SNAKE CONTOURS IN THREE-DIMENSIONS FROM COLOUR STEREO IMAGE PAIRS

by

Roozbeh Ghaffari
B.Sc., Sharif University of Technology, 2002

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

© Roozbeh Ghaffari 2005
SIMON FRASER UNIVERSITY
Spring 2005

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without the permission of the author.

APPROVAL

Name:	Roozbeh Ghaffari
Degree:	Master of Science
Title of thesis:	Snake contours in three-dimensions from colour stereo image pairs

Examining Committee: Dr. Greg Mori
Chair

Dr. Brian Funt, Co-Supervisor

Dr. Ghassan Hamarneh, Co-Supervisor

Dr. Tim Lee, External Examiner

Date Approved: April 11, 2005

SIMON FRASER UNIVERSITY

PARTIAL COPYRIGHT LICENCE

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

W. A. C. Bennett Library
Simon Fraser University
Burnaby, BC, Canada

Abstract

Snakes (active contour models) are extended to segment regions of interest on the surface of 3D objects. Stereo images taken with calibrated cameras are used as input. For this method the depth map for the whole image does not need to be computed. Instead, 3D external forces are designed to keep the contour on the surface of the object while moving it toward the desired boundaries. Color information is used to improve the ability of snakes in detecting the boundaries, in contrast to the majority of previous methods which are based on intensity information alone.

The proposed method produces 3D contours on the surface of the object with coordinates in physical units, e.g. millimeters. These contours can be used to view the structure and dimensions of any distinguishable region on the surface of an object. Examples include oral lesions and skin diseases.

The whole process requires minimal human interaction; however, user input can be used to improve segmentation.

To Shirin, Mohammand, Azar and Sara
for their love, support, tolerance and patience

"Do not hover always on the surface of things,
nor take up suddenly, with mere appearances;
but penetrate into the depth of matters,
as far as your time and circumstances allow,
especially in those things which relate to your profession."

— *Isaac Watts*

Acknowledgments

I am mostly grateful to my supervisors Dr. Brian Funt and Dr. Ghassan Hamarneh for their generous support and invaluable remarks on my work.

I would like to thank my beloved wife, friend and colleague, Sara, for all her love and support; for never letting me feel alone and constantly encouraging me throughout my life.

I thank my parents and my grandparents for their unconditional love and continuous support; for all their sacrifices...

Contents

List of Tables

List of Figures

List of Programs

Chapter 1

Introduction

In the process of recording the development of oral lesions, pictures of the diseased tissues are very important. This archive of images can be used to track changes in the location and dimensions of the lesion. This can be true also for any kind of visible lesion or skin disease. However, there are two big disadvantages to traditional photos as a means of recording objects.

First, the images can be distorted due to physical limitations of the camera. Also there is no reference for measuring the dimensions in the images. Using pre-measured probes in the pictures can be useful for estimating the lengths in the image. However, the problems of distortion and perspective in the images remains to be addressed. Also, the length of the probe is only comparable with objects that are at the same distance from the lens of the camera as the probe. Second, images are 2D projections of 3D objects. Obviously during this transformation there will be loss of information. Curved objects may appear planar. Also the dimensions of different objects or even different parts of the same object may not be comparable after this projection.

One of our goals when recording information about the dimensions and structure of an object would be having exact 3D structure data in real world metrics. In the course of this thesis, we will gather this information from 2D images taken from the subject.

One of the simplest and also frequently used techniques for gathering 3D information from 2D images is *stereo imaging.* By presenting slightly different images of one scene to each eye, each from a different point of view, the brain is able to extract depth information from the images and have a 3D perception of the scene. Chapter 2 provides more information on how the stereo imaging techniques are used in this work to extract the information needed to reconstruct the 3D structure of tissues of interest.

Before extracting any 3D information from images, it is very important to have data about the characteristics of the camera used to take the pictures. This information helps us eliminate the undesired effects created by cameras during image acquisition. *Camera calibration* is the process of determining the geometric and optical characteristics of a camera. These characteristics have a direct effect on the way a scene is projected onto a planar image. Combining the information provided by the stereo vision techniques and camera calibration information enables us to extract precise metric information from 2D images. More details about camera calibration and the effects of stereo imaging on this process are available in Chapter 3. It will be shown that there are more characteristics that need to be determined during the calibration of a stereo system than a single camera.

One of the goals of this work is the automatic segmentation of regions of interest in images. These regions are assumed to be visually different and distinguishable from their neighborhoods. The act of distinguishing objects from background is called *image segmentation* in computer vision literature. One of the popular approaches to this problem are edge-based methods, which rely on boundary detection. Classical *deformable models* [17] are in this category and they are usually implemented with incorporating the physics-based dynamics of flexible materials. The theory of elasticity describes deformable materials such as rubber and flexible metals. This theory is employed [16] to construct differential equations that model the dynamics of non-rigid curves as a function of time. These dynamic curves respond to forces from the images to embrace the regions of interest. Chapter 4 contains detailed information about the techniques used in elastic deformable models. *Snake* [9], the energy-minimizing spline, is the active contour model used in this work. The traditional snake works on intensity images, i.e. single channel images. Use of multi-channel images has been investigated for snakes by Sapiro[13]. This work also makes use of color information to improve the performance of snakes in segmentation. This proved to be much more effective than simply using the intensity channel.

We try to apply the same ideas available for 2D flat images to stereo images taken from objects with distinguishable regions. A curve that is not necessarily planar is fit around that region. Precise measurements of the dimensions of the contour as well as 3D shape of region's border will be provided.

This is accomplished by performing the segmentation on both left and right images taken with a stereo camera and deforming the contour in 3D space under the effect of the forces computed based on planar projections of the region. Minimal user interaction is required to initialize the contour and it is possible to make use of user input to enhance the results. The algorithm is presented in Chapter 5 in detail.

The idea of merging stereo information with active contours has been explored before. Markovic and Gelautz in [10] used depth maps extracted from stereo images to enhance the performance of snake segmentation. They still produce 2D contours though. Cham and Cipolla investigated "stereo coupled active contours" in [3] to enhance motion tracking in stereo by coupling pairs of active contours in different views. They use stereo information to track an object on both left and right views.

