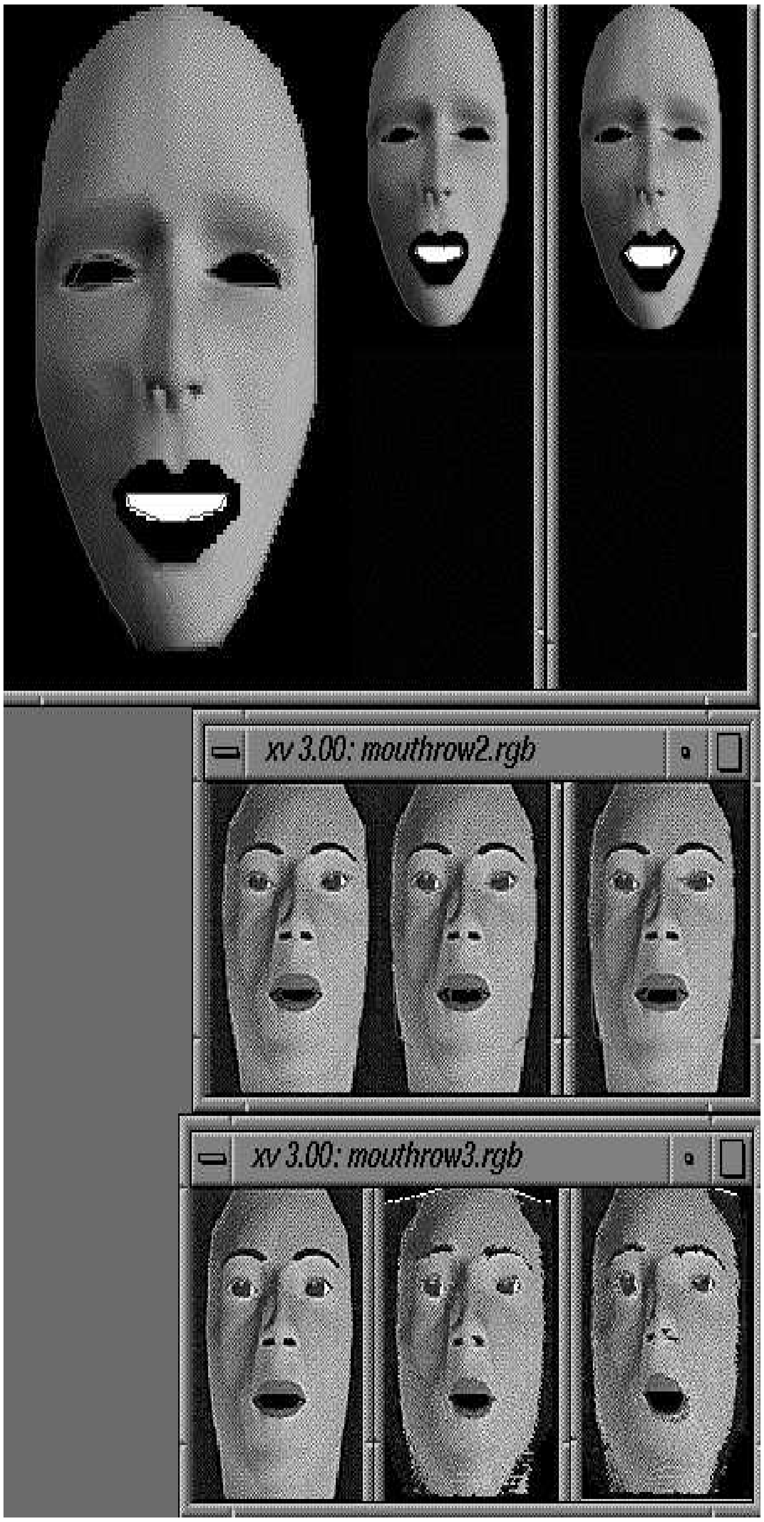

Figure 7.5: Opening Mouth

Figure 7.6: Marylin Monroe is kissing

Chapter 8

CONCLUSION

In this work, 2D facial animation controlled by drawings is presented. By aligning curves, lines and points with features (salient objects on a face), intuitive controls for image warping are constructed. Motion is basically obtained by animating drawings and applying image warp at each frame of the sequence.

A sequence of facial images which are taken from a video sequence of the same face or synthetic facial image sequences are taken and a still image of another face to be animated is selected. This face image is animated with the same expressions as those of the given sequence.

In the first step, salient facial features are outlined on the first frame of the sequence. The selection of these features are very important. Besides the major features that we want to animate, some auxiliary features should also be specified. The change of a feature mostly affects its nearest neighbors but its effect is global. This is the typical characteristic of thin plate splines which are used in image warping.

To extract the motion in video sequence, features which are indicated on the first frame should be tracked. This sequence may include hundreds of frames and outlining each feature on each of them manually is very time and effort consuming. Also, consistency may not be guaranteed. Therefore, automatic tracking of features is strongly necessary. In automatic tracking, for end-points of the features block matching, and for non end-points, optical flow methods and then snakes are used to find the real places of the features on the next frame.

After corresponding features are specified on both images, their control points should be mapped. This mapping can be one-to-one or many-to-one. One-to-one mapping is applied in this work. Since the corresponding features are drawn by different number of control points, a preprocessing operation should be done to equate them. Each corresponding drawings are linearly divided into a preset number. For each feature drawing, a different number is set according to its length.

