FITTING ACTIVE CONTOUR MODELS TO FACES IN VIDEOPHONE SCENES

E. TUĞRUL AKYÜZ

BOĞAZİÇİ UNIVERSITY

1995

FITTING ACTIVE CONTOUR MODEL TO FACES IN VIDEOPHONE SCENES

by

TUĞRUL AKYÜZ

B.S. in E.E., İstanbul Technical University,1992

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Electrical Engineering

Boğaziçi University

1995

ACKNOWLEDGEMENTS

I would first like to express all my thanks to my thesis supervisor Prof. Emin Anarım for his encouraging supervision during all stages of my thesis work.

I want to express by deep gratitude to Prof. Bülent Sankur for his continuous help and perfect advices and comments.

I would like to thank Assoc. Prof. Işıl Bozma for here sincere help she has given to me.

I want to thank Prof. Aytül Erçil for accepting to be on my thesis committee.

I would also like to thank to my frinds at Büsim for being kind and helpful during the development of my thesis.

ABSTRACT

This thesis presents an alternative approach to Teleconferencing Systems and refers to a system which uses a face sequence as input for animation of a wire frame model. The system exploits a general model of human face originally developed for realistic facial animation. The visual observation is achieved by using active contour models as snakes which work reciprocally with a physical model describing facial expression. The active contours serve to track the nonrigid motions of facial features and solve the correspondence problem of 3-D motion estimation. The 3-D rigid motion parameters are estimated by a neural network that is trained using back propagation algorithm. This system uses wireframe models and provides the imitation of a face sequence based upon the facial expression and 3-D rigid motion parameters estimation.

ÖZET

Bu tez , Tele-konferans sistemlerine bir seçenek teşkil edecek olan ve yüz imge dizisini girdi olarak kullanan bir kafes çerçeve modelinin canlandırılmasını sağlayan bir sistemi açıklamaktadır. Bu sistem, gerceğe cok yakın bir yüz canlandırması için geliştirilmiş genel bir insan yüz modelini ortaya koymaktadır. Görüntüsel tarama, yüz ifadelerini tarifleyen bir fiziki model ile eşzamanlı olarak çalısan kobra benzeri bir etkin çevrit modeli ile elde edilmektedir. Etkin çevrit modeli, yüz özniteliklerinin değişen devinimlerinin izlenimini sağlamaktadır ve üç boyutlu kati devinim kestiriminin denklik problemine çözüm getirmektedir. Üç boyutlu kati devinimin parametreleri, geri yayılım algoritmasi kullanılarak bilgi yüklenmis yapay bir sinir ağı ile kestirilmektedir. Sonuç olarak bu sistem, yüz ifadelerini ve üç boyutlu kati devinimlerin kestirilmesi suretiyle bize yuz imge dizisinin kafes çerçeve modeli ile taklit edilmesini sağlamaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1. Model-Based Image Coding

The human face is one of the most important parts of the body which has a communicative power in social interaction. Facial expressions and related changes in facial behavior inform us of the emotional state of a person. In order to understand more about the power of communicativeness of the face, it is necessary to study the perception of a face and related information processing.

Of all the nonverbal behaviors -body movements, posture, voice- the face is probably the most accessible part among the mechanisms which govern our emotional and social lives. The current technological developments provide the possibility of automated systems for monitoring facial expression and animating synthetic facial models.

The automation of the human face processed by a computer is a significant step towards developing effective human-machine interface. Therefore it is necessary to investigate the techniques and related algorithms focussed on understanding facial gestures and to automate these interpretations. In this context, understanding can be called analysis and the automation can be considered as the synthesis part.

Realistic animation is one of the most important tasks which would be serving as a bridge over the gap in between man and machine. In this perspective, computers may be extensively used in the training of children at various ages for all types of educational purposes. Another distictive area is the use of the methodology in the creation of high quality effects in the film industry. For example, using the animated vision of a death film star in a new movie. An important application of automated facial extraction, the one with most significant commercial potential in mutimedia

services, is video-phone or teleconferencing application. It is argued that for very low bit rate high quality video coding model-based coding of facial expressions in head-and-shoulder scenes, is a viable alternative. Such a scheme is realized by employing techniques from computer graphics. Beside being used in high compression application, these model-based systems can potentially be used to realize telepresence and virtual offices.

The analysis of facial expressions consists first in extracting features of the face such as eyes, lips (the most expressive parts indicating the emotional state of a person) and then tracking the evolution in time of these and other contours. For the extraction, a technique based on active contours model (snakes) is used . This method traces linear facial features, estimates the corresponding parameters of three dimensional wireframe face models, and reproduces facial expressions. This approach is introduced by Terzopoulos and Waters [21].

Another approach has been proposed by Mase who developed a method for tracing facial expression using optical flow. This approach is an extension of Pentland's work on lips reading.

1.1.1. Problem Statement

We address the problem of the rigid and local motion estimation of human face, and animation of wireframe model with respect to these parameters. This problem is solved in three steps .

1) Extraction of feature points (eyes, lips or mouth).

2) 3-D human head rigid motion estimation (rotation, translation)

3) Animation of the wireframe.

For the extraction of features, we use active contour (snakes) models. The

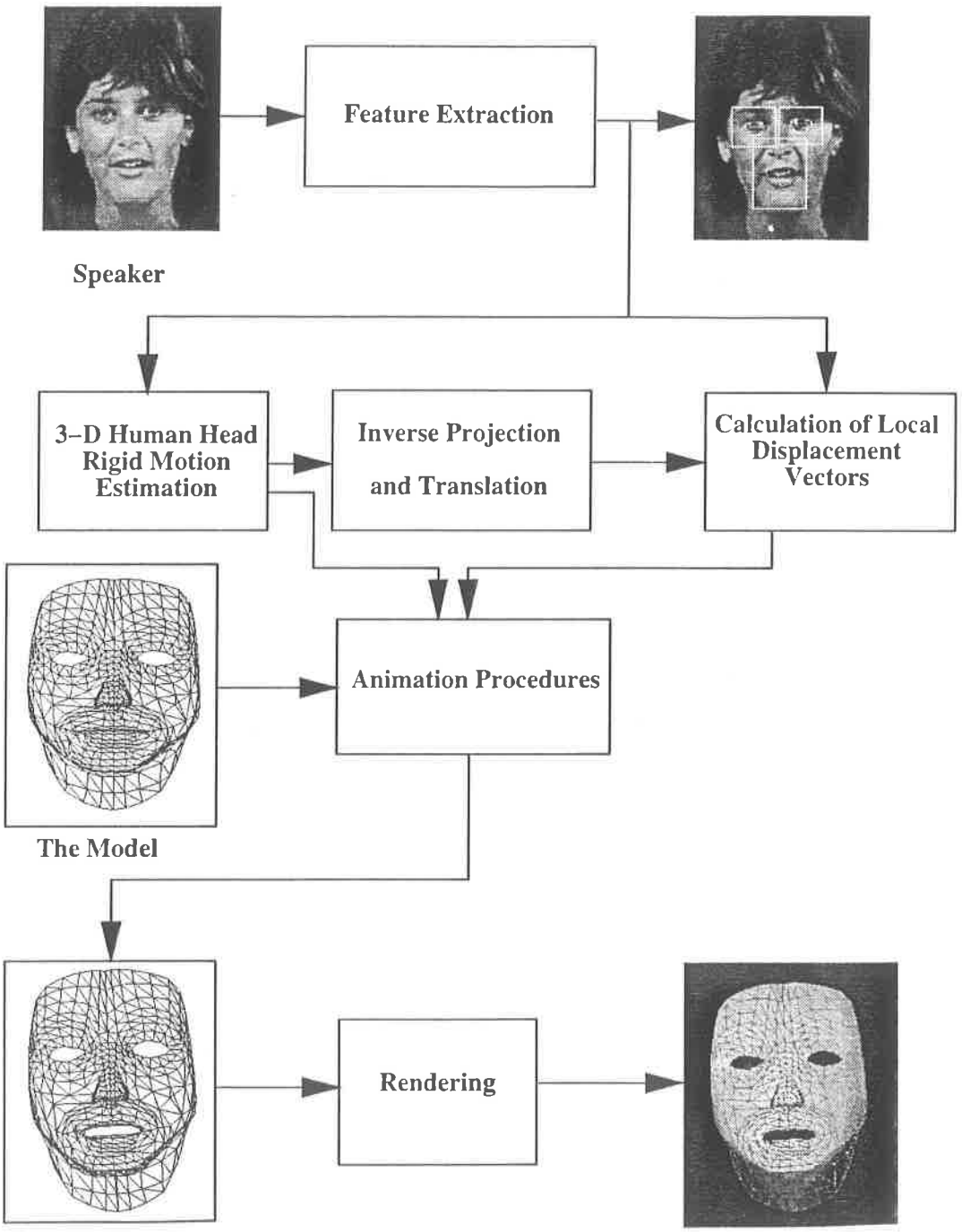

FIGURE 1.1. Tracking of a speaker's face scene by a wireframe

displacements of stationary features such as eyes and lips, give us information about the rigid motion of the head, such as head shaking and the local contour deformations caused by expression changes such as smiling, talking or eye blinding.

The system of the human head movement imitation is as follows: Features are extracted by using active contours. Later 3-D motion parameters are obtained out of the features that globally change. The local displacement vectors are calculated by using the 3-D motion estimation parameters and features points. Then animation is performed by using all of these informations. The system under consideration is shown in Figure 1.1.

THESIS OUTLINE

In this thesis, 3 main parts of model-based image coding such as feature extraction, 3-D motion estimation, animation are presented:

Chapter 1 describes an active contour that is a model which can alter its shape under the influence of energy fields. And the effectiveness of this model is discussed by following experiments.

Chapter 2 includes the 3-D human head motion estimation problem and gives an alternative solution different from the traditional analytical approaches, by using artificial neural network.

Chapter 3 discusses the animation procedures that is based on principles of mathematical physics. And then a two-layer model is developed for the animation of the 3-D wireframe human head.

2. Feature Extraction by Snakes

2.1. Introduction

In recent computational vision research, low level tasks such as edge or line detection and motion tracking have been widely regarded as autonomous bottom-up processes. The computations proceed by utilizing only what is available in the image itself and provide a unique answer for image segmentation. This rigid approach causes some mistakes made at low level without giving opportunity for correction.

This situation imposes a requirement for more reliable low level mechanisms. Such low level mechanisms should provide sets of alternative organizations among which higher level processes may choose rather than providing the users with a unique answer.

One of the important low level tasks is to find image boundaries. Such boundaries can be used for different purposes such as eyes and lips tracking for teleconferences or finding the shape of an organ for medical fields. The snake, which is a deformable model, permits us to simultaneously solve these segmentation and tracking problems.

2.2. Overview

A deformable model is a model which can alter its shape under the influence of energy fields. These energies effect the deformable model and force it to change its shape. Therefore , it is important to design energy functions with local minima comprising sets of alternative solutions, available to higher-level process. The choice among these alternatives could require some type of optimization in the absence of a

well-developed high-level mechanism. By adding suitable energy terms to the minimizations, it is possible for user to push the model out of a local minimum towards the desired solution. The result is an active model that falls into the desired solution when placed near to it.

Snakes can be represented as energy-minimizing splines guided by external constraint forces and image forces such as lines, edges, subjective contours and region homogenities found in the image. Furthermore, internal spline forces impose smoothness constraints on the modelled contours. By combining and integrating various type of information found in the image, snakes can lead to results that are comparable with other image-segmentation techniques such as edge linking and boundary detection.

The dynamic behavior of the snakes is its most distinguishing feature. This dynamic property can be used both for images , as well as for the image sequences. A snake will stick to an image feature by following any small deformation that may occur during image feature displacements from frame to frame. If the image feature moves only a small distance between frames, then snakes can function properly and catch up with the displacements. For larger displacements , snakes are relatively blind in their search of the desired contour. The snake is able to track only small deformations on its own, that is, without the help of higher level or more complex process. Most of the processing time required by this active contour model to converge to a image feature is consumed in the initial frame. Then the snake can track the image feature by dynamically sticking to its boundaries. However, some amount of user interaction is required to provide an initial position for the snakes in the first analyzed frame.

This variational approach in finding image contours, differs from traditional approaches of detecting edges and then linking them. Detection of edges gives a set of pixels which seldom characterizes a boundary completely due to noise or the breaks in the boundary from non uniform illumination, and other effects that introduce superious intensity discontinuities. Thus , edge detection algorithms typically are followed by

linking and other detection procedures such as neighborhood operators designed to assemble edge pixels into meaningful boundaries. For example, one of the simplest approaches for linking edge points is to analyze the characteristics of pixels in a small neighborhood about every point in an image, forming a boundary of pixels that share some common properties.

2.3. Previous Studies

Since the keystone of the segmentation and the tracking is the notion of an active contour model, we will consider briefly the previous studies in this direction.

The snakes are dynamic contours leading to the Euler-Lagrange equations of motion. The original model as presented by Kass [15] is used to locate smooth curves in 2-D imagery. What Kass proposed is an active contour which changes its shape with respect to internal and external energies. Snake's total energy can be defined mathematically as follows:

$$E_{snake} = \sum(E_{int} + E_{ext}) \tag{2.1}$$

Kass has argued that the internal energy prevents excessive snake deformations which in turn bring within the continuity constraints.

Terzopoulos and Watkin [15] add new external constraints such as line and corner point attractions in order to get the desired shape of the image objects. External energy is defined as follows;

$$E_{ext} = E_{image} = E_{edge} + E_{corner} + ...other types of energies. \quad (2.2)$$

Due to the flexibility of the above equation, further generalization is possible with the addition of new terms related to the image features.

Frederic Leymarie and Martin D. Levine use snakes for cell tracking and they propose improvements to the original description of the model and an improved termination criteria for the optimization scheme [13].

Laurent Cohen and Isaac Cohen introduce the balloon model as a way to generalize and solve some of the problems encountered with the original model proposed by Kass, Watkin and Terzopoulos. The problem arises when the snake is not affected by any counterbalancing forces. In that situation, it tends to shrink onto itself due to the minimization of internal energy. Accordingly, an internal pressure is introduced, by considering the snake as a balloon which is inflated. Furthermore, they used snakes for 3-D object reconstruction by defining 3-D surfaces as series of 2-D planar curves [5] [8].

An important aspect of research is about the minimization procedure of the snake energy. Amir Amani, Terry E. Weymouth and Ramesh C. Jain use dynamic programming for solving variational problems originating from the snake energy dissipation. Dynamic programming determines the minimum not by means of derivatives as in the variational approach, but rather by a straightforward search technique [2].

Kuk and Lai present in [12] an integrating approach in modeling, extracting, detecting and classifying deformable contours directly from noisy images. They develop an minimax criterion whereby the parameters of snakes can be automatically and implicitly determined along the contour. Furthermore they formulate a set of en-

ergy functionals which yield snakes that contain Hough transform that is used for the initialization of the deformable contours.

Alan Yuille proposed a method for detecting and describing features of faces using deformable templates [1]. The features of interest, for an eye, is described by parameterized templates. An energy function is defined which links edges, peaks, and valleys in the image intensity to correspond properties of the templates. The template then interacts dynamically by changing its parameter values in order to minimize the energy function.

2.4. The Model

A snake is a model of deformable curve or contours (if closed) composed of abstract elastic materials. Two types of material such as strings and rods are used are used in snakes. Strings make the snake resistant to stretching, whereas rods make it resistant to bending. The snake is constrained to lie in P , which is called potential surface under the action of a gravitational force g . In other words, a weight is assigned to the snake in order to make it fall down the slope of surface P.

Depending on the nature of the surface considered, the snake will be used for different purposes. Typically, the surface P can correspond to image intensities or to contrast values.

The natural way to force the snakes to move in order to reach a lower gravitational potential energy, by seeking valleys in the potential surface, is to convert potential energy to kinetic energy. The potential energy of the snake is given by sum of its gravitational potential energy and of the potential energy terms obtained from the internal and external constraints acting on it. Then, the kinetic energy is dissipated

by damping, resulting in the snake reaching a new lower equilibrium, that is, lower in terms of potential energy and, thereby, lower in terms of height. By following such a natural description of the dynamics of an active contour, the snake model has been first described in terms of a Lagrangian formulation of the motion.

Features in the image plane can be considered as data sets that can be represented by a deformable model. Reconstruction can be implemented in terms of this model. Using deformable models (snakes) for image segmentation leads us to the reconstruction problems which are inherently inverse problems which tend to be mathematically ill-posed. However, through regularization, they may be reformulated as well-posed variational principles. The Tikhonov regularization employs a specific class of so-called stabilizing functional to restrict admissible solutions to the space of smooth functions.