In order to evaluate the proposed method, we applied it to both synthetic and real data. Measurements provided with this system are compared to actual measurements done with physical instruments. The 3D structure is compared as well. The results have been satisfactory for most cases. These experiments and results are provided in Chapter 6. Detailed information about the specifications of the cameras used, accuracy of measurements relative to image resolution and object distance, and physical properties of the implemented system are also provided.

Chapter 7 concludes this work and suggests further potential improvements. Because of the inherent modularity that the proposed algorithm has, there are multiple choices for different stages of it. Each section can be replaced to use a different method to suit the context the algorithm is used in the most.

Chapter 2

Stereo Vision

Stereoscopic or binocular vision is the ability of both eyes to look at the same point from a slightly different angle. The eyes' visual fields overlap in the center, and the brain merges these two images to create a sense of depth important for judging distance. Humans and other mammals have stereoscopic vision. Birds, fish, and snakes[1] have monocular vision in which each eye sees a separate image covering a wide area on each side of the head. The same idea has long been used in computer vision to extract depth information from 2D images [11].

The main problem in stereo vision is to find the correspondence between the points on the two images. After finding the corresponding points the depth can be calculated from the disparity of the two. Because of different factors such as noise, occlusion, lighting variations and perspective distortions, the appearances of the corresponding point will differ in left and right images. This makes it hard to establish correspondence for various features of one image. In order to improve the process of matching the features of the left and right images, it is usually necessary to use additional constraints. Four commonly used ones, as described in [15], are

Epipolar constraint : The matching points must lie on the corresponding epipolar lines of the two images. For epipolar rectified images, the matching points lie on the same image scanlines of a stereo pair.

Uniqueness constraint : Matching should be unique between the two images.[2]

[1]Not to be confused with the active contours used in this work to perform segmentation!

[2]There is no one-to-one correspondence between the points on the left image and the ones on the right

Smoothness constraint : Local regions of the disparity map should be relatively smooth except for the regions with occlusion or disparity discontinuity.

Ordering or monotonicity constraint : For points along the epipolar line in one image the corresponding points have to occur in the same order on the corresponding epipolar line in the other image.

Section 5.1.2 has more information about the way these constraints are used in our work. It also provides details on how the corresponding points are found.

After finding the corresponding points depth information can be extracted from the images. The basic idea is described in [19] and illustrated in Figure 2.1. This figure is very similar to the one shown in the cited book.

It is easy to show

$$d = \frac{fb}{\alpha + \beta}. \tag{2.1}$$

Thus distance to a point is inversely proportional to $\alpha + \beta$, the amount of shift of the point's position in one image relative to the point's position in the other. This shift is called *disparity*. Unfortunately this simple formula only works in ideal circumstances. Cameras have to have the same characteristics. The direction of the cameras should be parallel and perpendicular to the line connecting them. In real world cases none of these happen and we need to calibrate the cameras and the stereo system. Section 3.3 describes how this "back-projection" works in our system.[3]

one necessarily. For example in case of occlusion or very steep surfaces more that one point on one image may correspond to a single or no point on the other, but *generally* there is one-to-one correspondence and there is no more than one way of associating the points of the images.

[3]It is worth mentioning that one other approach, which is often used in stereo imaging is to *rectify* the original images so that they appear as if they are taken with two cameras without distortion and parallel to each other. Details of this rectification process is out of the scope of this thesis. It is not used in our work, since reconstruction of the whole surface is not among our goals and performing the rectification on the whole images would have been unnecessary computations.

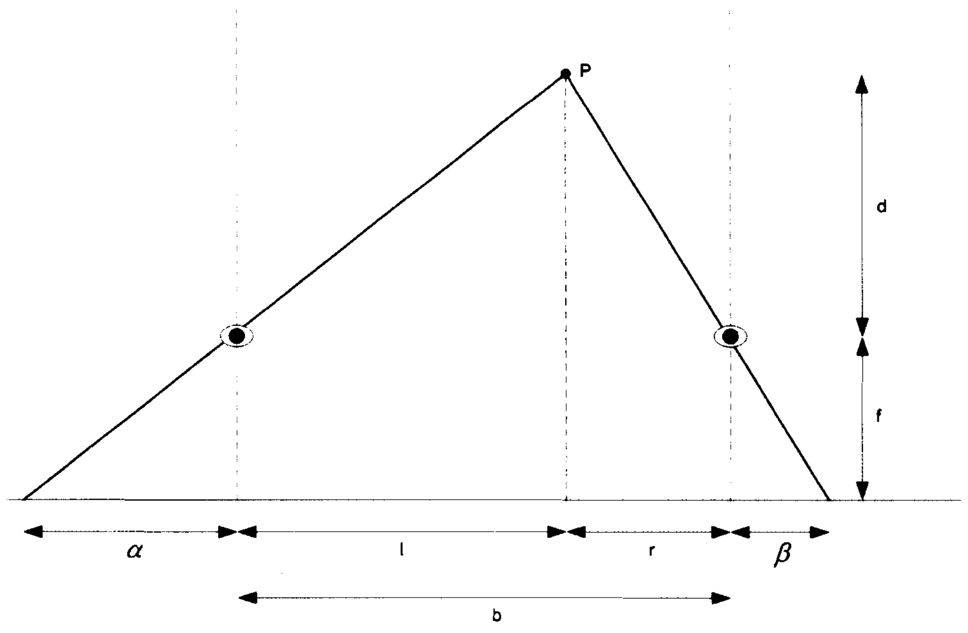


Figure 2.1: Geometry of the two cameras involved in stereo imaging (adopted from [19]). P is an arbitrary point in the space. d represents the distance of P to the cameras and f is the focal length.

Chapter 3

Camera Calibration

The first step in a system for 3D reconstruction of real world objects is to make sure that it has reasonably precise measurements of the objects. This is specially true for medical applications where the size of the lesions and tissues are of great importance. Thus, it is important to know the characteristics of the device used in measurements.

Camera calibration in the context of computer vision is the process of determining geometric and optical characteristics of the camera. Camera calibration is an essential step in 3D reconstruction methods in order to extract measurements from the images. In many cases the performance and accuracy of the machine vision system strongly depends on precise calibration of the camera.