After mapping the features of the first frame to the given image by using multilevel B-Spline interpolation, a function which maps each pixel of the first frame to a pixel of given image is found. By using this function, for each tracked frame of the video sequence, a corresponding animated feature drawings of the image can be produced.

The last step is to find a warp function by multigrid surface reconstruction method for each produced drawings that corresponds to a frame of the video sequence by using original drawings specified on the image. Warp function is basically a pair of $(\Delta x, \Delta y)$ displacement for each pixel on the original image. By using these displacements, a new image is constructed which shows the same expression as the corresponding video frame.

This work encourages the reuse of animated motion by gathering facial motion sequences into a database. For any single image, a sequence can be selected and animation can be realized. New features can be added at any time to both images (first frame and given image) without modifying the current mapping. By using motion sequence of a human face, non-human or synthetic faces can be realistically animated in cartoons and films. Similarly, by the motion sequence of simple characters, more complex characters can be animated. As a new consideration, by using these motion sequences, some objects other than a face can be animated by aligning some of their parts with features of the faces.

Some future improvements can be proposed related to this work. Multilevel surface reconstruction algorithm takes too much time, so it is the bottleneck of our animation system. Parallel implementation of this part greatly reduces the total animation time.

For mapping corresponding features one-to-one mapping is used in this work but, many to one mapping may be more convenient and produce better results.

Extra information besides the given image to be animated helps to produce more realistic results. For example, if a neutral face image is given which will be animated, mouth may be opened at any animated frame. In that case, to produce realistic images, teeth should be visualized. Small image parts other than the neutral face could be very useful to produce the more realistic animations.

Bibliography

[1] A. A. Amini, S. Tehrani, and T. E. Weymouth. Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints. In *Second International Conference on Computer Vision*, pages 95–99, Florida, December 1988.

[2] T. Beier and S. Neely. Feature-based image metamorphosis. *Computer Graphics*, 26(2):35–42, July 1992.

[3] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide*. Academic Press, NY, second edition edition, 1990.

[4] A. Forrest. On coons and other methods for the representation of curved surfaces. *Computer Graphics and Image Processing*, 1:341–369, 1972.

[5] R. Harder and R. Desmarais. Interpolating using surface splines. *Journal of Aircraft*, 9:189–191, February 1972.

[6] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[7] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):211–221, 1988.

[8] M. J. Kilgard. OpenGL and X, Part 3: Integrating OpenGL with Motif. Technical report, Silicon Graphics Inc., May 1994.

[9] S. Y. Lee, K.Y. Chwa, S. Y. Shin, and G. Wolberg. Image metamorphosis using snakes and free-form deformations. In *SIGGRAPH 95 Conference Proceedings*, pages 11–12, 1995.

[10] P. Litwinowicz and L. Williams. Animating images with drawings. In *SIGGRAPH 94 Conference Proceedings*, pages 409–412, 1994.

[11] P. Litwinowizc. Inkwell: A 2 1/2-d animation system. *Computer Graphics*, 25(4):113–121, 1991.

[12] P. Litwinowizc and M. Hoch. Enhanced snakes for edge tracking. Technical Report 57, Advanced Technology Group Apple Computer, Inc., February 1994.

[13] W. Niblack. *An Introduction to Digital Image Processing*, chapter Filtering, pages 73–76. Prentice/Hall International, 1986.

[14] E. Patterson, P. Litwionwitcz, and N. Greene. Facial animation by spatial mapping. In *Computer Animation 1991*, pages 31–44. Springer-Verlag, 1991.

[15] D. Pilcher. *Computer Aided Geometric Design*, chapter Smooth Parametric Surfaces, pages 237–253. Barnhill and Reisenfeld, Academic Press, NY, 1974.

[16] P. Sederberg and S. Parry. Free-from deformations of solid geometric models. *Computer Graphics*, 20(4):151–160, August 1986.

[17] D. Shepard. A two-dimensional interpolation function for irregularly spaced data. In *23rd Nat. Conf. ACM*, pages 517–523, 1968.

[18] J. R. Shewchuck. An introduction to the conjugate gradient method without the agonizing pain. Technical Report CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, March 1994.

[19] D. Terzopoulos. Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics, and Image Processing*, 24:52–96, 1983.

[20] D. Terzopoulos. Multilevel reconstruction of visual surfaces: Variational principles and finite-element representations. In *Multiresolution Image Processing and Analysis*, pages 237–310. Springer-Verlag, 1983.

[21] D. Terzopoulos. *Multiresolution Computation of visible surface representations.* PhD thesis, Dep. Elec. Eng. Comput. Sci. MIT, Cambridge, MA, Jan. 1984.

[22] D. Terzopoulos. The computation of visible-surface representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):417–438, July 1988.

[23] C. van Overveld. A technique for motion specification in computer animation. *The Visual Computer*, 6:106–116, 1990.

[24] D. J. Williams and M. Shah. A fast algorithm for active contours and curvature estimation. *CVGIP: Image Understanding*, 55(1):14–26, January 1992.

[25] G. Wolberg. Skeleton based image warping. *The Visual Computer*, 5(1/2):95–108, March 1989.