To better understand the Tikhonov regularization, we should consider the example given below for reliably estimating point derivatives. Example: Let $v_x(x_i)$ be a function of $v(x)$ given only a certain approximate data sequences $\{x_i, v_i\}_{i=1}^N$ are assumed to be independent, normally distributed random variables with zero means and variance σ^2.

Difficulties arise because no matter how small the errors ϵ_i, the differences between the true point derivatives $v_x(x_i)$ and the numerical derivatives of data say $(v_i - v_{i-1})/h$ can be arbitrarily large. Since one can not guarantee that the solution will be stable with respect to small perturbations of the data, numerical differentiation unlike numerical integration is an ill-posed problem. The above problem may therefore be approached through regularization [20].

Tikhonov proposed a general stabilizer for univariate regularization, so that the pth order weighted Sobolev norm is given as [14]:

$$\|v\|_p^2 = \sum_{m=0}^{p} \int_{\mathcal{R}} w_m(x)(\frac{d^m(v(x))}{dx^m})^2 dx \tag{2.3}$$

where the sequences $\{w_m(x)\}$ are prescribed nonnegative and continuous weighting functions. The stabilizer for the snake, as will be shown in the next section is a particular norm $\|v\|_p^2$ for $p = 2$ and $w_0 = 0$, $w_1 = \alpha$ and $w_2 = \beta$.

2.4.1. The Dynamics

Consider a deformable curve $v(s,t)$, with parameters s (spatial index) and t (time), defined on given open intervals Ω and T, respectively. The deformable curve is a function of two coordinate variables x and y with the same parametrization is as defined:

$$v(s,t) = (x(s,t), y(s,t)) : s \in \Omega, t \in T$$

The snake has two deformable degrees of freedom in the plan, that is, in x and y coordinates resepectively. The potential energy function of the snake $E_{snake}(v)$ is defined as:

$$E_{snake}(v) = \frac{1}{2} \int_{\Omega} [E_{int}(v) + E_{ext}(v)] ds \tag{2.4}$$

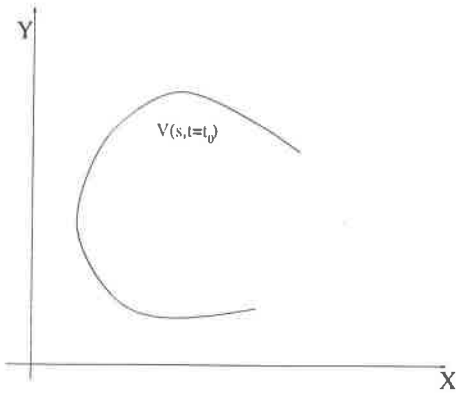

FIGURE 2.1. Snake structure

As it is seen clearly the integration is taken along the path Ω. E_{ext} gives rise to external constraint or gravitational field forces, respectively. E_{int} represents the internal potential energy of the snakes; and it is defined as follows:

$$E_{int}(v(s)) = \alpha(s) \mid v_s \mid^2 + \beta(s) \mid v_{ss} \mid^2 \tag{2.5}$$

where we define $v_s \equiv \frac{\partial v}{\partial s}$ and $v_{ss} \equiv \frac{\partial^2 v}{\partial s^2}$ respectively. The first term $\alpha(s) \mid v_s \mid^2$ forces the snake points to behave in a *string* like movement whereas the second term $\beta(s) \mid v_{ss} \mid^2$ forces them to line up on a *rod* . The weight $\alpha(s)$ regulates the tension of the snakes, whereas $\beta(s)$ regulates its rigidity. Position or tangent discontinuities may be introduced along the snake by setting these weights to zero.

The external potential energy E_{ext} may arise from two complementary types of forces: pulling and pushing. It is suggested that the terms *spring* and *volcano* to model these two forces. A spring is created by defining tension force between a snake element and a selected point on the potential surface. A volcano is created by locally deforming the potential surface. Springs are typically used to force the snake

to attach to the desirable feature, such as the curvature extrema of a contour, whereas the volcanos are used to push the snake away from undesirable local minima of the potential field, like those due to noise or isolated edges.

By adding the coefficients 1/2 to the terms, the equation 2.4 can be simplified as follows:

$$E_{snake}(v) = \int_{\Omega} E_{ext}(v) + \frac{1}{2}(\alpha(s) \mid v_s \mid^2 + \beta(s) \mid v_{ss} \mid^2) ds \tag{2.6}$$

Representing the integrand by $E_{snake} = F(s, v_s, v_{ss})$, the Euler-Lagrange necessary condition is obtained as (Details are found in the Appendix A)

$$F_v - \frac{\partial}{\partial s} F_{v_s} + \frac{\partial^2}{\partial s^2} F_{v_{ss}} = 0. \tag{2.7}$$

Substituting the terms in the equation above one obtains a pair of independent Euler-Lagrange equations,

$$-\frac{\partial}{\partial s}(\alpha(s) x_s) + \frac{\partial^2}{\partial s^2}(\beta(s) x_{ss}) + \frac{\partial(E_{ext}(v))}{\partial x} = 0 \tag{2.8}$$

$$-\frac{\partial}{\partial s}(\alpha(s) y_s) + \frac{\partial^2}{\partial s^2}(\beta(s) y_{ss}) + \frac{\partial(E_{ext}(v))}{\partial y} = 0 \tag{2.9}$$

Given the potential energy function E_{snake} for a specific initial position, a min-

imization procedure can be applied to reach a more stable energy level. The minimization procedure must transform the potential energy to kinetic energy by using Newtonian approach as:

$$mx_{tt} + \gamma x_t + f_{int} = f_{ext} \tag{2.10}$$

where γ is the dissipation coefficient and m is the mass. In the Eq. 2.10 the system will be stable when internal force and external force are equal. Consequently the Eq. 2.8 and 2.9 can be thought of as the sum of external and internal forces. The iterative solution of these equations becomes:

$$mx_{tt} + \gamma x_t - \frac{\partial}{\partial s}(\alpha(s)x_s) + \frac{\partial^2}{\partial s^2}(\beta(s)x_{ss}) = -f_x(v) \tag{2.11}$$

$$my_{tt} + \gamma y_t - \frac{\partial}{\partial s}(\alpha(s)y_s) + \frac{\partial^2}{\partial s^2}(\beta(s)y_{ss}) = -f_y(v) \tag{2.12}$$

where $x_t \equiv \frac{\partial x}{\partial t}$, $y_t \equiv \frac{\partial y}{\partial t}$, $x_{tt} \equiv \frac{\partial^2 x}{\partial t^2}$, $y_{tt} \equiv \frac{\partial^2 y}{\partial t^2}$, and $f_x \equiv \frac{\partial}{\partial x}(E_{ext})$, $f_y \equiv \frac{\partial}{\partial y}(E_{ext})$

Fixing $m = 0$ does not cause any problems in our approach. The system will reach a more stable energy level in that situation too.

2.4.2. Potential Field

Snake deformations are dependent on both internal forces and external forces. Internal forces act like dynamic constraints of snakes, which deform according to external forces.

External energy can be called potential field. External forces must meet the requirement of gathering information from data (edge points). Edge operator when applied to a smoothed image, gives disperse set of edge points. In an edge image, the snake does not know how to detect the image edge points. So the potential field somehow must provide information to snakes about edge locations. That is, an edge should be recognizable by snakes.

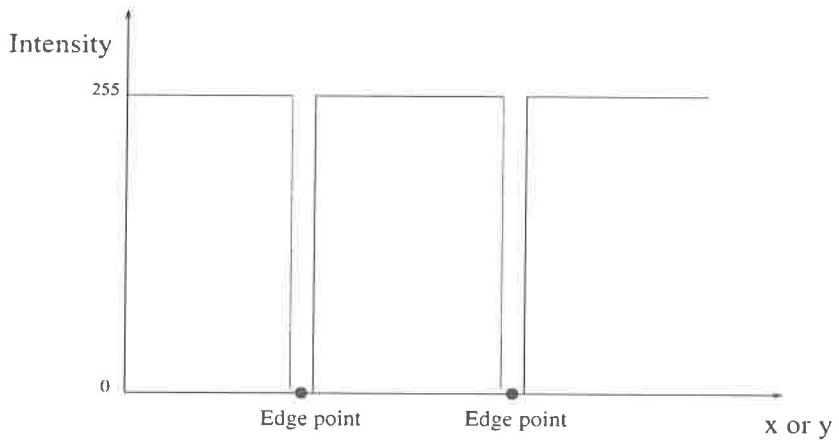

FIGURE 2.2. 1-D aspect of an image after edge detector

In order to obtain a good localization of edge points and to reduce the noise effect, first the image is smoothed by a smoothing operator. Than by an edge operator, we can extract image boundaries. In Figure 2.2, a representation of an edge image structure is shown. Here, after the edge detecting and tresholding, the negative of the intensity image has been taken. Now, a potential field can be obtained by using image edge information. One of the solutions is to apply again a smoothing operator to an

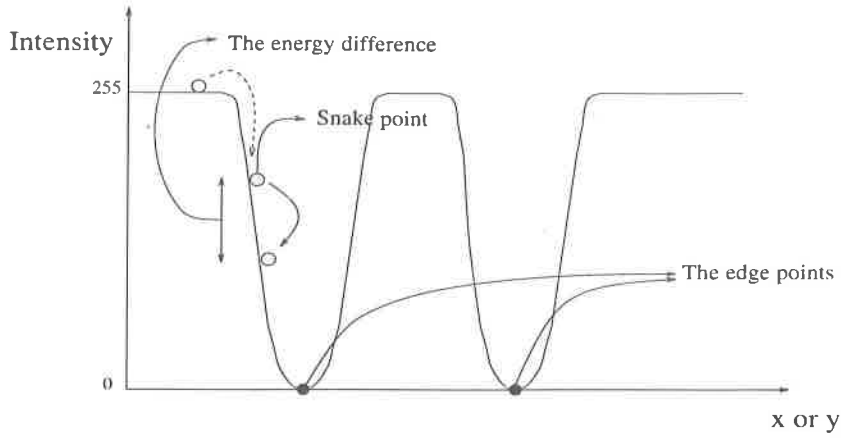

FIGURE 2.3. 1-D aspect of an image after smoothing in order to get potential field

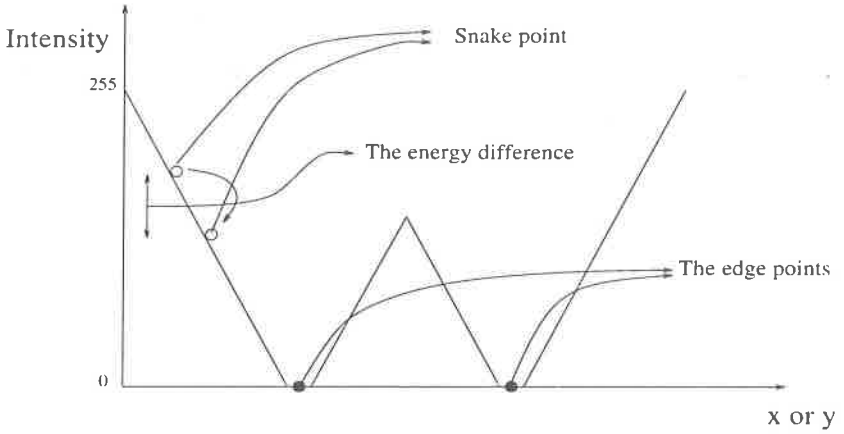

FIGURE 2.4. 1-D aspect of an image after Euclidean distance operator in order to get potential field

edge image. As seen in Figure 2.3, a smoothing operator, in some sense , forms hills in the image plane. As the snake point reaches the curvature point , it starts sloping downwards and comes to the desired edge point.

The smoothing operator does not give satisfactory results and doesn't form large scaled hills. When the snake is located at a high plane region , it is not attracted by external forces that are produced by the potential field. One of the ways for obtaining large scaled hills is to convolve the edge image by a large size window smoothing operator, which inceases the computational complexity.

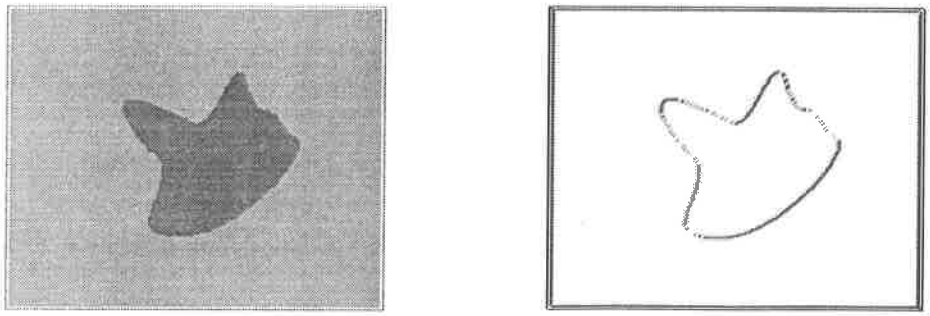

FIGURE 2.5. The original image (left) and its edges (right)

Euclidean distance operator serves better than the smoothing operator. It is much less time consuming and one can obtain easily large scaled hills. In Figures 2.7 and 2.8 the potential field of the image shown in Figure 2.5 is depicted. In the Appendix-C , the Euclidean distance operator is fully explained [17].

The derivatives of energy field give the forces applied to each snake point. Furthermore the derivatives of the energy field, that is obtained from Euclidean distance operators are constant everywhere (with - or +). Therefore, it is more suitable to obtain derivatives that have large values when the the snake point is close to an edge point, and small values when it is too far from an edge point. So we can put the potential field in a different form as follows:

$$P(x,y) = Q(I_{euclidean}(x,y)) \tag{2.13}$$

where $I_{euclidean}(x,y)$ is the intensity image that comes from a Euclidien distance operator, and Q is a sigmoid function:

$$Q(x) = 2I_{max}(\frac{1}{1+e^{-x/\sigma}} - \frac{1}{2}) \tag{2.14}$$

The constants σ and I_{max} are defined to be a smoothing factor and amplitude of the function Q respectively.

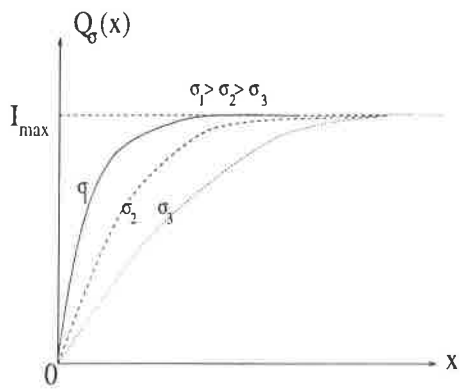

FIGURE 2.6. The function $Q(x)$

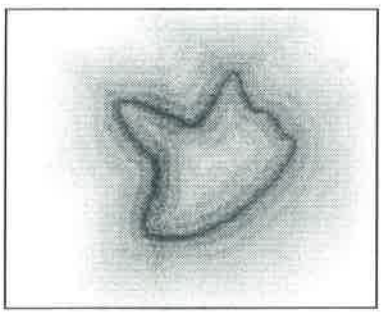

FIGURE 2.7. Potential field

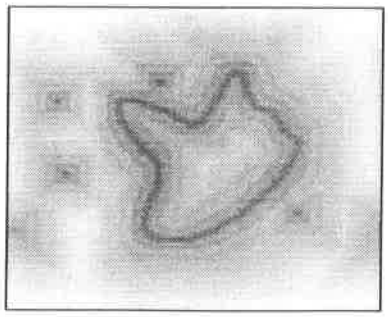

FIGURE 2.8. Potential field (from noisy image)

2.5. Discretization

Discretization of the snake is performed in two domains: Space and Time. The spatial discretization is done by sampling the deformable curve v or the interval $\Omega \in M_{\overline{s}}$ nodes, leading to a discrete set $\overline{\Omega}(i = \overline{s} \in \overline{\Omega}) = \{1, ..., M_{\overline{s}}\}$. The distance between successive snake elements is denoted by $h(\overline{s})$. For an initial snake, this distance is usually fixed to be costant: $h(\overline{s}) = h = \frac{1}{M} \int_{\Omega} \mid v_s \mid ds$. The time discretization is similarly achieved and defined to start at time 0. It consists of value $\overline{t}$ regulary separated by a constant time step $\Delta\overline{t}$: $\{\overline{t} \in \overline{T} : \overline{T} = \{0, \Delta\overline{t}, 2\Delta\overline{t}, ...\}\}$. This discretization of time is obviously to perform the numerical integration of the evaluation equation.