Physical camera parameters are normally categorized into *intrinsic* and *extrinsic* parameters. Intrinsic camera parameters include the focal length f, skew coefficient, aspect ratio, distortion coefficients and the image center, also called the *principal point*. On the other hand extrinsic parameters contain the information needed to transform object coordinates to a camera-centered coordinate frame. In multi-camera systems, such as stereo systems, the extrinsic parameters also include the relationship between the cameras. [7]

We used the "Camera Calibration Toolbox for Matlab" [1], which is capable of performing camera calibration, undistorting images, stereo calibration and many other features not used in our work. The toolbox is also capable of rectifying stereo images but due to specific techniques used in our algorithm, image rectification is not necessary in our work. Chapter 5 has more information about these techniques. The camera model used in our work is the same as the model used in the aforementioned toolbox.

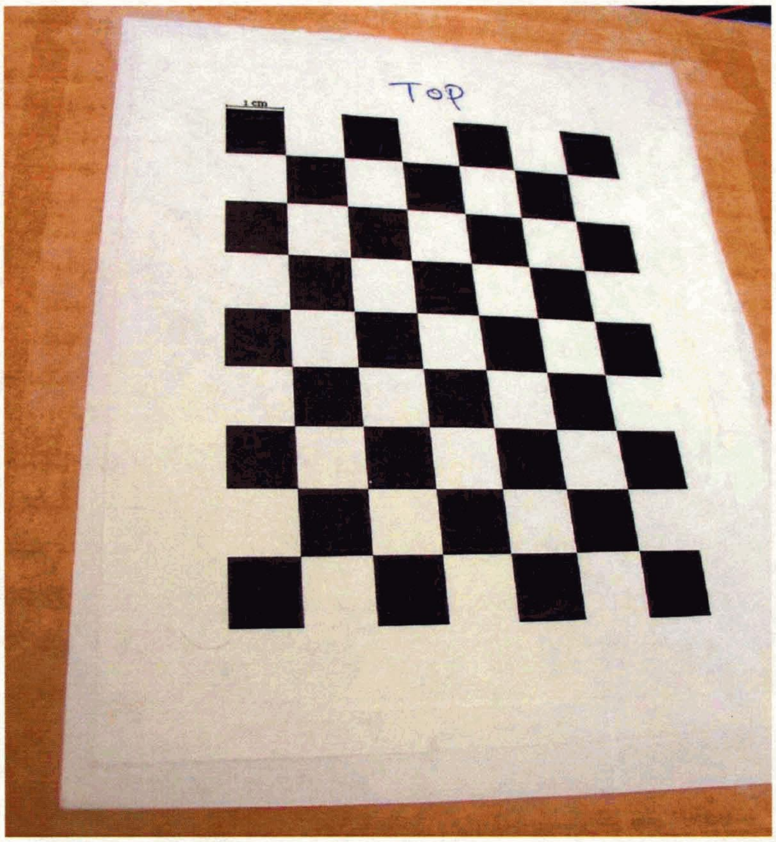


Figure 3.1: A checkerboard used for calibrating the cameras

3.1 Single Camera Calibration

Stereo calibration is calibration of two separate cameras and calculating the translation and rotation vectors that specify the relation between the location and direction of the two cameras. In order to perform the calibration, we use the method proposed in [22]. A number of pictures (about 10) are taken from a planar checkerboard in which the sizes of the squares are known (Figure 3.1). These pictures are provided to the calibration toolbox and after a phase of "corner extraction" intrinsic parameters of the camera are calculated.

During the corner extraction phase, the corners of the checkerboard squares are detected after the user manually selects the 4 corners of the checkerboard with a pointing device such as a mouse. Then the system minimizes the pixel reprojection error in the least squares sense over the intrinsic camera parameters, and the extrinsic parameters (3D locations of the grids in space). A gradient decent algorithm is employed to perform this minimization [1].[1]

3.1.1 Internal Camera Model

The internal camera model used is similar to the one used in [7]. This model has four parameters for the camera:

Focal length or f is the camera's focal length in pixels. The real value of f is stored in a 2×1 vector because the pixel are assumed not to have a square shape necessarily. So the value of f along the x axis might be different than the value of f along the y axis. The ratio of these two values f_y/f_x is often called the "aspect ratio". This value is usually 1 for rectangular CCDs.

Principal point c is the coordinates of the image center. This is the point in the image where the camera is "looking at".

Skew coefficient α defines the cosine of the angle between the x and y pixel axes. For the modern cameras available today we can assume $\alpha = 0$.

Distortions are the radial and tangential image distortion coefficients stored in vectors k_r and k_t.

For example, having a point $P = (P_x, P_y, P_z)$ in the camera reference frame (in 3D space)[2] we can calculate the projection of that point on the image. Let $\mathbf{p}$ be the *pinhole image* projection

$$\mathbf{p} = \begin{pmatrix} P_x/P_z \\ P_y/P_z \end{pmatrix}$$

Assuming $r^2 = \mathbf{p}_x^2 + \mathbf{p}_y^2$, lens distortion is applied

$$\mathbf{q} = (1 + k_{r_1} r^2 + k_{r_2} r^4)\mathbf{p} + \mathbf{dx}$$

where $\mathbf{dx}$ is the tangential distortion vector[3]

$$\mathbf{dx} = \begin{pmatrix} 2k_{t_1}\mathbf{p}_x\mathbf{p}_y + k_{t_2}(r^2 + 2\mathbf{p}_x^2) \\ k_{t_1}(r^2 + 2\mathbf{p}_y^2) + 2k_{t_2}\mathbf{p}_x\mathbf{p}_y \end{pmatrix}$$

[1]Gradient decent is an algorithm for finding the nearest local minimum of a function. It presupposes that the gradient of the function can be computed. It is also called the steepest decent method.

[2]The lens is at $(0, 0, 0)$ and looks along the Z axis. Y is to the top and X to the right.

[3]This distortion model was introduced by D.C. Brown in [2]. The tangential distortion is due to "decentering" of the lens components. More details are available in Brown's original publication.

The final pixel coordinates $\mathbf{c}$ in the images are

$$\begin{pmatrix} \mathbf{c}_x \\ \mathbf{c}_y \\ 1 \end{pmatrix} = \mathbf{KK} \begin{pmatrix} \mathbf{q}_x \\ \mathbf{q}_y \\ 1 \end{pmatrix}$$

where $\mathbf{KK}$ is known as the *camera matrix*, and defined as

$$\mathbf{KK} = \begin{pmatrix} f_x & \alpha f_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}.$$

3.1.2 Calibration by Viewing a Checkerboard

The calibration process starts with taking a few (about 10) pictures of a checkerboard in which the dimensions of the squares are known precisely.[4] Pictures should be taken from different angles. We got the best results by taking pictures from about the same distance the actual pictures are taken. Because the size and the shape of the checkerboard is known, every picture taken adds a set of equations to the system. The intrinsic parameters of the camera are limited. After taking enough pictures it is possible to calculate the those parameters as well as the exact location of the checkerboard in each camera reference frame. In other words we can also calculate the location and direction of the camera in each shot. Extra equations from the extra shots can be used to improve the accuracy of the calculations. Details of solving this system can be found in [22].

For each shot of the checkerboard, the corners' coordinates are calculated in two ways. First by searching the image for them and second by calculating their locations using the camera model and parameters. The more accurate the model and parameters are, the less difference there will be between the two sets of coordinates. Using a local search technique, the mean square error is minimized and camera parameters are calculated.

3.2 Stereo Calibration

Stereo calibration means calibrating the left and right cameras and finding the relation between the location and direction of the two. In other words in addition to intrinsic

[4]Using a graphics program such as Adobe Photoshop it is possible to print a checkerboard with known metrics. So no physical measuring is needed.

parameters of each camera we want to calculate a translation and a rotation vector which transforms the left camera reference frame into the right camera reference frame.

Shooting a single set of pictures from the checkerboard with the stereo system provides enough data to extract the information needed. The images taken with each camera can be used to extract the internal parameters of that camera. As we noted in Section 3.1.2, it is possible to calculate the location and direction of the camera in each shot with respect to the checkerboard. Knowing the fact that the coordinates of the cameras with respect to each other are constant, it is straightforward to extract the rotation vector om and the translation vector T. The om vector can also be represented as a rotation matrix R where

$$R = \text{rodrigues}(om)$$

and rodrigues is a mathematical functions that transforms a 3D rotation vector of size 3×1 to a 3D rotation matrix of size 3×3. The result of a stereo calibration using 10 different shots is shown in Figure 3.2.

3.3 Back-projecting into 3D Space

The transformations described in 3.1.1 are not reversible. It is not possible to say which point in 3D space was projected into a pixel in an image. There are an infinite number of points in the space that can project onto a single point in an image. Usually we see only the closest one to the camera. However, all those points in the space lie on the line that goes through the point on the image and the center of the camera lens.[5] Any point on that line could be the point that we actually see in the picture.

Stereo imaging is about taking two shots at the same time from a single scene. Any particular point in the scene which is present on both left and right images has one projection on each image. Those two projections each specify a line through the space whose all points could be the actual point in the scene. Those two lines intersect in only one point.[6]

For any point on one of the images (P_l) knowing the corresponding point on the other image (P_r), using the technique described above we can calculate the exact location of

[5]This is assuming the camera is a pinhole camera which is not an unrealistic assumption.

[6]If the cameras are calibrated accurately, those two line will intersect. Small errors in calibration results in two lines narrowly passing each other. In order to take care of such cases, instead of intersecting the two lines we choose the point that has the minimum total distance from the lines. If they intersect that point is at the intersection, otherwise it is the best estimate.

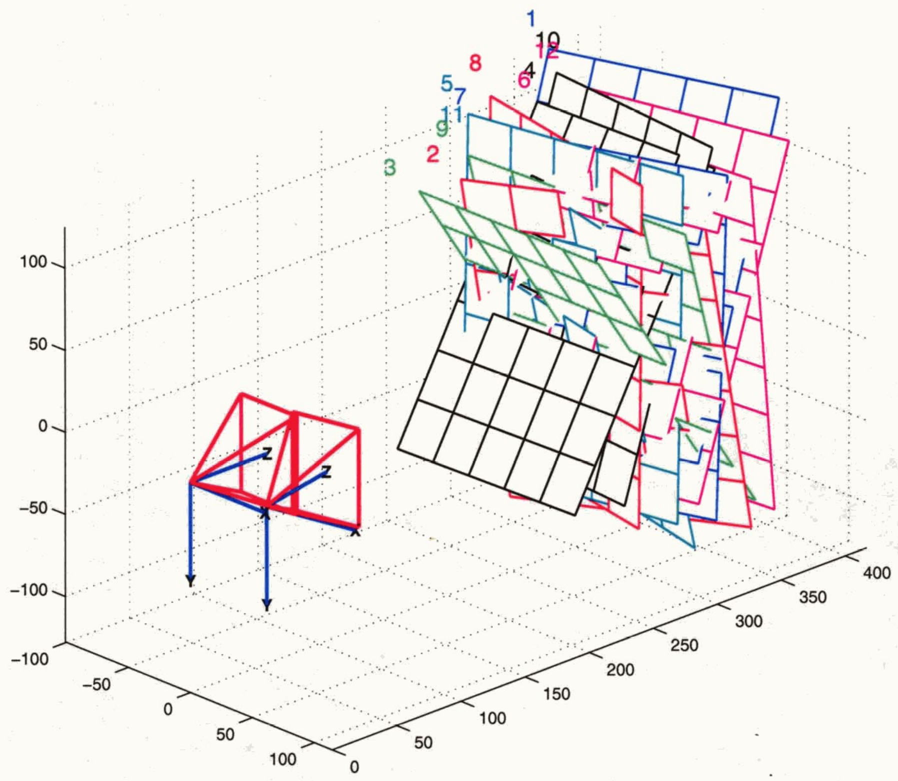


Figure 3.2: Result of a stereo calibration using 10 different shots. Location of the checkerboard in different shots and left and right cameras with respect to each other (all coordinates in left camera reference frame)