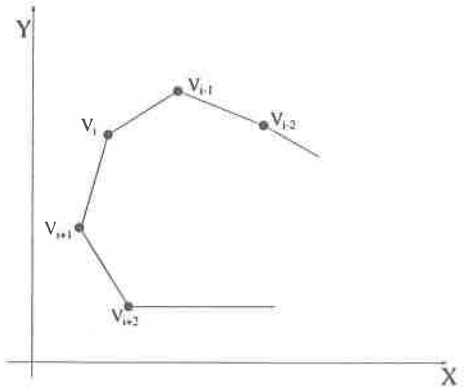

FIGURE 2.9. Discrete snake structure

To solve numerically, the Euler equations 2.12 with $m = 0$, $\gamma = 1$ and assuming that $\Delta\bar{t} = 1$, one obtains:

$$x_i^{\bar{t}+1} = x_i^{\bar{t}} - \alpha_i(x_i^{\bar{t}} - x_{i-1}^{\bar{t}+1}) + \alpha_{i+1}(x_{i+1}^{\bar{t}} - x_i^{\bar{t}}) - \beta_{i-1}(x_{i-2}^{\bar{t}+1} - 2x_{i-1}^{\bar{t}+1} + x_i^{\bar{t}}) + 2\beta_i(x_{i-1}^{\bar{t}+1} - 2x_i^{\bar{t}} + x_{i+1}^{\bar{t}}) - \beta_{i+1}(x_i^{\bar{t}} - 2x_{i+1}^{\bar{t}} + x_{i+2}^{\bar{t}}) - F_x \quad (2.15)$$

$$y_i^{\bar{t}+1} = y_i^{\bar{t}} - \alpha_i(y_i^{\bar{t}} - y_{i-1}^{\bar{t}+1}) + \alpha_{i+1}(y_{i+1}^{\bar{t}} - y_i^{\bar{t}}) - \beta_{i-1}(y_{i-2}^{\bar{t}+1} - 2y_{i-1}^{\bar{t}+1} + y_i^{\bar{t}}) + 2\beta_i(y_{i-1}^{\bar{t}+1} - 2y_i^{\bar{t}} + y_{i+1}^{\bar{t}}) - \beta_{i+1}(y_i^{\bar{t}} - 2y_{i+1}^{\bar{t}} + y_{i+2}^{\bar{t}}) - F_y \quad (2.16)$$

where F_x and F_y are the x and y components of $\vec{F}$ respectively (in the Equation 2.20). In the Equations 2.15 and 2.16 α and β are related to the spacial index $i = \bar{s}$, but we can assume that they are equal to a constant value if the image boundaries are not very complex.

2.5.1. The Weighting Coefficients α and β

The parameters α and β provide some sort of control upon *intrinsic behavior* of the snake with respect to the tension and rigidity. In order to mimic a physical behaviour, it is natural to define α as a function of the distance between snake point, and β as a function of curvature in the neighborhood of each snake point.

Another subject that must be examined with respect to the parameters α and β is the *discontinuity*. There are two types of discontinuities: The tangent and the position discontinuities. The former corresponds to fixing $\beta(s)$ to 0 and the later

corresponds to fixing $\alpha(s)$ to 0. In some cases introducing those discontinuities is very useful. For example the position discontinuity can be used for obtaining an open curve. On the other hand, tangent discontinuity can be used with the intention to being able to track on fit sharp corners. Tangent discontinuities are useful when the snake is closer to the data or when the corners are sharp, but they require an interpretation stage that recognizes the need for introducing such breaks. For example , when a snake reaches a stable stage, by observing the differences between image edges and snake position , interpretation stage can decide to introduce tangent breaks at the required locations.

These parameters can depend on s by defining some metrics L and C which are called *the natural arc length* and *the natural curvature* respectively. L is used to prescribe the "desirable" length of snake. Then, the tension parameters α can be fixed with respect to L as follows:

$$\alpha_{snake} = k(\sum_{\bar{s}=1}^{M} l(\bar{s}) - L) \tag{2.17}$$

where $l(\bar{s}) = \sqrt{\Delta x(\bar{s})^2 + \Delta y(\bar{s})^2}$ is the actual distance between successive snake points, k is the normalization factor and M is the number of snake points that need to be determined to attain a given image boundary precision. Since the snake is always trying to minimize its energy and in particular, $E_{int}(v)$, when the total length of the snake is big α is always positive and the snake should shrink. On the contrary when the snake is too short , α is negative and the snake should expand. We can redefine α as a weight that affects to each snake points respectively;

$$\alpha(\overline{s}) = k(l(\overline{s}) - \frac{L}{M}) \tag{2.18}$$

When snake points are too much apart from each other , $\alpha(\overline{s})$ is positive and the snake should shrink. On the contrary, for snaxels that are too close to each other , $\alpha(\overline{s})$ is negative and the snake should expand. C is used to define the desirable curvature at each snake point. Then, the rigidity parameters β can be fixed as follows:

$$\beta(\overline{s}) = k(c(\overline{s}) - \frac{C}{M}) \tag{2.19}$$

where $c(\overline{s})$ is the approximated curvature at snaxels $\overline{s}$. There are advantages as well as disadvantages that arise from this kind of updating α and β. At each iteration, α and β is modified for all snake points and this can cause oscillation at the value of L and C, yielding some unrealistic snaxel displacement. Some techniques proposed by Leymarie and Levine [13] can be used such as clipping the high values of α and β in order to limit the amplitude that they can take.

2.5.2. The Balloon Model

To force the snake find its way, an initial guess of the contour has to be provided manually. This has many consequences on the evolution of the curve.

*If the snake is not close enough to an edge, it is not attracted to the edge point.

*If the snake is not effected by any counterbalancing forces, it tends to shrink on itself.

Accordingly, an inertial pressure by considering the curve as a *balloon*, which is inflated, is introduced [5]. The *pressure* force is added to the internal and external forces to push the curve outward. The curve expands and is attracted to edges as before. But if the edge is too small or too weak with respect to the pressure force, the curve passes over the edge, growing outward.

The force f now becomes

$$\vec{F} = k[f_x, f_y] - k_1\vec{n}(s) \tag{2.20}$$

where $\vec{n}(s)$ is the unit vector normal to the curve at point $v(s)$ and k_1 is the amplitude of the force. The coefficient k_1 and k are chosen in such a way that they fall into the same level, which is smaller than a pixel size (the length unit).

Note that F depends on not only the position of $v(s)$, but also on the normal at this position.

2.6. Comments

Deformable models such as snakes have some similarities with deformable templates. However, there is an important difference. All the deformable models have forces which interact with the image and other forces that prevent the structure from deforming too much. This means that snakes are too floppy. In contrast, the structure

forces on the deformable templates are global (as big as the template) and interactions are long range. A deformable template can only be used when a priori knowledge about the shape of the feature or range of parameter values is known. Snake models can be thought of as a deformable template in the limit as the number of parameters goes to infinity.

An advantage as well as a limitation of the snake models emerges from global way in which an optimal solution is evaluated. This approach to the segmentation problem makes the active contour or snake seek a global minimum of its energy function. It offers the advantage of integrating information about the derived potential surface features along the entire trajectory of the closed snake. This is implemented by seeking a global minimum of this snake energy function.

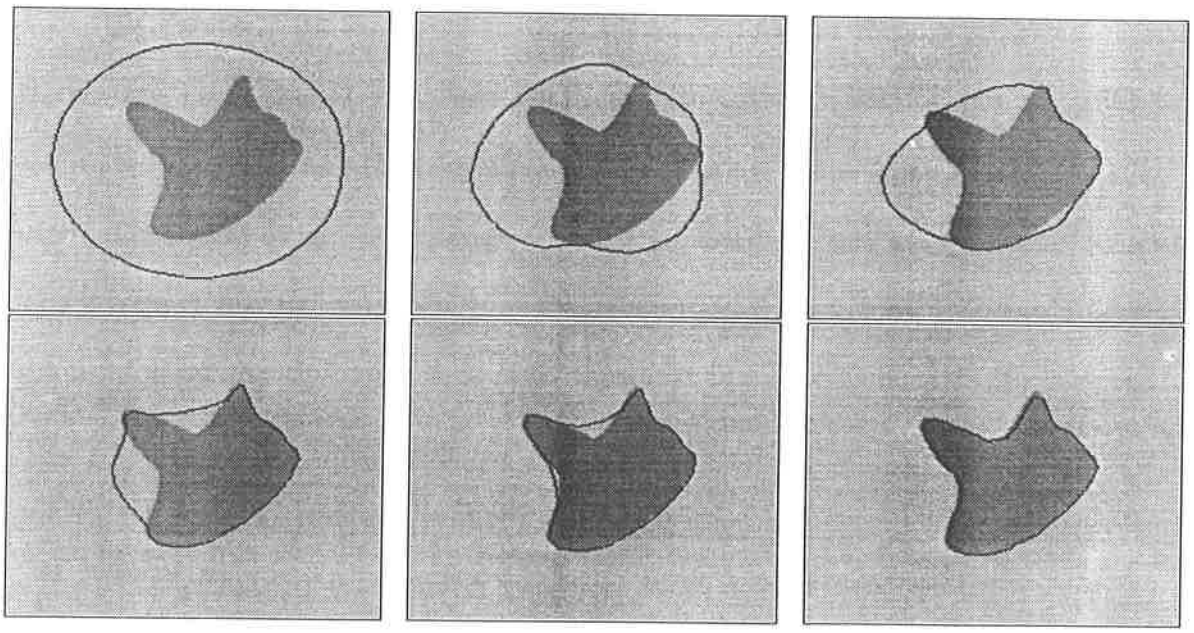

FIGURE 2.10. Snakes iteration

The snake model suffers from the following major difficulty : The snake has a bias toward solutions that reduce its length, that is, the snake naturally tends to shrink. This negative effect has been reduced by using different approaches for updating α and β parameters and balloon model.

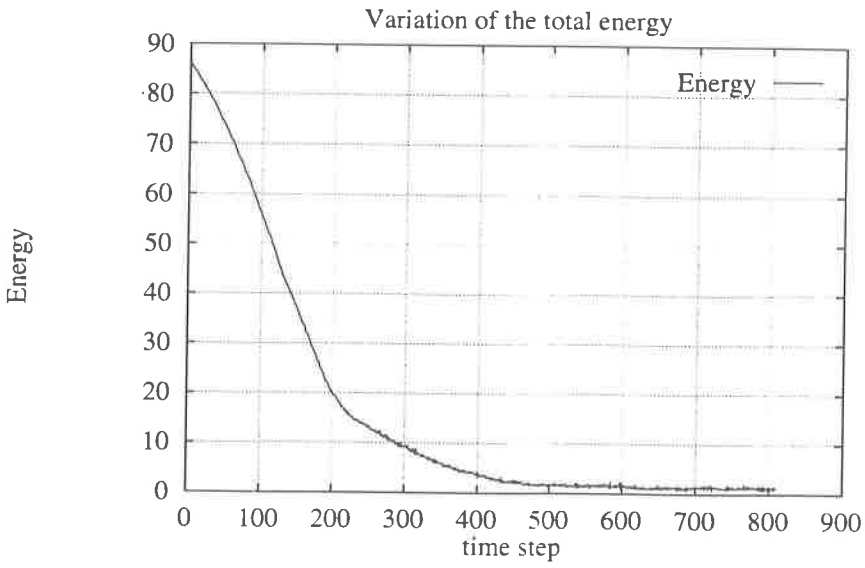

FIGURE 2.11. The total snake energy as a function of iteration

In Figures 2.10, the convergence of the snake to an artificially produced object is shown. The snake is initialized not too close to the object location in order to see its trajectory during the convergence. In the image plane there is no noise effect used. The snake weight parameters α and β are fixed to constant values $+0.02$ and $+0.02$ respectively and the number of snake points is 30 for a good precision of object shape. Approximately in 600 time steps the snake finds the global energy level which represent the desired image boundary. In Figures 2.11, 2.12 and 2.13, the total, internal and external energy level of the snake are shown.

The average external energy per snake point is derived as follows:

$$E_{ext} = \frac{1}{M} \sum_{i=1}^{M} P(v_i) \tag{2.21}$$

where $P(v_i)$ is the intensity value of the potential image that is obtained from the edge image. The average internal energy per pixel is obtained as follows;

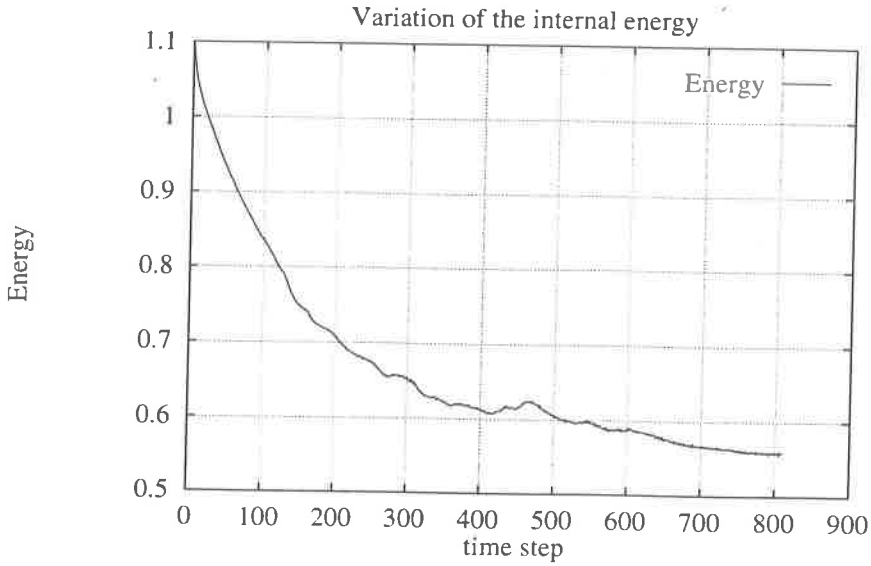

FIGURE 2.12. The internal snake energy as a function of iteration

$$E_{int} = \frac{1}{M}\sum_{i=1}^{M} \alpha \mid v_i - v_{i-1} \mid^2 + \beta \mid v_{i+1} - 2v_i + v_{i-1} \mid^2 \qquad (2.22)$$

Augmenting the values of weight parameters increases the smoothing constraint effect, and this causes the snake to be indifferent to the sharp corners. Decreasing α causes the snake points to be dispersed irregularly on the snake contour. β makes the snake points became condense on the sharp corners but if it is fixed to a large number, the external force becomes ineffective on the snake and the model can not find the image boundaries.

In figures 2.14 the convergence of the snake in a noisy image is seen. The noise, that is applied to all pixels is a gaussian noise with mean $m = 0$ and variance $\sigma^2 = 10$. The image is first smoothed by a smoothing operator in order to reduce the noise effect, then preprocessing operators (edge operators, distance operators) are applied

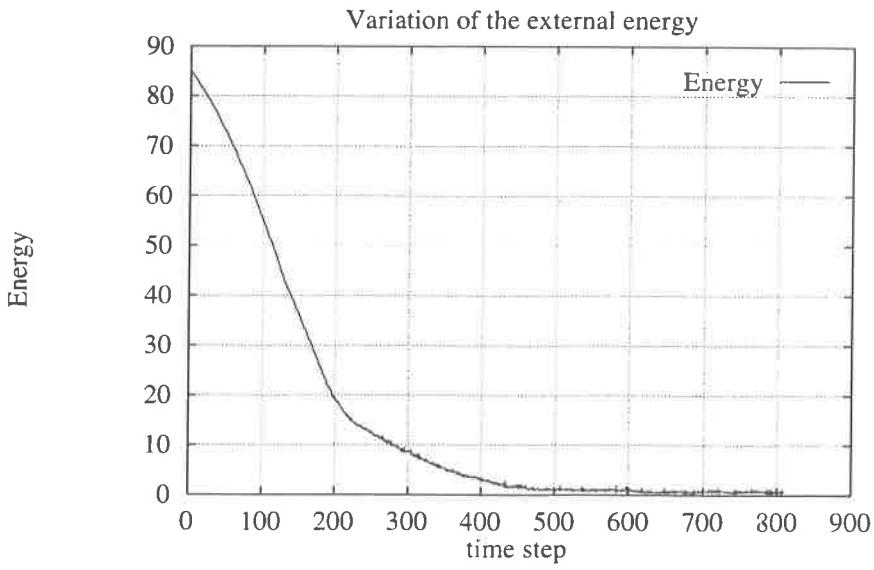

FIGURE 2.13. The external snake energy as a function of iteration

to the image. α and β are fixed to $+0.07$ and $+0.08$ respectively in order to force the snake to pass through the local minimum . The snake is initialized as the previous experiment. At first, it converges to a local minimum, where the snake finds the noisy points supposing that they are edges (In Figure 2.15. Iteration upto 800) Then, by the effect of internal forces, it passes through the local minimum down to the global minimum. But, because of the large values of the internal forces the snake becomes a smoothed contour model and can not find sharp corners.