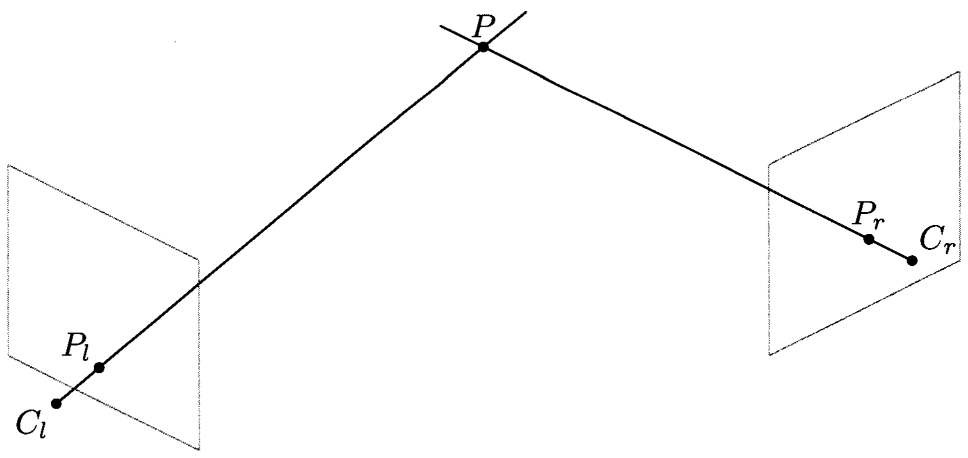


Figure 3.3: Back-projecting into 3D space. P_l and P_r are left and right projections of P. C_l and C_r are the coordinates of the centers of the left and right cameras. The coordinates of C_l, P_l, C_r and P_r are known in the space.

the point in the scene (P) that projected onto those two points in the images. Figure 3.3 is helpful in understanding the geometry of the problem. Since the calibration data are available the coordinates of C_l, P_l, C_r and P_r are known and by intersection the lines passing C_l, P_l and C_r, P_r we can calculate the coordinates of P.

Chapter 4

Deformable Models

With the increasing role of medical imaging in the diagnosis and treatment of diseases, the challenging task of processing medical images using computers and extracting useful information has become more important. Image segmentation is to distinguish objects from background. There are different approaches to this task. Deformable models are among them.

A deformable contour is a planar curve which has an initial position and an objective function associated with it. A special class of deformable contours called *snakes* was introduced by Witkin, Kass and Terzopoulos [9] in which the initial position is specified interactively by the user and the objective function is referred to as the energy of the snake. The snake then moves on the image to minimize its energy. This energy depends both on the shape of the snake and the features of the image.

4.1 Snakes

Snakes or active contours are widely used in computer vision and image processing for segmentation purposes.They are also used in motion tracking [18] and shape modeling [16]. As introduced in [9], "a snake is an energy-minimizing spline guided by external constraint forces and influenced by image forces that pull it toward features such as lines and edges." What makes snakes reliable is the effect of internal forces on the shape of the contour that makes it less sensitive to noise in the image and gaps in the boundaries.

The same idea has been used in segmenting 3D volumes to yield surfaces around the target object. These active surfaces deform according to similar rules that exist for active contours. They have internal forces that resist extreme changes in surface normal and position during the segmentation.

A traditional snake is a curve $\mathbf{x}(s) = [x(s), y(s)], s \in [0,1]$, that moves on an image to minimize the energy functional

$$E = \int_0^1 \frac{1}{2} \left((\alpha |\mathbf{x}'(s)|^2 + \beta |\mathbf{x}''(s)|^2) + E_{\text{ext}}(\mathbf{x}(s)) \right) ds \tag{4.1}$$

where $\mathbf{x}'$ and $\mathbf{x}''$ are first and second derivatives of $\mathbf{x}$ with respect to s. There are two parameters in (4.1), α and β that are weighting parameters that control the snake's *tension* and *rigidity*, respectively. The external energy function in (4.1) E_{ext} is designed to have smaller values near the features of interest such as edges.

Given a gray level-image $I(x,y)$ a typical external energy designed to push an active contour toward the boundaries is

$$E_{\text{ext}}(x,y) = -|\nabla(G_\sigma(x,y) * I(x,y))|^2 \tag{4.2}$$

where $G_\sigma(x,y)$ is a two-dimensional Gaussian function with standard deviation of σ around the point (x,y), ∇ is the gradient operator and $*$ is convolution. Assuming an edge detection filter has been applied to the image the formula can be simplified to[1]

$$E_{\text{ext}}(x,y) = G_\sigma(x,y) * I_b(x,y) \tag{4.3}$$

where I_b is a binary black-on-white edge map.

Applying the Gaussian smoothing filter is very useful in tackling the problems caused by noise in the images.[2] Also the value of σ is usually used to change the "capture range" of the active contour. Increasing the value of σ reduces the sensitivity to small objects and at the same time increases the distance at which the boundary is felt by the contour. However, it is not possible to increase σ very much because the details of the image will be lost. Using this approach for increasing capture range is not very successful. A sample force field resulting from this approach is shown in Figure 4.1. The gradient of the energy

[1]Using this technique, the task of detecting the edges can be separated from the design of the external energy function. This technique has been used in this work to make use of color information in edge detection. A separate module that is capable of using color information is used to calculate the *edge map* and that simple and single-channel edge map is used in calculating the external forces. Section 5.1.1 contains the details.

[2]The gaussian smoothing filter can be applied either before the edge detector or after. In our work, we applied it before edge detection. The binary edge map was used directly in calculating the external forces.

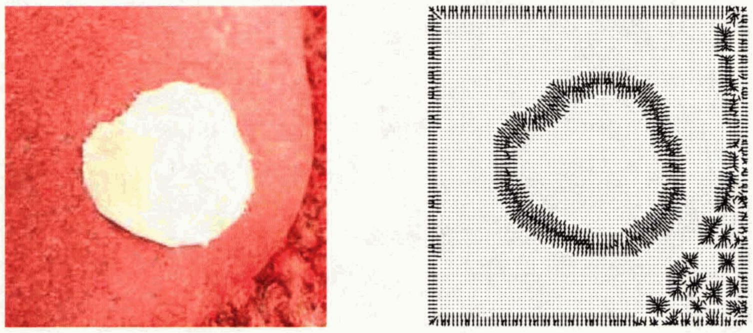


Figure 4.1: Normalized force field resulting from traditional potential forces. The image on the left is taken of a white patch on a rock. More picture are available in Chapter 6.

function (∇E_{ext}) produces this field. Note that the vectors shown in the right hand side of the figure are normalized to have a unit length (except for zero vectors). The actual lengths of the vectors are larger near the edges and get smaller as we move away from them. This has been done to make the direction of the vectors easier to observe.

A snake that minimizes E satisfies the Euler equation [9]

$$\alpha \mathbf{x}''(s) - \beta \mathbf{x}''''(s) - \nabla E_{\text{ext}} = 0 \tag{4.4}$$

Assuming $\mathbf{F}_{\text{int}} = \alpha \mathbf{x}''(s) - \beta \mathbf{x}''''(s)$ and $\mathbf{F}_{\text{ext}} = -\nabla E_{\text{ext}}$ this can also be viewed as a force balance equation

$$\mathbf{F}_{\text{int}} + \mathbf{F}_{\text{ext}} = 0$$

In order to find a solution to (4.4), the contour is made dynamic by making $\mathbf{x}$ a function of time as well as s. Then the partial derivative of $\mathbf{x}$ with respect to time is set equal to the left hand side of (4.4)

$$\mathbf{x}_t(s,t) = \alpha \mathbf{x}''(s,t) - \beta \mathbf{x}''''(s,t) - \nabla E_{\text{ext}} \tag{4.5}$$

After few iterations[3], when the solution $\mathbf{x}(s,t)$ stabilizes, the term $\mathbf{x}_t(s,t)$ disappears and we get a solution to (4.4). This is a gradient descent algorithm designed to find a minimum for E.

[3]By discretizing the equation along the time axis and solving the discrete system iteratively we find a solution to it.

To add additional flexibility to external forces, more complicated fields may be used to push the snakes. A good example of this approach are the balloon models[4] in which pressure or *inflation* forces are applied in addition to edge map potential forces. These kind of extra forces are used to move the snake toward the features of interest and overcome the short capture range of potential forces. The other reason that necessitates use of other external forces in addition to potential force is the fact that convergence to concave regions is a problem in traditional snakes, because the contour is not made to progress into the concavities.

Balloon models, even though they solve some of these problems, have their own challenges. For example, the strength of the pressure forces may be difficult to set, since they must be large enough to overcome weak edges and noise, but small enough so they do not overwhelm legitimate boundary forces.

4.2 Gradient Vector Flow

One of the other methods for applying external forces to the contour is *Gradient Vector Flow* as described in [20, 21]. The GVF field points toward the object boundary when very near to the edge, but varies smoothly over homogeneous image regions. The main advantage of the GVF field is that it can attract a snake from a long range and can force it into the concavities.

An edge map is defined as

$$f(x, y) = -E_{\text{ext}}(x, y)$$

having the property that it is larger near the edges. The field ∇f used in traditional snakes has vectors pointing toward the boundaries, but with a narrow capture range. Also, in homogeneous regions where $I(x, y)$ is constant, ∇f is zero and there is no information about nearby or distant edges. These properties were noticeable in Figure 4.1.

Xu and Prince in [20] defined the *gradient vector flow* field to be the vector field $\mathbf{v}(x, y) = (u(x, y), v(x, y))$ that minimizes the energy functional

$$\varepsilon = \iint \left(\mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |\mathbf{v} - \nabla f|^2\right) dx\, dy \tag{4.6}$$

This variational formulation follows a standard principle, that of making the result smooth when there is no data. In particular, we see that when $|\nabla f|$ is small, the energy is dominated by partial derivatives of the vector field, yielding a smooth field. On the other hand, when $|\nabla f|$ is large, the second term dominates the integrand, and is minimized by setting $\mathbf{v} = \nabla f$. The parameter μ is a regularization parameter governing the tradeoff between the first term and the second term. This parameter should be set according to the amount of noise present in the image (more noise, increase μ).

Using the *calculus of variations* [5], it can be shown that (4.6) can be minimized by treating u and v as functions of time and solving the following equations [20]

$$\begin{aligned} u_t(x,y,t) &= \mu\nabla^2 u(x,y,t) - (u(x,y,t) - f_x(x,y)) \cdot (f_x(x,y)^2 + f_y(x,y)^2) \\ v_t(x,y,t) &= \mu\nabla^2 v(x,y,t) - (v(x,y,t) - f_y(x,y)) \cdot (f_x(x,y)^2 + f_y(x,y)^2) \end{aligned} \tag{4.7}$$

in which ∇^2 is the Laplacian operator.

By replacing the potential force $-\nabla E_{\text{ext}}$ in (4.5) by $\mathbf{v}(x,y)$ we get

$$\mathbf{x}_t(s,t) = \alpha\mathbf{x}''(s,t) - \beta\mathbf{x}''''(s,t) + \mathbf{v} \tag{4.8}$$

which is used as the dynamic equation of *GVF snake.*[4] A simplified version of the Matlab code written by Xu and Prince that is used in GVF calculation is displayed in Program 4.1.

As can be seen in Figure 4.2, the resulting field of solving (4.8) has a much better capture range and is much more tolerant to the initial contour. Note that because of the way the edge detection module works, the borders of the image are treated as edges. So if the snake gets close enough to the border, the "gravity" of the border outperforms that of the actual object. This must be taken into consideration when initializing the contour.

4.3 3D Contours

Most of snakes work has been either on 2D deformable contours or 3D deformable surfaces, while the focus of this thesis is on 3D active contours. This work is on segmenting objects on the surface of a sub-manifold.