2.7. Experiment on a sequence

The snakes are useful for an image sequence because of its dynamic behavior. Theoretically, they can easily catch any small deformations of a moving object in the image plane. Consequently, important features such as eyes and lips can be tracked by snakes.

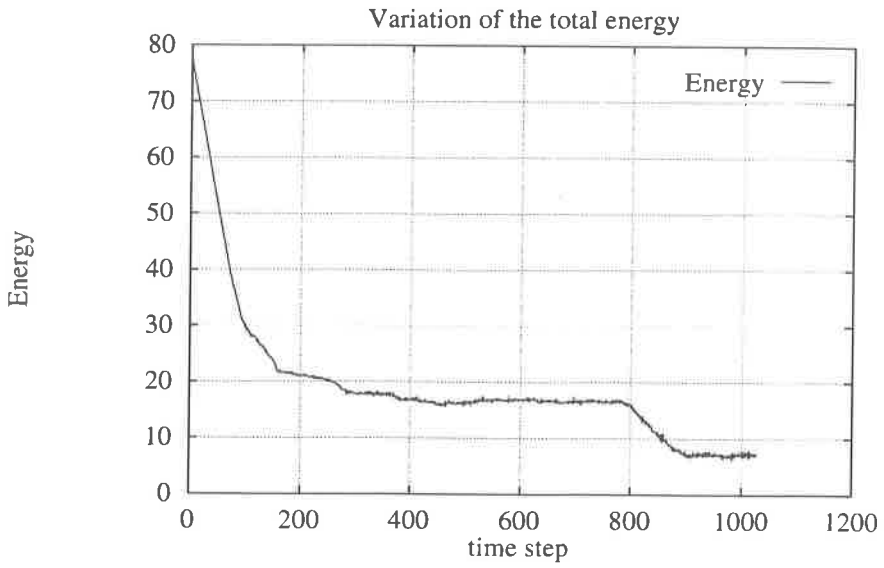

FIGURE 2.15. The total snake energy as a function of iteration (with noise in the image)

The experiments are implemented on the Miss America sequence which consists of 76 frames, with 360x288 resolution.

Observations throughout the experiment lead to the fact that snakes are too sensitive to the thickness of the edge points. Spurious external forces created by the thick edges cause the snakes to go astray with respect to internal forces, and consequently, mismatched paths can be produced. In order to achieve better results , the edge operator that is used before the Euclidean distance operator must be a powerful one. The Canny edge operator gives the optimum result in finding edges. Yet the undesired edge point causes the snake to move towards itself and consequently undesired results can be obtained.

In order to find the boundaries of the eye (as shown in the Figures 2.18- 2.19), Sobel type edge operator gives the most reasonable result, whereas the horizontal edge operator provides the best result in finding the boundaries of the lips.

Another difficulty arises when large displacements of feature happen between

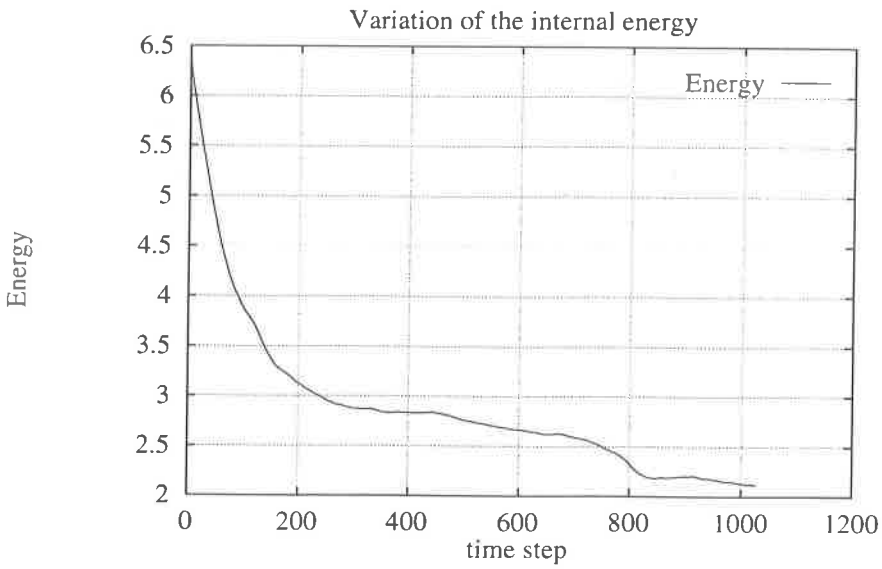

FIGURE 2.16. The internal snake energy as a function of iteration (with noise in the image)

frames. For example, if the eye location moves approximately more than 6-7 pixels per frame, the snake may come across to the attractive potential fields of the pupil or eyebrow. This negative effect can be removed by using snake's *center of gravity* that is defined below:

$$g(x, y; t = t_0) = \frac{\sum_{i=1}^{M} v_i(t = t_0)}{M} \tag{2.23}$$

where M is the number of snake points. Using Taylor expansion so that $g(t + \Delta t) \approx g(t) + g'(t)\Delta t/1 + g''(t)\Delta^2 t/1.2 + ...$ and assuming that frames are in continuous domain, the next center of gravity can be estimated as follows:

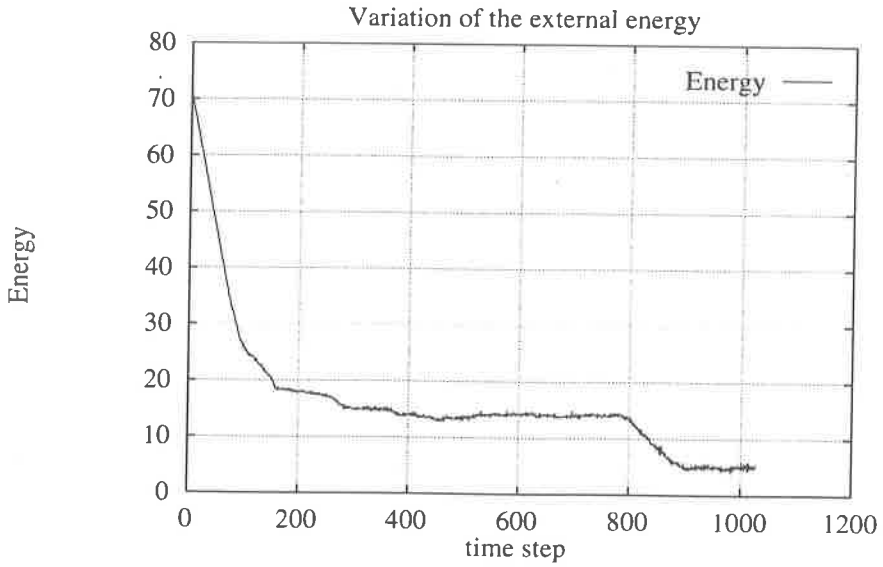

FIGURE 2.17. The external snake energy as a function of iteration (with noise in the image)

$$g_e(\bar{t} + \Delta\bar{t}) = (g(\bar{t}) - g(\bar{t} - \Delta\bar{t})) + g(\bar{t}) \tag{2.24}$$

where g_e is the estimated center of gravity. The estimation of *center of gravity* for the next frame $\bar{t} + \Delta\bar{t}$, can prevent the snake from falling into the potential field of the unrequired edges.

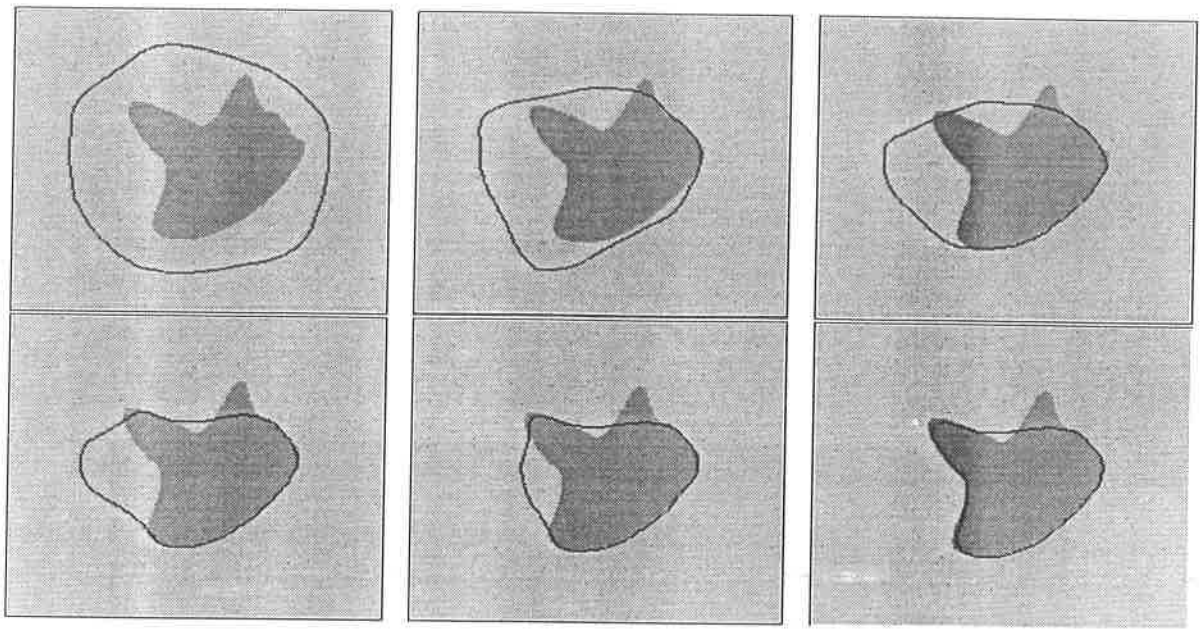

FIGURE 2.14. Snakes iteration (there is additive gaussian noise in the image)

FIGURE 2.18. Snakes for eyes and lips location (1. frame)

FIGURE 2.19. Snakes for eyes and lips location (63. frame)

3. Motion

3.1. Introduction

The dynamic evolution of images in time domain provides important information, such as motion parameters. Estimated motion parameters play an important role for different purposes in computer vision research such as intelligent robotic systems, or model-based image coding . Model-based image coding is related to the subject of this thesis and we will focus on 3-D motion estimation problems.

3.2. Projection and Coordinate Systems

A 3-D transformation means that a point (x, y, z) is rotated by θ_x, θ_y and θ_z around x, y, z axes respectively, and translated by T_x, T_y and T_z along x, y, z axes respectively to obtain a point (x', y', z'). This leads us to three nonlinear equations that are difficult to solve and computational expensive.

$$R_z = \begin{pmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad R_y = \begin{pmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{pmatrix}$$

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{pmatrix} \tag{3.1}$$

$$R = R_x R_y R_z \tag{3.2}$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} \tag{3.3}$$

Object-space lies on 3-D coordinates (x, y, z) and a point place is determined by them. In a camera system, the point is projected to a plane as shown in Figure 3.1. The mathematical formulation of the projection is as follows:

$$X = f_c \frac{x}{f_c - z} \tag{3.4}$$

$$Y = f_c \frac{y}{f_c - z} \tag{3.5}$$

where f_c is the focal length, (x, y, z) are object-space coordinates and (X, Y) are image-space coordinates as it is seen in figure 3.1. The focal length is an important parameter which indicates whether it is a *orthographic projection* or *perspective projection.* The former is the special case of the later one when the focal length goes to infinity. While the perspective projection is applied , the depth information can be extracted easily from its focal length information. But when the orthographic projection is applied , we can't extract the depth information because the focal length is indefinite. Consequently it is necessary to focus on how to obtain important information about rigid motions from the differences between the two image frames. So by using the displacement of a point, assuming that the object makes a rigid motion (not local displacement),

information about three dimensional parameters (Rotation and translation parameters) can be obtained. In teleconferences and video-phones system the projection to the camera plane is generally orthographic. Consequently we will focus on this case.

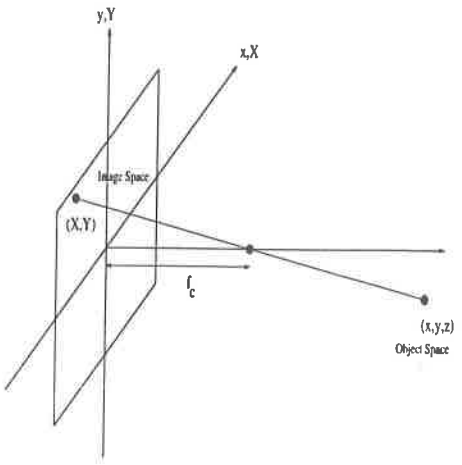

FIGURE 3.1. Basic geometry for three-dimensional motion estimation

3.3. Previous Work

Most of the previous works are about the analytical approach of obtaining 3-D motion parameters. Substantial amount of these works has been devoted to methods for estimating object motion based on a short sequence.

Generally motion parameters are estimated in between two consecutive frames. In order to estimate the motion parameters, it is necessary to solve the correspondence problem. There are two substantial approaches about this problem that most of researchers focus on: Optical flows and object based feature extraction procedures as deformable template and snakes.

Optical flow for motion estimation

Determination of optical flow has been a very active search area: Smoothed optical flow and motion field can be interpreted as vector fields tangent to flows of planar dynamical systems. Stable qualitative properties of motion field , which give useful information about the 3-D velocity field and the 3-D structure of the scene, can be usually obtained from optical flow if special conditions are satisfied . Ballard and Kimball discuss an approach using optical flow to estimate rigid motion.

Optical flows provide a large number of correpondences, but without looking if either the displacements of the points are global or local (global displacements are related to rigid motion).

Feature based motion estimation

The use of the deformable models provide an extra information about properties of the object's shape: When we use them, we certainly know which point corresponds global motion.

The work of Tsai is based of discrete features (object correspondence points). Tsai and Hung give an analysis of one approach to the motion estimating problem. Significantly, they have shown that in many cases the solution can be achieved by solving a set of linear equations, eliminating the difficulties with nonlinear search algorithm [18] [22].

John Roach and J.K. Aggarwal discusses the problem of determining 3-D model and movement of an object belonging to an image from a sequence of 2-D images. The solution of this depends on solving nonlinear equations using modified least-squares error method.[10]

A. Azabayejani and A. Pentland proposed a formulation for recursive recovery of motion, pointwise structure, and focal length from feature correspondences tracked through an image sequence. They used a Kalman filter by forming a set of parameters of the basic geometrical concepts [3] [4].

3.4. The Stage of Motion Estimation

Rigid 3-D motion of the object (in this case; head of the speaker) is estimated in two steps, namely first correspondences between 2-D features points are established and secondly 3-D motion is estimated from 2-D feature displacements.

The first step is to solve the correspondence problem and it can be implemented either via a continuous approach or discrete approach. A continuous approach allows only a small amount of interframe motion based on optical flow. A discrete approach allows relatively large interframe motion. Points (or corner and center of region), edges (or lines) contours, and locally intensity patterns can be utilized as features. Correspondences between features may be established through matching or interframe tracking.

The second stage concerns estimation of motion parameters and the structure of the scene from established correspondences. As it is seen in equation 3.3 the equations are nonlinear. Nonlinear equations generally have to be solved through iterative methods with an initial guess or through global search. Iterative methods run the risk of diverging or of converging to a local minimum. Searching in the space of motion parameters is computationally expensive. Linear algorithms solve linear equations and give a closed-form solution. The main advantages of linear algorithm over nonlinear ones are that they are fast and uniqueness is guaranteed except in degenerate cases [18].