One example of such objects are oral lesions. The lesion is on the surface of a 3D object like a tongue. Calculating the 3D contour around the lesion is very helpful for measuring

[4]This equation can also be solved by discretizing along the time axis and iterating.

```
function [u,v] = GVF(f, mu, ITER)
%GVF Compute gradient vector flow.
%      [u,v] = GVF(f, mu, ITER) computes the
%      GVF of an edge map f. mu is the GVF regularization coefficient
%      and ITER is the number of iterations that will be computed.

[m,n] = size(f); fmin = min(f(:)); fmax = max(f(:));
f = (f-fmin)/(fmax-fmin);   % Normalize f to the range [0,1]

[fx,fy] = gradient(f);      % Calculate the gradient of the edge map
u = fx; v = fy;             % Initialize GVF to the gradient
SqrMagf = fx.*fx + fy.*fy;  % Squared magnitude of the gradient field

% Iteratively solve for the GVF u,v
for i=1:ITER,
    % 4*del2(u) is a finite difference approximation
    % of Laplace's differential operator applied to u
    u = u + mu*4*del2(u) - SqrMagf.*(u-fx);
    v = v + mu*4*del2(v) - SqrMagf.*(v-fy);
end
```

Program 4.1: Matlab code to calculate gradient vector flow

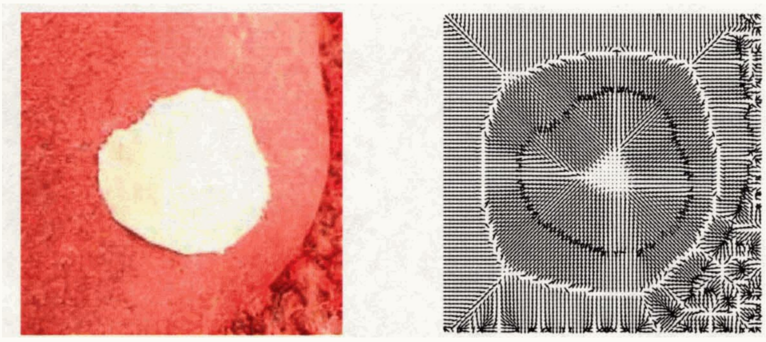


Figure 4.2: Force field resulting from gradient vector flow

its dimensions. Using images taken with a calibrated camera the dimensions of the lesion can be measured very accurately.

A 3D snake is a curve $\mathbf{x}(s) = [x(s), y(s), z(s)]$, $s \in [0, 1]$, that moves on the surface of an object to minimize the energy functional

$$E = \int_0^1 \frac{1}{2} \left((\alpha |\mathbf{x}'(s)|^2 + \beta |\mathbf{x}''(s)|^2) + E_{\text{ext}}(\mathbf{x}(s)) \right) ds \tag{4.9}$$

We will propose a method to deform such 3D contours on the surface of objects.

Chapter 5

Proposed Method

The method proposed is this thesis has two phases:

- 3D contour initialization
- Refinement

The goal of the initialization phase is to place the 3D contour at a location very close to the optimum answer. This is done by performing the segmentation on 2D images independently. The two resulting contours are then matched to find the corresponding points. Having the correspondence between the points of the two contours it is possible to calculate the 3D coordinates of the actual contour.

Without this approximation to the answer, it would be very difficult to guide the 3D contour in space toward the final contour. On the other hand the result of the initialization phase is far from perfect because of the inherent noise in the matching process. Small mistakes in finding the correspondences can result in large errors in the depth calculations [19]. As we can see in Figure 2.1 when d/f is big, small changes in α or β can result in big changes in d. In addition, because of imperfectness of the snakes algorithm on planar images and the differences of the two images, not every point from the left contour has a match on the right one and vice versa. All these problems yield a rough and noisy contour after the initialization phase, which in spite of being very close to the best answer is far from perfect. Figure 5.1 displays a sample output from the initialization phase. A few spikes along the Z axis are noticeable in the picture. These are the results of the mistakes made

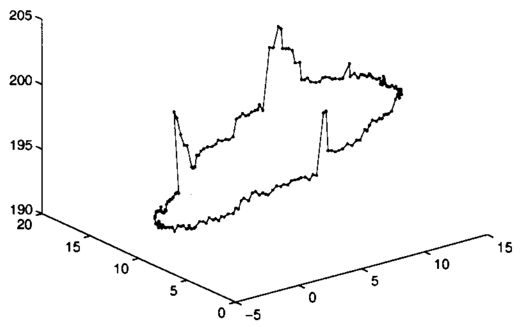


Figure 5.1: Result of 3D contour initialization phase

by the matching process.[1]

All the problems mentioned above necessitate the existence of a second *"refinement"* phase. In this phase the resulting 3D contour of the first phase is manipulated by applying external and internal forces, which in addition to driving it toward the final answer, make it smooth as in planar snakes. Satisfying the internal constraints of the snake makes sure that we get a reasonably smooth contour and reduces the effects of noise and error in 2D segmentations phase.

These errors might be caused by lighting variations, specularity, perspective distortion and other reasons. Some of these problems, particularly specularity, cause the snake to include unwanted areas or exclude wanted regions. Having two different views of the object causes the specularity effect to show up on different parts of the images. Refining the contour in the final stage by applying forces from both images at the same time reduces the effect of such factors.

5.1 Initialization Phase

Initializing the contour by approximating the final answer helps in stabilizing it on the surface of the object. Without this, it would have been very hard to drag the snake in 3D space toward the region of interest and to put it on the surface.

[1]They appear mostly in the areas where the epipolar line becomes tangent to the 2D contour. For more information refer to Section 5.1.2.

The initialization phase performs a segmentation on the left and right images independently. Snakes and Gradient Vector Flows[20] are used to segment the regions.

5.1.1 2D Segmentation

Segmentation of the region of interest on the planar images is done using the traditional 2D GVF snakes. Therefore, it involves calculating the gradient vector flow on the left and right images independently and then deforming an active contour on both images to capture the boundaries.

Calculating the Gradient Vector Flow

Before calculating the gradient vector flow field, the image is smoothed out by convolving with the following gaussian kernel:

$$k(x,y) = \frac{e^{\frac{-x^2+y^2}{2\sigma^2}}}{2\pi\sigma^2} \tag{5.1}$$

The actual kernel width is truncated to 8σ and it is normalized to have $\sum\sum k(i,j) = 1$. For $\sigma > 3$, the fast fourier transformation is used to accelerate the convolution. Each color channel is smoothed independently. It is also possible to use other color spaces such as CIE L*a*b*.