3.4.1. Least-squares approach

One of the methods is the use of the least-squares approach which is commonly used to obtain 3-D motion parameters by locally approximating nonlinear equations as linear equations and iterating them. In order to solve this problem, the coordinate system is changed as in Figure 3.2

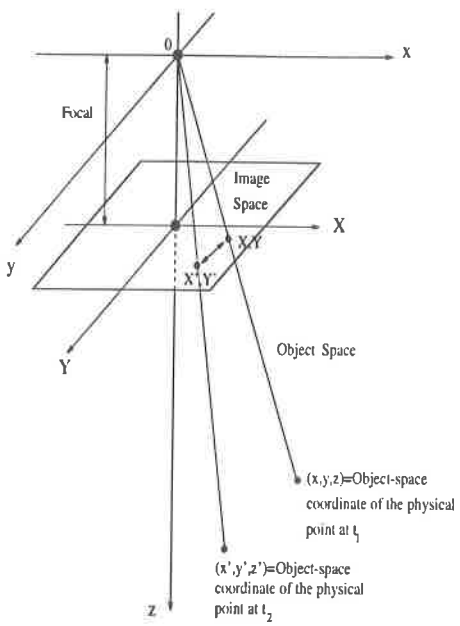

FIGURE 3.2. The variation geometry

In order to solve the estimation parameters using the least squares [18], the geometry of the coordinate system is changed as shown in Figure 3.2. Without loss of geometry we can assume that the focal length is unity. Thus the image plane is located at $Z = 1$. Then, the perspective projection on the image is given as:

$$X = \frac{x}{z} \tag{3.6}$$

$$Y = \frac{x}{z} \tag{3.7}$$

We then obtain a set of equations as follows:

$$x' = r_{11}x + r_{12}y + r_{13}z \tag{3.8}$$

$$y' = r_{21}x + r_{22}y + r_{23}z \tag{3.9}$$

$$z' = r_{31}x + r_{32}y + r_{33}z \tag{3.10}$$

where r_{ij} $i, j = 1, 2, 3$ are the elements of the linearized matrix R. By using Equation 3.5 we obtain a set of equations as follows:

$$X' = \frac{(r_{11}X + r_{12}Y + r_{13})}{(r_{31}X + r_{32}Y + r_{33})} \tag{3.11}$$

$$Y' = \frac{(r_{21}X + r_{22}Y + r_{23})}{(r_{31}X + r_{32}Y + r_{33})} \tag{3.12}$$

Tsai and Huang had worked on the estimation problem of the rotation matrix element from this point of view and they use 8 points correspondences.[18]

3.5. Using the neural network model

As neural networks are very effective in solving nonlinear equations, it is very suitable for solving the 3-D motion estimation problem. The method to estimate the 3-D motion of a speaker's head based on a head model, we use a "two layer neural network" for model-based image coding. The neural network consists of an input layer, and an output layer [11]. The input layer represents 2-D motion vectors of feature points, while the output layer represents 3-D motion parameters.

The method works as explained below:

1) Selecting five 2-D feature points (P1....P5) as in figure 3.3. The points (P1....P5) are defined P1=(X_1, Y_1),...P5=(X_5, Y_5).

2) Estimation of motion parameters using neural networks.

This method consists of two steps.

In the first step, the facial contour and feature points of the speaker are extracted using filtering techniques and the snake algorithm. Feature points on a speaker's facial image are tracked between consecutive picture frames which give 2-D motion vectors of the feature points. Then in the second step, the 3-D motion of a speaker's head is estimated using a two-layered neural network model.

The neural model for 3-D motion estimation is in Figure 3.6. There are ten nodes in the input layer (5 X and 5 Y values) , and five nodes in the output layer (θ_x,θ_y,θ_z,T_x,T_y). The number of nodes in the hidden layer is arbitrarly provided so that thay are not very few or too many.

Input signal: After detecting the corresponding feature points in the kth frame, motion vectors are calculated as follows:

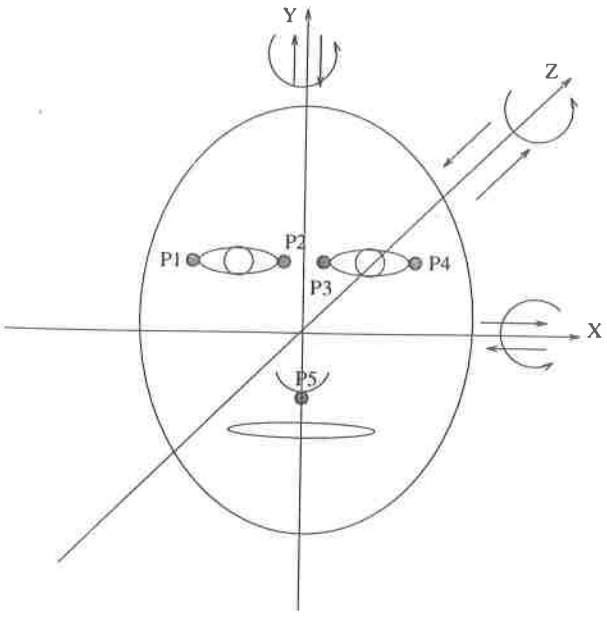

FIGURE 3.3. Feature points that will be used as a input for neural network

$$(D_x)_k^j = (X_k)^j - (X_0)^j \tag{3.13}$$

$$(D_y)_k^j = (Y_k)^j - (Y_0)^j \tag{3.14}$$

where left terms denote the motion vector of the feature point j between the initial frame and the kth frame.

In the analytical method of solving the problem of 3-D motion estimation, cumulative errors from earlier frames frequently cause problems. But, in the neural network approach , we can get around the negative effect of cumulative errors because the displacement vectors are not calculated from the differences between the consecutive frames; as seen in Eq. 3.14. They are calculated from the differences of current and initial frames.

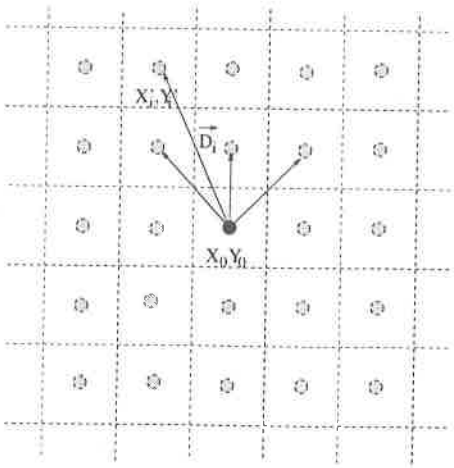

FIGURE 3.4. The displacement vectors during training

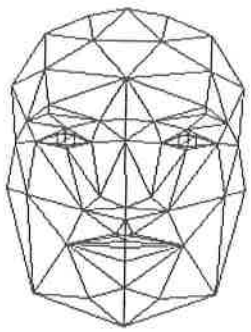

FIGURE 3.5. Candid Model

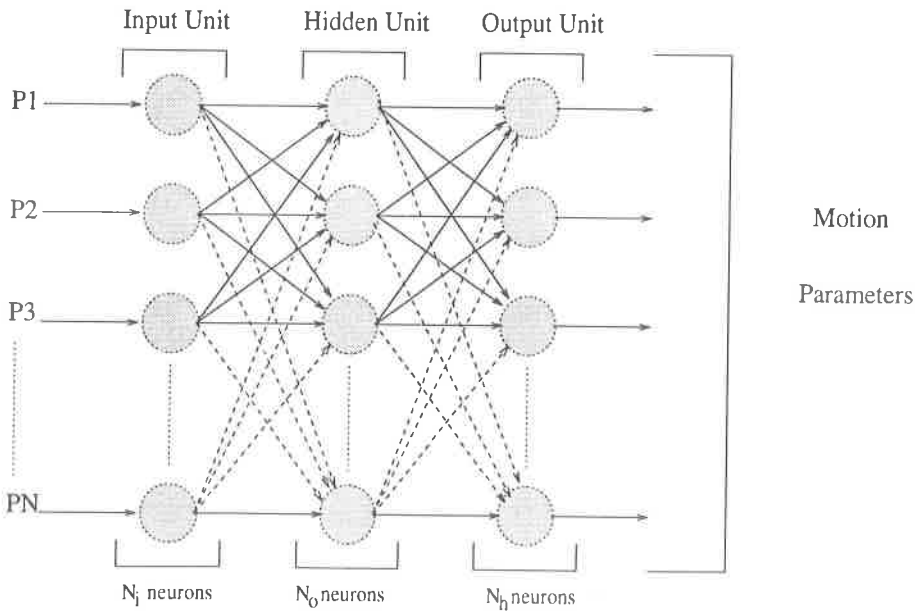

FIGURE 3.6. A typical neural network

Output signal: Output nodes of the network correspond to the 3-D motion parameters θ_x,θ_y,θ_z, T_x,and T_y.

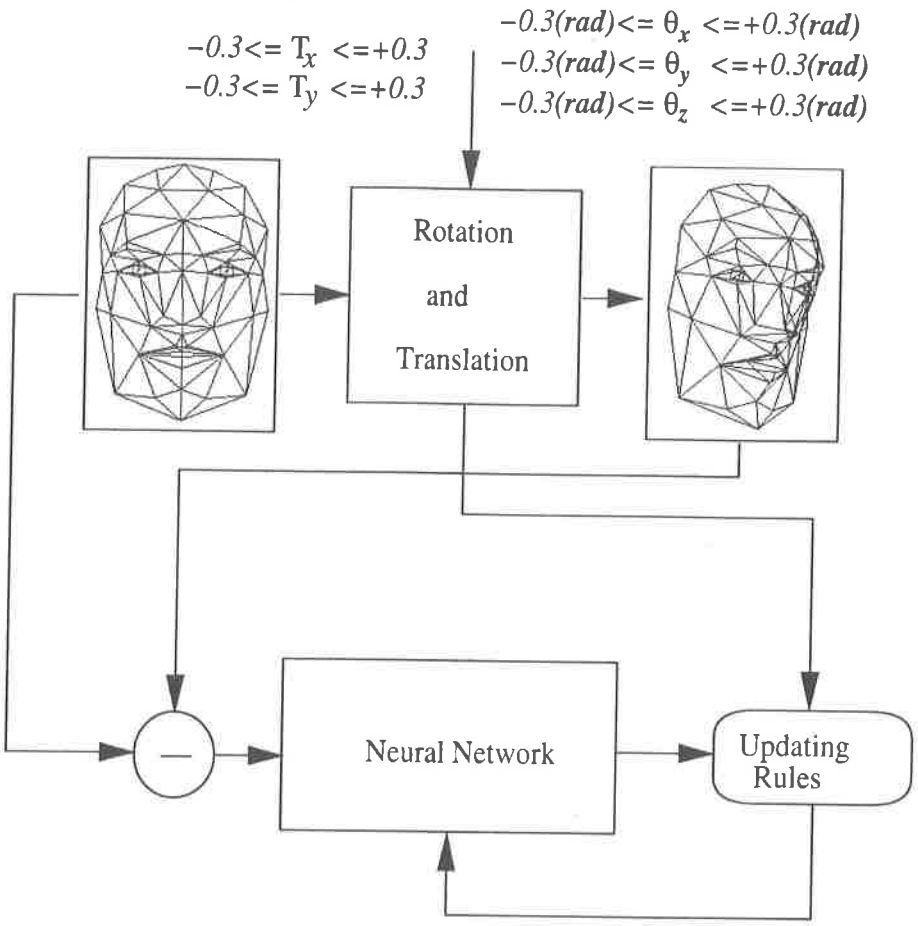

FIGURE 3.7. Block diagram of ANN training

Network Training to recognize motion: For classification, network is trained by candid model, as shown in Figure 3.5, to learn the set of noise-free normalized patterns. The procedure of training the network is shown in figure 3.7. Possible motion patterns of the human head can be randomly generated but in that situation the convergence of the neural network for training can take time. Therefore, we put patterns in order for a faster training (in each epoque, the same order). The rotation values of angles are constrained between -0.3 rad and +0.3 rad, and the normalized translation vector components between -0.3 and +0.3 rad by step 0.1 rad. As we have 5 outputs,

they are total 7^5 motion patterns [19].

Furthermore, too many motion patterns need vast calculation in learning process and consequently the estimation performance isn't better. During training, the 3-D shape model is rotated and translated according to the 7^5 motion patterns by computer calculation. Then by projecting the feature points the new co-ordinates of them are calculated to give the 2-D motion vectors which are input to the neural network.

The back propagation method for training the network has been used in which learning is performed by computing the error between the desired and actual output and feeding back this error signal level to the inputs, changing each weight inversely in propagation to its contribution to the output error.

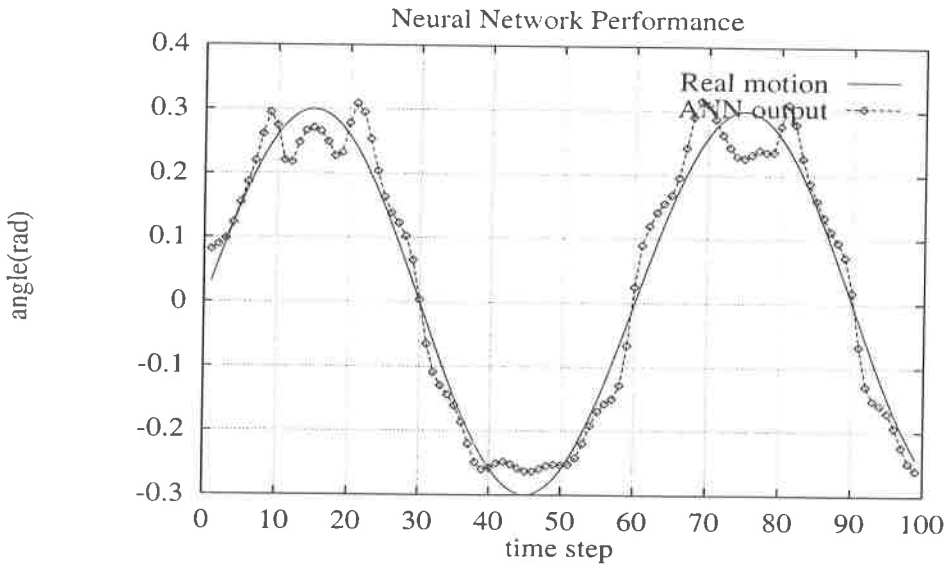

FIGURE 3.8. Synthetic motion and ANN tracking for rotation with respect to x

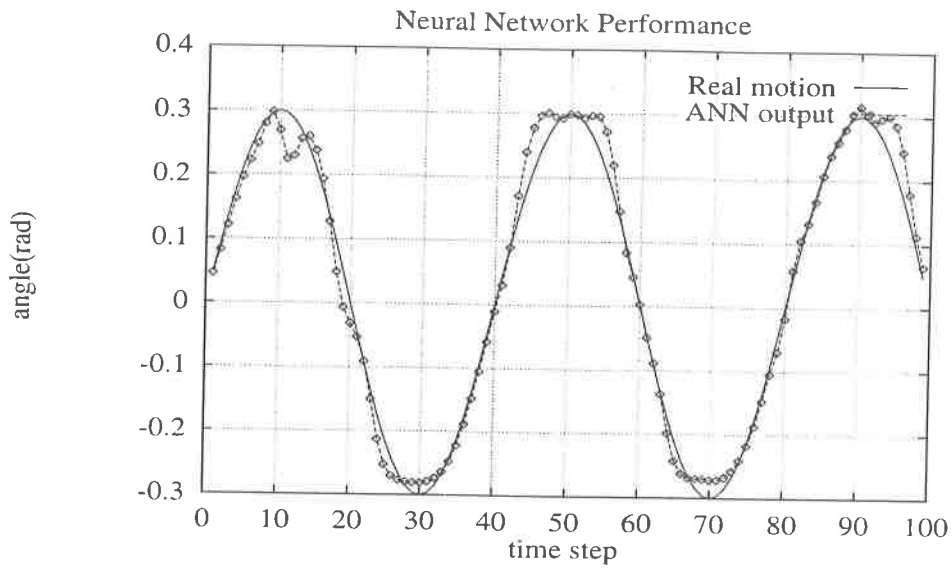

FIGURE 3.9. synthetic motion and ANN tracking for rotation with respect to y

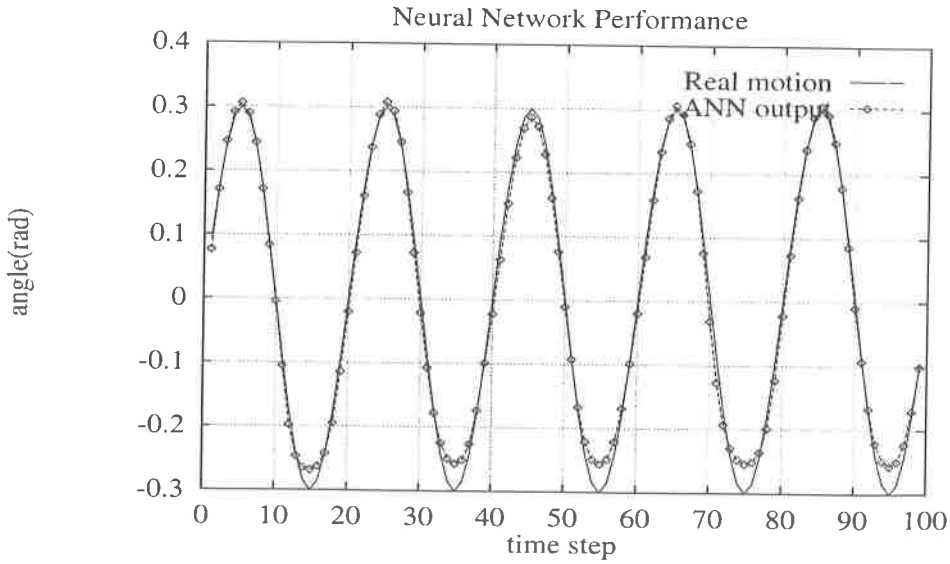

FIGURE 3.10. Synthetic motion and ANN tracking for rotation with respect to z

3.5.1. Reduction of Time Complexity for the Training of the ANN

In the first approach for training, it is assumed that rotation and translation are performed with respect to the center gravity of the candid model. So we have 5 outputs and 10 inputs and as seen in Figure 3.3 . The z translation is negligible assuming that in teleconferences the speaker does not move too much with respect to z.

As mentioned in the previous section, the desired rotation and translation parameters can take values between -0.3 rad and $+0.3$ rad by step 0.1 throughout the training. This leads us to 7 different values. Furthermore, the number of outputs is 5, as a result we have 7^5 motion patterns. ANN is trained 5000 times (5000 epoques. 1 epoque has 7^5 updatings). The training of ANN takes 36 hours using Spark 2 machine.

We may assume rotation and translation as two different geometric transformation matrices which can be applied to any object. Translation is less complex than rotation, so it is not necessary to use ANN in the estimation of translation parameters.

We can remove the translation parameters estimation from the neural network by assuming that rotation and translation are executed with respect to nose coordinates. As snakes track nose easily, the translation parameters can be obtained from x and y displacements of the nose itself. As the number of the inputs is 8 and the outputs is 3 (only rotation parameters), we have 7^3 patterns. This approach uses approximately 1/49 of the time the previous methods used for the same task.

3-D motion parameters which are estimated by neural network are shown in Figure 3.8, 3.9 and 3.10. The rotation and the translation are created sythetically. These parameters are applied to the candid model. Then the displacement vectors of the corresponding features are used as inputs of the neural network. As a result the rotation parameters are estimated in the ouput. Neural network could't estimate

the rotation parameters, with respect to x, and y coordinates very properly as shown in Figures 3.11, 3.12. That is because of the overlapping of the displacement vectors during the training procedure.

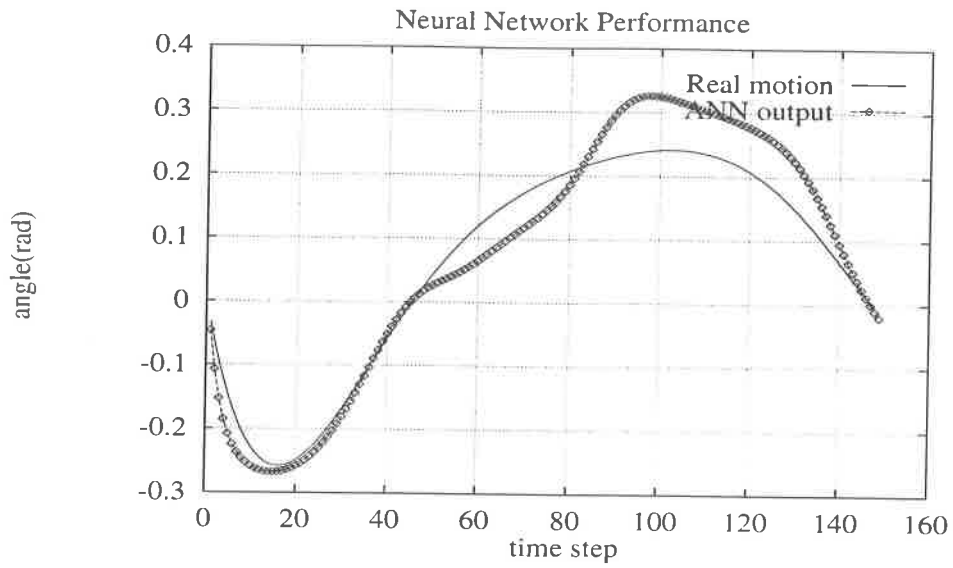

FIGURE 3.11. Realistic motion and ANN tracking for rotation with respect to x

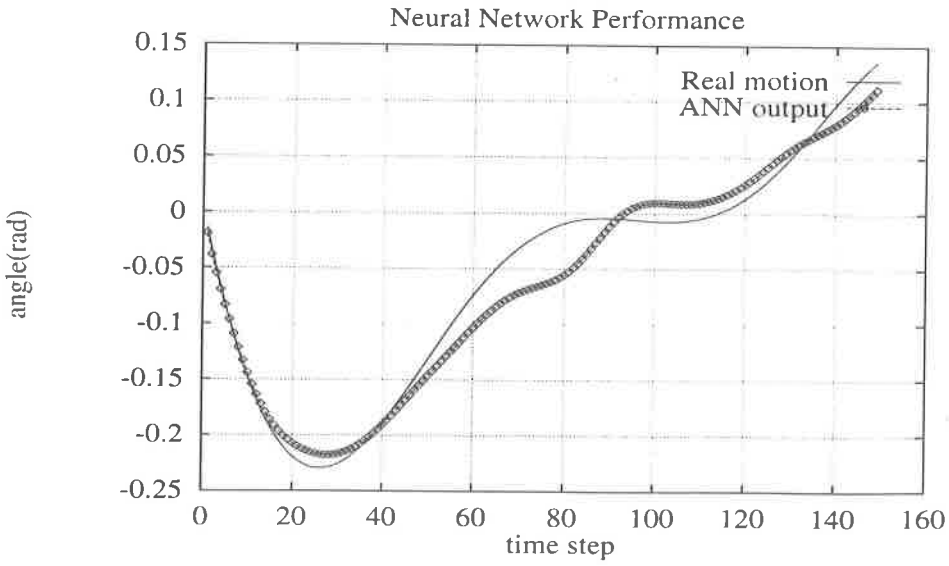

FIGURE 3.12. Realistic motion and ANN tracking for rotation with respect to y

	θ_x	θ_y	θ_z	$\theta_x^{(neural)}$	$\theta_y^{(neural)}$	$\theta_z^{(neural)}$
Frame 1	-0.032	-0.013	-0.037	-0.046	-0.018	-0.045
Frame 10	-0.228	-0.147	-0.126	-0.257	-0.143	-0.128
Frame 20	-0.248	-0.219	-0.017	-0.257	-0.207	-0.024
Frame 30	-0.167	-0.225	0.053	-0.180	-0.216	0.039
Frame 40	-0.057	-0.188	0.023	-0.050	-0.190	0.011
Frame 50	0.043	-0.131	-0.065	0.026	-0.146	-0.069
Frame 60	0.120	0.074	-0.141	0.065	-0.103	-0.143
Frame 70	0.177	-0.031	-0.152	0.117	-0.071	0.153
Frame 80	0.207	-0.009	-0.084	0.185	-0.055	-0.084
Frame 90	0.229	-0.004	0.031	0.295	-0.011	0.020
Frame 100	0.240	-0.007	0.141	0.324	0.009	0.130

Table 3..1 Numerical Results

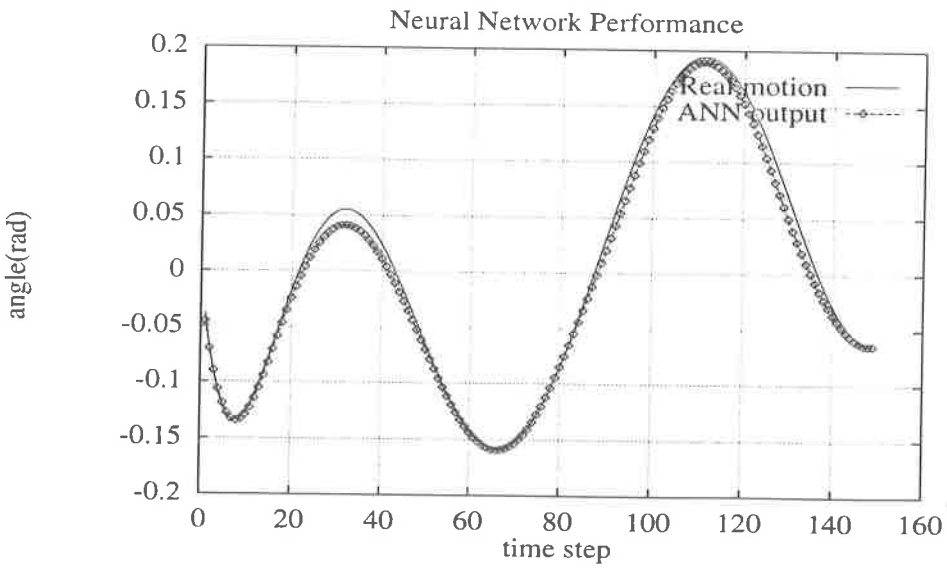

FIGURE 3.13. Realistic motion and ANN tracking for rotation with respect to z

4. Animation

4.1. Introduction

Quick, robust synthesis of facial image sequences is desirable for numerous applications. A promising application in this respect is the low-bandwith teleconferencing by model-based image coding, involve realtime animation. Teleconferencing and other applications require facial models that are not only computationally efficient, but also realistic enough to accurately synthesize the various nuances of facial structure and motion.

4.2. Previous Work

Frederic Parke's [16] face model is the seminal work in the field of human character animation. The skin is represented by polygones. While polygone shading techniques create the illusion of as smooth surface, their polygone nature is revealed in the profile and border of the face.

Keith Waters's [21] muscle model is one of the first systems to be used for the simulation of muscle activation. The muscles are defined by point attachements, assumed to be fixed to bones, and the point of insertion, attached to skin. The simulated muscles deform the polygonal skin model by moving the vertices in a manner that mimics the effect of contraction of the skin. Terzopoulos [21] developed a dynamic skin model using the same polygonal surface and muscle place.

Irfan A. Essa, Stan Scarloff and Alex Pentland's [7] geometric modeling ap-

proach is based on the use of parametrized implicit functions, particularly polynomial deformation of superquadric functions. The physical model is developed by discretizing the geometric model via the finite element method, making use of Galerkin polynomial approximation for shape interpolation. The geometric and physical models are then combined into a unified model by using the finite element model shape function as polynomial shape deformations applied to an underlying superquadric implicit function.

Recently, several physical-based models have been proposed for automatic computation of motion and deformations. Most of them are restricted to elastic material, which comes back to its rest shape after any deformation. Inspite of their well developed structures, the methods mentioned, in general have computational complexity problems, in real-time applications.

4.3. Dynamics of Elastic Models

Deformable models are based on principles of mathematical physics. They react to applied forces (such as gravity), constraints (such as linkages), ambient media (such as viscous fluids), impenetrable obstacles (such as supporting surfaces) and to physical objects .

The dynamics of deformable curves, surfaces and solids are based on elasticity theory. By simulating physical properties such as tension and rigidity, we can model static shapes exhibited by a wide range of deformable objects, including string, rubber, cloth, paper and flexible metals. Furthermore, by including physical properties such as mass and damping, we can simulate the dynamics of these objects. The simulation involves numerically solving the partial differential equations that govern the evolving

shape of the deformable objects and its motion through space.

The equations of motion govern the dynamics of the deformable model under the influence of applied forces. The equations of motion are obtained from Newtonian mechanics and balance the externally applied forces with the forces due to deformable model.

4.3.1. Coordinates Systems

We first recall some definitions and formulate the mathematical problem in order to understand the dynamics of elastic models more clearly. Let $\underline{s}$ be material coordinates in a point of a body Ω. For a solid body, $\underline{s}$ has three components: $[s_1, s_2, s_3]$, for a surface two components $[s_1, s_2]$, and for a curve a single component $[s_1]$. The Euclidean 3-space positions in the body are given by vector valued function of the material coordinates $\mathsf{p}(\underline{s}, t) = [x(\underline{s}, t), y(\underline{s}, t), z(\underline{s}, t)]$. The body in its natural rest state is specified by $\mathsf{p}^0(\underline{s}, t^0) = [x^0(\underline{s}, t^0), y^0(\underline{s}, t^0), z^0(\underline{s}, t^0)]$.

4.3.2. Dynamical Equations

The equations of motion can be derived from Newton's second law:

$$\vec{F} = m\vec{a} \tag{4.1}$$

where $\vec{F}$ is the net force on a mass m, and $\vec{a}$ is the acceleration of the point. In 1-D system, consider the position $x(t)$ of a mass attached to a spring , a dashpot, and a

time-dependent external force. Equation (1) turns into

$$m\frac{d^2x}{dt^2} = F_{(spring)} + F_{(dashpot)} + F_{(external)} \tag{4.2}$$

If the spring exerts a linear restoring force and the dashpot exerts a force proportional to the velocity, then the equation of motion is

$$m\frac{d^2x}{dt^2} = -kx - \gamma\frac{dx}{dt} + F_{(external)}(t) \tag{4.3}$$

or

$$m\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + kx = F_{(external)}(t) \tag{4.4}$$

The potential energy associated with the springs:

$$U_{(spring)} = \frac{1}{2}kx^2 \tag{4.5}$$

such that when x is near 0 the spring is considered to be in low energy state

and when x is far away from 0, it is in a high energy state. $F_{(spring)}$ will try to make the system be in a low energy state. $F_{(spring)}(x)$ is related to $U_{(spring)}(x)$ through

$$F_{(spring)}(x) = -\frac{U_{(spring)}(x)}{dx} \tag{4.6}$$

so we can get a more general equation of 1-D motion:

$$m\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + \frac{U_{(spring)}(x)}{dx} = F_{(external)}(t) \tag{4.7}$$

Now we have a more generalized equation of motion of each point that obeys Newton's laws. However, the equation for the potential energy is more complicated. The equations governing a deformable model's motion can be written in Lagrange's form as follows:

$$\frac{\partial}{\partial t}(\tau(\underline{s})\frac{\partial \mathsf{p}(\underline{s},t)}{\partial t}) + \gamma(\underline{s})\frac{\partial \mathsf{p}(\underline{s},t)}{\partial t} + \frac{\delta\varepsilon(\mathsf{p}(\underline{s},t))}{\delta \mathsf{p}(\underline{s},t)} = \vec{F}(\mathsf{p}(\underline{s},t),t) \tag{4.8}$$

where $\mathsf{p}(\underline{s},t)$ is the position of the particle $\underline{s}$ at time t, $\tau(\underline{s})$ is the mass density of the body at $\underline{s}$, $\gamma(\underline{s})$ is the damping density, and $\vec{F}(\mathsf{p},t)$ represents the net externally applied forces. $\varepsilon(\mathsf{p})$ is a functional which measures the net instantaneous potential energy of elastic deformation of the body. The force is the derivative of this functional,

$\delta\varepsilon(\mathsf{p})/\delta\mathsf{p}$. However, normal calculus can only take derivatives of functions. So *the Calculus of Variations* must be used.

The first term is the inertial force due to the model's distributed mass. The second term is the damping force due to dissipation. The third is the elastic force due to the deformation of the model away from its natural shape.

4.4. Energies of Deformation

The energies of deformation quantifie the model's actual deformation away from its rigid shape and in this section we develop potential energies of deformation $\varepsilon(\mathsf{p})$ associated with the elastically deformable models. These energies are functionals that define the internal elastic forces of the models.

The shape of a body is determined by Euclidean distances between nearby points. As the body deforms, these distances change. Let $\underline{s}$ and $\underline{s} + d\underline{s}$ be the material coordinates of two nearby points on the body. The distance between these points on the deformed body in Euclidean 3-D space is given by

$$dl^2 = \sum_{i,j} G_{ij} ds_i ds_j \tag{4.9}$$

where the symmetric matrix has the (i, j)'th component given by:

$$G_{ij}(\mathsf{p}(\underline{s})) = \frac{\partial \mathsf{p}}{\partial s_i} \frac{\partial \mathsf{p}}{\partial s_j} \tag{4.10}$$

and is known as the metric tensor or first fundamental form. The curvature tensor denoted by C is

$$C_{ij}(\mathsf{p}(\underline{s})) = \vec{n}\frac{\partial^2 \mathsf{p}}{\partial s_i \partial s_j} \tag{4.11}$$

where $\vec{n}$ is the normal vector to the surface.

Potential energies of deformation for elastic curves, surfaces and solids can be defined by using the above differential quantities. In rigid body motion , the potential energy is zero, but in non-rigid motion the energy grows larger with respect to the quantities of deformations.

For elastic bodies, the energy is a norm of difference between the fundamental forms of the deformed body and fundamental forms of the natural, undeformed body. This norm measures the amount of deformation away from the natural state.

The natural shape of the deformable bodies will be denoted by supercript 0. For this shape we get the metric tensor as follows:

$$G^0_{ij}(\mathsf{p}^0(\underline{s})) = \frac{\partial \mathsf{p}^0}{\partial s_i}\frac{\partial \mathsf{p}^0}{\partial s_j} \tag{4.12}$$

Thus an analogous strain energy for deformable surface in space is

$$\varepsilon_{surface}(\mathsf{p}) = \int_{\Omega} \| G - G^0 \|^2 + \| C - C^0 \|^2 \, ds_1 ds_2 \tag{4.13}$$

and the strain energy for a deformable solid is

$$\varepsilon_{solid}(\mathsf{p}) = \int_{\Omega} \| G - G^0 \|^2 \, ds_1 ds_2 ds_3 \tag{4.14}$$

The deformation energy is zero for rigid motions and they include the fewest partial derivatives necessary to restore the natural shapes of non-rigid curves, surfaces, and solids respectively. However, high-order derivatives can be included to further constrain the smoothness of admissible deformations of these bodies. Now we can minimize the integration as follows:

$$\mathsf{e}(\mathsf{p}) = \frac{\partial \varepsilon(\mathsf{p})}{\partial(\mathsf{p})} \tag{4.15}$$

$$\mathsf{e}(\mathsf{p}) = K(\mathsf{p})\mathsf{p} \tag{4.16}$$

where K is the spring matrix. The equation ?? leads us to a calculus of variation (In appendix A. calculus of variation is fully explained).

Consequently minimzation of the energy mentioned in the equation 4.13 gives us the spring forces which deform the shape of an object.

4.5. Solving Linear Equations

To animate the wire frame model, the equation 4.8 must be solved and this brings us to sparse linear systems. Direct methods such as LU decomposition, and relaxation methods, such as Gauss-Seidel method can be used to solve these sparse linear systems. LU decomposition may be expensive for large systems, since the full matrix version takes $O(N^3)$. Thus, one possible way of solving such systems is through iterative methods. A simple one is the Gauss-Seidel method which is an iterative scheme. Let us first consider the second order differential equation:

$$M\frac{\partial^2 \mathsf{p}}{\partial t^2} + D\frac{\partial \mathsf{p}}{\partial t} + K(\mathsf{p})\mathsf{p} = \mathsf{F}_{(external)}(t) \tag{4.17}$$

where M, D and K are matrix forms of the mass, damping and spring respectively, and p is the vector that contains all the points of the object. Evaluating p for first and second order in discrete form leads us to:

$$\frac{\partial \mathsf{p}}{\partial t} = \frac{(\mathsf{p}^{t+\Delta t} - \mathsf{p}^{t-\Delta t})}{2\Delta t} \tag{4.18}$$

$$\frac{\partial^2 \mathsf{p}}{\partial t^2} = \frac{(\mathsf{p}^{t+\Delta t} - 2\mathsf{p}^{t} + \mathsf{p}^{t-\Delta t})}{\Delta t^2} \tag{4.19}$$

Now we can estimate the new position $\mathsf{p}^{t+\Delta t}$ by obtaining a new matrix A such

that: $A\mathsf{p}^{t+\Delta t} = \mathsf{b}$ The matrix A is as follows:

$$A = K + \frac{M}{\Delta t^2} + \frac{D}{2\Delta t} \tag{4.20}$$

and the vector b is obtained as follows:

$$\mathsf{b} = \mathsf{F}_{(external)}(t) + (\frac{M}{\Delta t^2} + \frac{D}{2\Delta t})\mathsf{p} + (\frac{M}{\Delta t} + \frac{D}{2\Delta t})\mathsf{q} \tag{4.21}$$

where $\mathsf{q} = (\mathsf{p}^t - \mathsf{p}^{t-\Delta t})/\Delta t$

At this stage we can consider the iterative scheme of Gauss-Seidel method in order to solve $A\mathsf{p}^{t+\Delta t} = \mathsf{b}$ where the ith appears as:

$$\sum_{j=1}^{N} a_{ij}x_j = b_i \tag{4.22}$$

Gauss-Seidel updates $\underline{x}$ one element at a time. Consider x_k:

$$a_{ik}x_k + \sum_{j=1}^{k-1} a_{ij}x_j + \sum_{j=k+1}^{N} a_{ij}x_j = b_i \tag{4.23}$$

This is a system of N equation, which overdetermines x_k. Thus , let us consider one equation:

$$a_{kk}x_k + \sum_{j=1}^{k-1} a_{kj}x_j + \sum_{j=k+1}^{N} a_{kj}x_j = b_k \;\; i = 1..N \tag{4.24}$$

The equation can be swept through k ,"solving" one row at a time, and loop for $k = 1$ to N

$$x_k^{n+1} = -\frac{1}{a_{kk}}(\sum_{j=1}^{k-1} a_{kj}x_j^{n+1} + \sum_{j=k+1}^{N} a_{kj}x_j^n - b_k) \tag{4.25}$$

Gauss-Seidel does not use very much storage. As the loop proceeds, the new x_k are stored and are used in further computation of the solution vector. If the matrix $\underline{a}$ is sparse, the iteration proceed quickly. However, the convergence properties of Gauss-Seidel are somewhat poor. It will converge if $\underline{A}$ is positive definite.

Consequently, using such a scheme gives us a confortable usage of the computers memory.

4.6. Approaches for the implementations

The elastic properties of materials constrain the motion and dynamics of objects in the real world, hence, modeling and simulating the physical characteristics of these objects is essential to obtain realistic computer modeling for graphics vision and

animation. This type of modeling is referred to as physics-based modeling.

The major problem for physics-based modeling has been experienced in the establishment of a unified approach for a geometric representation that should provide efficient calculation of both geometric relations and non-rigid physical response to imposed forces. This problem has been addressed by researchers in both computer graphics and machine vision.

The synthetic tissue is a deformable surface , which is an assembly of point masses connected by springs, that is, discrete deformable model. Let node i, where $i = 1, ..., N$, represent a point mass m_i, whose three-space position is $\mathsf{p}_i(t) = [x_i(t), y_i(t), z_i(t)]$. The velocity of the node is $\vec{v}_i = d\mathsf{p}_i/dt$, and its acceleration $\vec{a}_i = d^2\mathsf{p}_i/dt^2$.

Let spring k have natural length l_k and stiffness c_k. Suppose the spring connects the node i to node j, where $\vec{r}_k = \mathsf{p}_j - \mathsf{p}_i$ is the vector of separation of the nodes. The actual length of the spring is $\| \vec{r}_k \|$. The deformation of the spring is $e_k = \| \vec{r}_k \| - l_k$. The force that the spring exerts on node i is

$$\vec{f}_k = \frac{c_k e_k}{\| \vec{r}_k \|} \vec{r}_k \tag{4.26}$$

The total force on node i due to springs that connect it to other nodes $j \in \mathcal{N}_i$ in the deformable surface is:

$$\vec{F}_{i(spring)}(t) = \sum_{j \in \mathcal{N}_i} \vec{f}_k \tag{4.27}$$

The discrete Lagrange equation of motion for dynamic node/spring system is

the system of second order ordinary differential equations as it explained in section ??.

$$m_i \frac{d^2 \mathrm{p}_i}{dt^2} + \gamma_i \frac{d\mathrm{p}_i}{dt} + \vec{F}_{i(spring)} = \vec{F}_{i(external)}; i = 1, ...N \tag{4.28}$$

where γ_i is the coefficient of velocity-proportional damping, dissipating kinetic energy in the surface, $\vec{F}_{i(spring)}$ is the net spring force, $\vec{q}_i$ is the net volume restoration force, and $\vec{F}_{i(external)}$ is the net external force acting on node i. It is possible for a facial point to be displaced from its specific attachment nodes by applying driving force $\vec{F}_{external(i)}$ to it. But in the project the $\vec{F}_{i(external)}$ forces are not used. Instead, The deformations are achieved by the pulling and the pushing of the point itself as shown at figure 4.1.

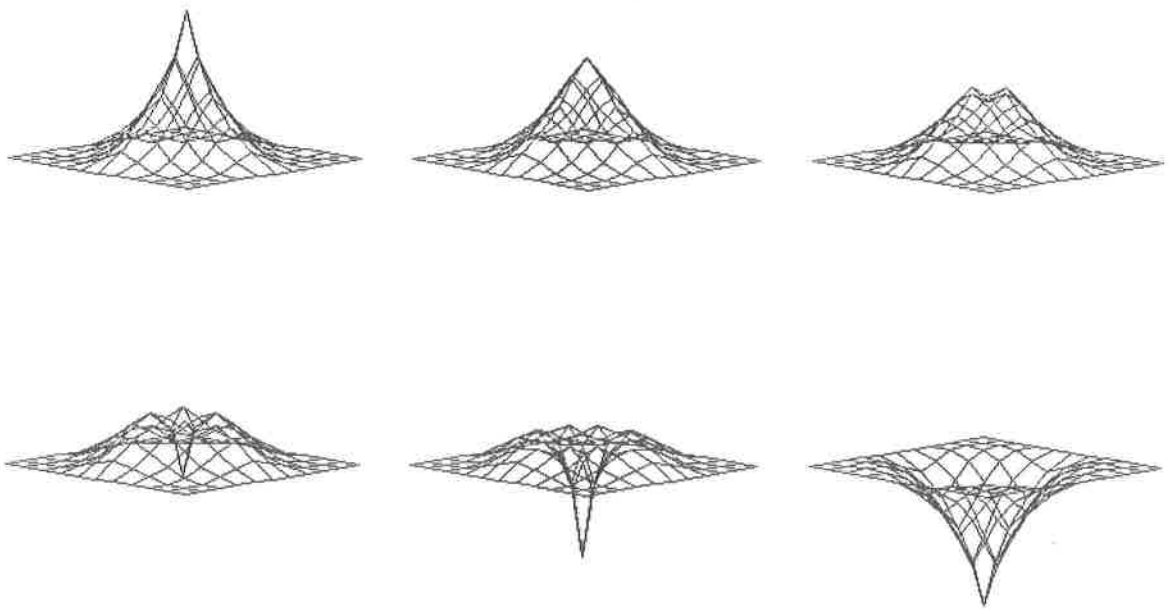

FIGURE 4.1. Deformation of a plane

To simulate the dynamics of a deformable surface, initial positions p_i^0 and the velocities $\vec{v}_i^0 = 0$ are provided for each node i, and the equations of motion are numerically integrated forward through time. Each time step requires the evolution of forces, accelerations, velocities, and positions for all of the nodes. The explicit Euler method is a simple and quick time-integration method, but it has a limited range of stability. Unfortunately, computational complexity will be unavoidable when a higher level of stability becomes a requirement. From the method explicit Euler integration :

$$\vec{f}_i^{nt} = \vec{F}_{i(external)}^t - \gamma_i \vec{v}_i^t - \vec{F}_{i(spring)} \tag{4.29}$$

$$\vec{a}_i^t = \frac{\vec{f}_i^{nt}}{m_i} \tag{4.30}$$

$$\vec{v}_i^{t+\Delta t} = \vec{v}_i^t + \Delta t \vec{a}_i^t \tag{4.31}$$

$$\mathsf{p}_i^{t+\Delta t} = \mathsf{p}_i^t + \Delta t \vec{v}_i^{t+\Delta t} \tag{4.32}$$

A convenient way to compute the net nodal forces is to think of $\vec{f}_i^{nt}$ variables as the nodal force accumulators. Each spring makes the appropriate force contributions into the variables of the two nodes to which it is connected to.

The discrete Langrange equation of motion, coupled with explicit Euler time-integration procedure, provides the foundation for simulating point mass and spring topologies. However, it is the ability to compute solutions rapidly on even modest hardware that is most attractive: A surface with several thousand springs and nodes can be computed and displayed on dedicated graphical workstations in real-time, pro-

viding not only valuable feedback to the user but also interact directly with the model [9]. Furthermore the node/spring element lends itself to parallel computation, enabling larger, more complex surface to be computed.

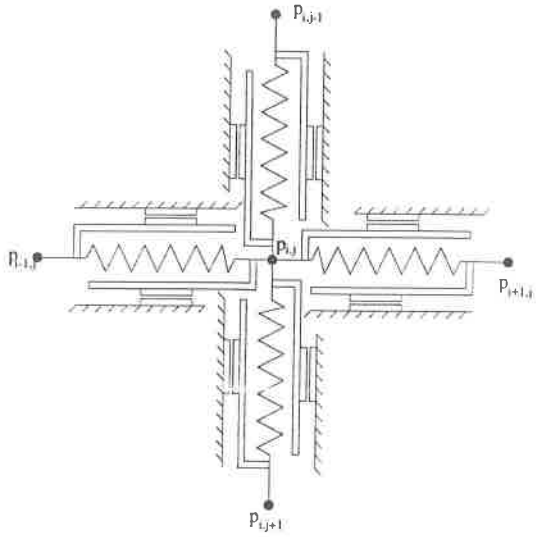

FIGURE 4.2. The structure of damping and spring system

The constants m_i and γ_i are chosen such that the facial surface exhibits a slightly overdamped behaviour.

4.6.1. Two Layered Approach for the Animation of a Face Model

As it is explained before, the approach to simulate motion based on elastic models using discretized continuous equations that are extended for modeling inelastic deformation. In this respect the two layered model represents inelastic material as an elastic component at rest with respect to a reference component used for computing motion. Inelasticity is modeled by allowing the reference component to progressively absorb some of the deformation of elastic layer. In this model, the topology changes

are limited.

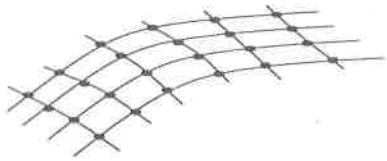

FIGURE 4.3. One layered system

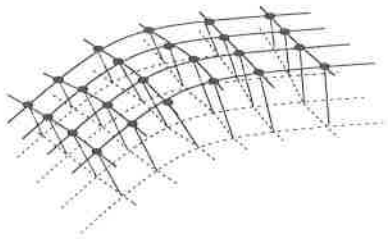

FIGURE 4.4. Two layered system

First, a one layered structure is applied to wire-frame model. But it is observed that when we pull a point and then let it, the relaxation of displaced points takes time and we obtain a shape different from the initial shape. By increasing the spring factor, theoretically we can obtain the initial shape but due to the discrete calculation, the wire system shows a chaotic behavior.

The use of the two layered approach provides us with a very effective animation. As it is seen in figure 4.4, the surface that is parallel to the original surface can be called a virtual layer which has virtual points that are connected to the original points by a spring. Consequently, we obtain a model that absorbs undesirable behaviours of the wire frame . The animation is done by only displacing virtual points that consequently force the original points to itself. In the implementation, this approach is used on mouth and the result is satisfactory.

Another way of getting a realistic animation is to define c_k as follows:

$$c_k = \begin{cases} \alpha_k & \text{when } e_k \leq e_k^c \\ \beta_k & \text{when } e_k > e_k^c \end{cases} \tag{4.33}$$

where the small-strain stiffness α_k is smaller then the large-strain stiffness β_k. Like real tissue, this biphasic is extensible at low strains but exerts rapidly increasing restoring stresses after reaching to a treshold e^c.

FIGURE 4.5. An instance from Waters model animation

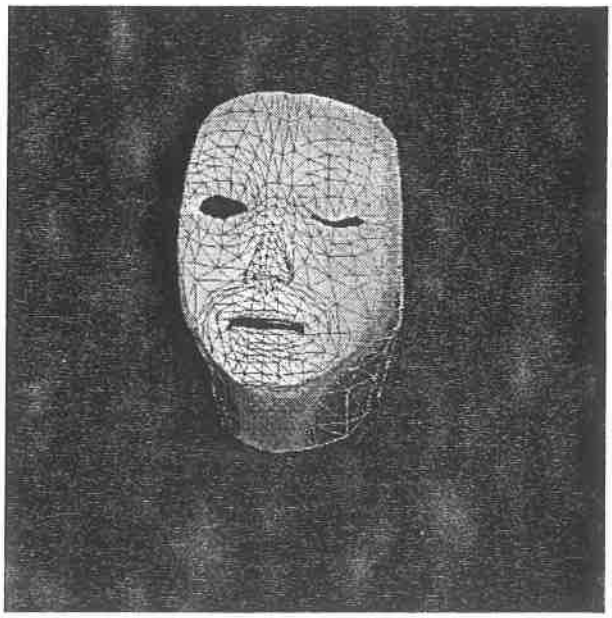

FIGURE 4.6. Another instance from Waters model animation

5. CONCLUSION

This thesis has presented a system which uses snakes for facial feature tracking and physics-based face models for animation.

A contour model, Snake , is used to track the position of the head and the nonrigid motions of eyes and lips. Snake generally requires a manual initialization .However , researches are being carried out for the automatic initialization of Snake using genetic algorithm [6] and Hough Transform [12]. Snakes are efficient in tracking non-amorph objects well defined shape such as cell. But for facial expession estimation in 2-D , Snakes can cause some problems when the head turns too much . The image sequences should have a adequete resolution in order to get an accurate extraction of feature points by snake.

For 3-D motion estimation, a neural network is used. Generally, the 3-D motion estimation problem is solved by analytical approaches which require a large number of correspondence. For solving the correpondence problem, snakes are used , but they don't provide the required number of correspondence points for analytical solution. Neural networks are used as a solution to these problems. A new localisation of the Candid Model has been developed to prevent the extreme time consumption of the training procedures. Neural network offers an off-line system. Once, it has been trained, the estimation is done automatically by only giving the displacement vectors of moved points and the result is the rigid motion parameters.

Theoretically, the technique is tolerent to possible discrepencies between the 3-D geometry of the subject's face and the face model. But pratically, it is necessary to add some extra information in order to obtain a significant result.

For a teleconferencing, this system will provide a very high compression ratio due to the fact that only a required subset of information will be transmitted.

A Calculus of Variations

Consider the function y(x), where x and y are scalars, and the functional

$$J[y] = \int_a^b F(x, y, y\prime)dx \tag{A1}$$

The esiest way to take a derivative is to consider the integral to be a sum of n terms and let $n \to \infty$. Divided the interval $[a, b]$ into n equal to segments, the end points of which are labeled by x_i. Let $h = x_{i+1} - xi$. And we want approximate the function by a polygonal line with the vertices

$$(x_0, y_0), (x_1, y_1), ..., (x_n, y_n), (x_{n+1}, y_{n+1}). \tag{A2}$$

The integral can be approximated by using finite differences as follows:

$$J(y_0, y_1, ...y_n) = h \sum_i F(x_i, y_i, \frac{y_{i+1} - y_i}{h}). \tag{A3}$$

If we compute the partial derivatives of J, we get:

$$\frac{\partial J(y_0, y_1, ...y_n)}{\partial y_i} = hF_y(x_i, y_i, \frac{y_{i+1} - y_i}{h}) \tag{A4}$$

$$+F_{y\prime}(x_{i-1}, y_{i-1}, \frac{y_i - y_{i-1}}{h}) - F_{y\prime}(x_i, y_i, \frac{y_{i+1} - y_i}{h}) \tag{A5}$$

where F_y is the function $\partial F/\partial y$ and $F_{y\prime}$ is the function $\partial F/\partial y\prime$. From functional analysis

$$\frac{\delta J[y]}{\delta y} |_{y_i} = \lim_{h\to\infty} \frac{1}{h} \frac{\partial J(y_0, y_1, ...y_n)}{\partial y_i} \tag{A6}$$

The extra factor of h allows the limit to exist, since the partial derivatives scale as h.

$$\frac{\delta J[y]}{\delta y}|_{y_i} = \lim_{h\to\infty} [F_y(x_i, y_i, \frac{y_{i+1} - y_i}{h}) \tag{A7}$$

$$-\frac{1}{h}(F_{y\prime}(x_{i-1}, y_{i-1}, \frac{y_i - y_{i-1}}{h}) - F_{y\prime}(x_i, y_i, \frac{y_{i+1} - y_i}{h}))] \tag{A8}$$

Passing to continuum limit yields the variational derivatives

$$\frac{\delta J[y]}{\delta y} = \frac{\partial F}{\partial y} - \frac{d}{dx}(\frac{\partial F}{\partial y\prime}) \tag{A9}$$

More generally , if a functional is the form

$$J[y] = \int_a bF(x, y, y\prime, ...y^{(N)})dx \tag{A10}$$

then the variational problem became

$$\frac{\delta J[y]}{\delta y} = \sum_{n=0}^{N}(-1)^n \frac{d^n}{dx^n}(\frac{\partial F}{\partial y^{(n)}}) \tag{A11}$$

where $y^{(n)}$ means the nth derivative of y

An even more general formula is when the functional is of the form

$$J[y] = \int_\Omega F(\underline{x}, y, y\prime)d\underline{x} \tag{A12}$$

where the integral is now a multiple integral over $\underline{x} \in \Omega$. In the case, the variational derivative is

$$\frac{\delta J[y]}{\delta y} = F_y - \sum_i \frac{\partial}{\partial x_i} F_{y_{x_i}} \tag{A13}$$

B The Back-Propagation Training Algorithm

The back-propagation training algorithm is an iterative gradient algorithm desired to minimize the mean square error between the actual output of a multilayer feed-forward perception and desired output. It requires onto diffretiable non-linearities. The following assumes a sigmoid logistic non-linearities is used where the function $f(\alpha)$ is

$$f(\alpha) = \frac{1}{1 + e^{-(\alpha - \theta)}} \tag{B1}$$

Step 1.Initilize weights and offsets Set all weights and nodes offsets to small random values

Steps 2.Present input and desired outputs Present continous valued input vector $x_0, x_1, ...x_{N-1}$, and specify the desired output $d_0, d_1, ...d_{M-1}$.

Step 3.calculate actual outputs Use the sigmoid nonlinearity from above and formulas to calculate outputs $y_0, y_1, ...y_{M-1}$

Step 4.Adapt weights Use a recursive algorithm starting at the output nodes working back to the hidden layer. Adjust weights by

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_j x_i\prime \tag{B2}$$

In this equation $w_{ij}(t)$ is the weights from hidden node i or an input to node j

at time $t, x\prime$ is either the output of node i or is an input, is a gain term, and is an error term for node j. If node j is an output node, then

$$\delta_j = y_j(1 - y_j)(d_j - y_j) \tag{B3}$$

where d_j is the desired output of node j and y_j is axtual output. If node j is an internal node, then

$$\delta_j = x_j\prime(1 - x_j\prime)\sum_k \delta_k w_{ik} \tag{B4}$$

where k is overall nodes in the layers above nodes j. Internal node thresholds are adopted in a similar manner by assuming thay are connection weights on link from auxilary constant-valued inputs. Convergence is sometimes faster if momentum term is added and weight changes are smoothed by

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j x_i\prime + \alpha(w_{ij}(t) - w_{ij}(t-1)) \tag{B5}$$

C Potential Field

In order to create a potential field from an intensity image the recursive Rosendeld and Pfaltz distance operator can be used.

The distance transformation operator can be implemented as either recursive or a nonrecursive operator. It requires a binary image whose border pixels are labeled 0 and whose interior pixels are labeled i. The purpose of distance transformation operator is to produce a numeric image whose pixels are labeled with the distance between each of them and their closest border pixel. The distance between two pixels can be defined by the length of the shortest 4-connected path (city block distance) or 8-connected path (max or chessboard distance) between them.

As a nonrecursive operator, the distance transformation can be achieved by successive application of the pair relationship operator. In the first application the pair relationship labels with a 1, all pixels whose label is i and that are next to a pixel whose label is 0. All other pixels keep their labels. In the nth application, tha pair relationship operator labels by an n, all pixels whose label is i and that are next to a pixel whose label is $n-1$. When no pixel has the label i, an application of the pair relationship operator changes no pixel value, and the resulting image is the distance transformation image. This implementation is related to the one given by Rosenfeld and Pfaltz.

Another way of implementing this operator nonrecursively is by the iterative application of the primitive function defined by

$$h(a_0,....a_N) = \begin{cases} i & \text{if } a_n = i,\ n = 0,...., N \\ \min & \{b \mid \text{for some } n \leq N,\ a_n \neq i, b = a_n + 1\} \\ & \text{if } a_0 = i \text{ and there exists } n \text{ such that } a_n \neq i \\ a_0 & \text{if } a_0 \neq i \end{cases} \tag{C1}$$

where i is the special interior pixel label.

Another way of implementing the distance transform involves the application of two recursive operators being applied in a left-bottom scan and the second operator in a right-left, bottom scan. Both operators are the primitive function h defined by

$$h(a_0,....a_N) = \begin{cases} 0 & \text{if } a_N = 0 \\ \min\{a_1,...., a_N\} + d & \text{otherwise} \end{cases} \tag{C2}$$

In 8-connected mode, the output symbol y of the first operator is defined by

$$y = h(x_2, x_6, x_7, x_3, x_0; 1) \tag{C3}$$

In 4-connected mode, the output symbol y is defined by

X_7	X_2	X_6
X_3	X_0	X_1
X_8	X_4	X_5

FIGURE C1. 8-connected window

$$y = h(x_2, x_3, x_0; 1) \tag{C4}$$

For the second operator, the primitive function is simply the minimum function. In the 8-connected mode, the output symbol y of the second operator is defined by

$$y = \min\{x_0, g(x_1, x_4, x_5, x_8; 1)\} \tag{C5}$$

In 4-connected mode, the output symbol y is defined by

$$y = \min\{x_0, g(x_1, x_4; 1)\} \tag{C6}$$

where $g(a_1, ..., x_N; d) = \min\{a_1, ..., a_N)\} + d$.

It is possible to compute a distance transformation that gives distances closely approximating Euclidean distance. For the first left-right, top-down pass, the output

y is given by

$$y = \min\{h(x_2, x_3, x_0; d_1), h(x_6, x_7, x_0; d_2)\} \tag{C7}$$

For the second right-left, bottom-up pass, the output y is given by

$$y = \min\{g(x_1, x_4; d_1), h(x_5, x_8; d_2)\} \tag{C8}$$

Montanari puts $d_1 = 1$ and $d_2 = \sqrt{2}$. This gives the correct Euclidiean distance for all shortest paths that are vertical, horizontal, or diagonal. Barrow and Nitzian and Agin use a scaled distance and put $d_1 = 2$ and $d_3 = \sqrt{2}$

D Animation of falling ball

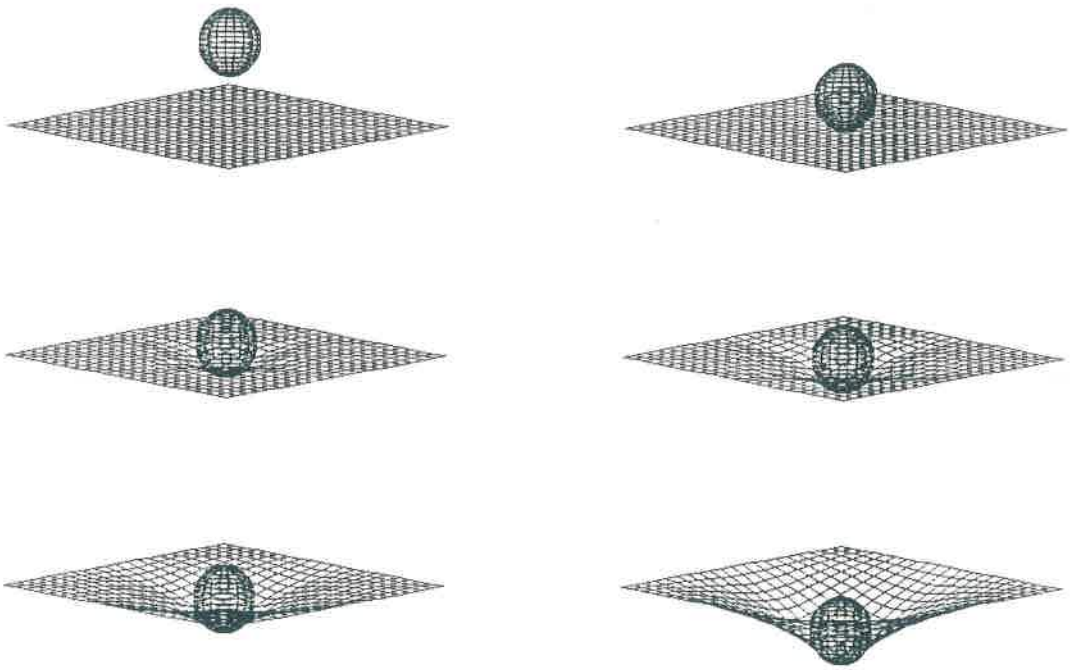

FIGURE D1. Falling ball

References

[1] Peter W. Hallinan Alan L. Yuill and David S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8(2):99–111, 1992.

[2] Terry E. Weymouth Amir A. Amani and Ramesh C. Jain. Using dynamic programing for solving variational problems in vision. *IEEE Trans. Pattern Anal. Machine Intell.*, 12(9):855–867, 1990.

[3] A Azarbayejani and A. Pentland. Recursive estimation of motion, structure, and focal lenght. Technical report, MIT Media Lab, 1994.

[4] Ted Broida and Rama Challapa. Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1), 1986.

[5] Laurent D. Cohen and Isaac Cohen. Finite-element method for active contours models and balloons for 2-d and 3-d images. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(11):1131–1147, 1993.

[6] Tony Heap and Ferdinando Samaria. Real-time hand tracking and gesture recognition using smart snakes. Technical report, Olivetti Research limited, 1995.

[7] Stan Scarloff Irfan A. Essa and Alex Pentland. Physically-based modelling for graphics and vision. Technical report, MIT Media Lab Perceptual Group Thec. Report No.14, 1992.

[8] Laurent D. Cohen Isaac Cohen and Nicholas Ayache. Introducing deformable surface to segment 3-d images and infer differential structures. Technical report, INRIA, 1991.

[9] K. Fleischer J. Platt, D. Terzopoulos and A. Barr. Elastically deformable models. Technical report, California Institute of Technology, Pasadena, 1993.

[10] J.K. Aggarwal Johm Roch. Determining the movement of an objects from a sequence of images. *IEEE Trans. Pattern Anal. Machine Intell.*, 2(6):554–562, 1980.

[11] Richard G. Palmer John Hertz, Anders Krogh. *Introduction to The Theory of Neural Computation.* Addison-Wesley Publishing Company, 1991.

[12] K. F. Lai and R.T. Chin. On regularization, formulation and initialization of the active contour models (snakes). *Asian Conference on Computer Vision*, pages 542–545, 1993.

[13] Frederic Leymarie and Martin D. Levine. Tracking deformable objects in the plane using an active contour model. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(6):617–634, 1993.

[14] Piotr Mikusinski Lokenath Debnath. *Introduction to Hilbert Space with Applications.* Academic Press INC. Hacourt Brace Jovanovich, Publishers, 1990.

[15] Andrew Watkin Micheal Kass and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 3(3):321–331, 1988.

[16] F. I. Parke. Parameterized models for facial animation. *IEEE Comput. Graphics Applications.*, 2(9):61–68, 1982.

[17] Linda G. Shapiro Robert M. Haralick. *Computer and Robot Vision.* Addison-Wesley Publishing Company, 1992.

[18] Thomas S. Huang Roger Y. Tsai and Wei-Le Zhu. Estimation three-dimensional motion parameters of rigid planar patch, singular value decomposition. *IEEE Trans. on Acoustic, Speech, and Signal Processing*, 30(4):525–534, 1982.

[19] T. Mukarami T. Fukuhara. 3-d motion estimation of human head for model-based image coding. *IEE Proceeding*, 14(1):30–40, 1993.

[20] Demetri Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4), 1986.

[21] Demetri Terzopoulos and Keith Waters. Analysis and sythesis of facial image sequences using physical and anotomical models. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(6):569–579, 1993.

[22] Roger Y. Tsai and Thomas S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):13–27, 1984.