After that, an edge detector filter is applied to the smoothed image. In order to make use of color information in the image, the Sobel filter is used on each color channel and the results are combined. Then a cutoff is selected using the same method used by Matlab's **edge** function.[2] The Matlab code performing this process is shown in Program 5.1. The advantage of using this cutoff value is that only stronger edges stay in the image and rest are thrown away. Also, the rest of the process will not be sensitive to the strength of the edges in the original images.

One technique that can be used in images with noisy edge maps is to perform an *erosion* followed by a *dilation* on the edge map. This is called *morphological opening*[14] and is useful for removing isolated single dots in a binary image. This can be very helpful when in spite of smoothing the image the edge map is full of such isolated dots. This usually happens when the region of interest has a harsh texture.

[2]According to Matlab documentations, the cutoff is determined based on RMS estimate of noise. Mean of the magnitude squared image is a value that is roughly proportional to SNR. [12]

```
k = fspecial('sobel');

r = I(:,:,1);
g = I(:,:,2);
b = I(:,:,3);

rh = conv2(r, k, 'same');
rv = conv2(r, k', 'same');
gh = conv2(g, k, 'same');
gv = conv2(g, k', 'same');
bh = conv2(b, k, 'same');
bv = conv2(b, k', 'same');

E = sqrt(rh.^2+gh.^2+bh.^2+rv.^2+gv.^2+bv.^2);

... take care of the image boundaries ...

% determine the threshold; see page 514 of "Digital Imaging Processing"
% by William K. Pratt
% Determine cutoff based on RMS estimate of noise
% Mean of the magnitude squared image is a
% value that's roughly proportional to SNR
% This is the same method used by Matlab's edge function
cutoff = 4 * mean2(E);
E = double(E > cutoff);
if params.erdi, % perform erosion and dilation?
   se = strel('disk', params.erdi, 0);
   E = imopen(E, se);
end
```

Program 5.1: Matlab code to detect edges in a color image

The problem of using (4.7) in calculating the gradient vector flow is that in high resolution images as we get farther from the edges it takes more iterations to propagate the effect of the edges. This can make the process of calculating the GVF field very time consuming. In order to overcome this problem we used a multi-resolution technique. The GVF field is first calculated for a scaled down version of the image with a length of about 200 pixels.[3] This low resolution field can then be used either to interpolate the high resolution field's values, or to fill in the initial values of higher resolution field before performing more iterations. This makes the propagation process much faster. Our studies showed that simply interpolating the values for high resolution image is sufficient for getting a precise field.

Deforming the 2D Snakes

The 2D contours are initialized by the user roughly around the final answer. They can simply be circles around the region of interest. Using (4.7) and (4.8) deforming the snakes on the 2D plane is straightforward. By discretizing the time axis and the value of s, deformation is performed in multiple iterations. In each iteration t, a $\Delta\mathbf{x}(s,t)$ is calculated for every value of s. The new location of each node in the contour is calculated as

$$\mathbf{x}(s,t+1) = \mathbf{x}(s,t) + \Delta\mathbf{x}(s,t) \tag{5.2}$$

The problem that is encountered after a few iterations is that some of the nodes might become too far or too close. Neither case is desirable because

- Nodes that are too close waste computations. Having tens of nodes in one place is no better that having just one. Considering that we might have a limited number of nodes on the contour, these nodes also reduce the precision in the regions with less nodes.
- Nodes that are too far cause part of the contour to become less affected by external forces coming from the image. The precision of the contour is also jeopardized in the sparse areas.

For this reason every few iterations, the nodes on the contour are rearranged to maintain a minimum and a maximum distance from each other. During this *interpolation* process, new

[3]Only the rectangle around the area of interest (specified by user) is used in calculating GVF, not the whole image. The length of this rectangle is scaled down to 200 pixels, not the whole image. The original images used in our experiments were 5 megapixels or 1944 × 2592.

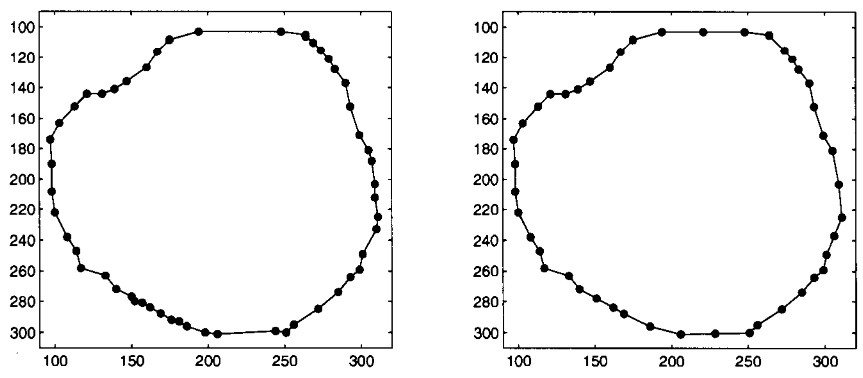


Figure 5.2: Snake interpolation; before (left) and after (right)

locations on the continuous contour are selected as representative nodes and the locations of those nodes are interpolated from the location of the old nodes. The effect of this procedure is shown in Figure 5.2.

In order to remove the nodes that are too close, the distance of each node to its next neighbor is calculated. Then the nodes with the distances shorter than a minimum threshold are eliminated. After that, new nodes are inserted between the nodes that have distances longer than a maximum threshold. The coordinates of these new nodes are interpolated based on the other nodes. The insertion procedure is repeated until there are no adjacent nodes with distances longer than the specified threshold.

5.1.2 Finding the Corresponding Points

As indicated in Chapter 2 the most challenging task of stereo vision algorithms is finding the correspondences between left and right images. Rectifying the images and only searching the epipolar line considerably enhances the performance of the matching process.

In the case of matching the left and right contours there is no need to rectify the images. As displayed in Figure 5.3, any particular point on the left image such as X_l is the projection of a point O in 3D space that has to be on the line passing X_l and O_l (the location of left camera or the origin of the left camera reference frame). The projection of this line on the right image that is shown in yellow and passes C_r and X_r, is called the epipolar line and the corresponding point of X_l is constrained to lie on this line. The coordinates of this line can

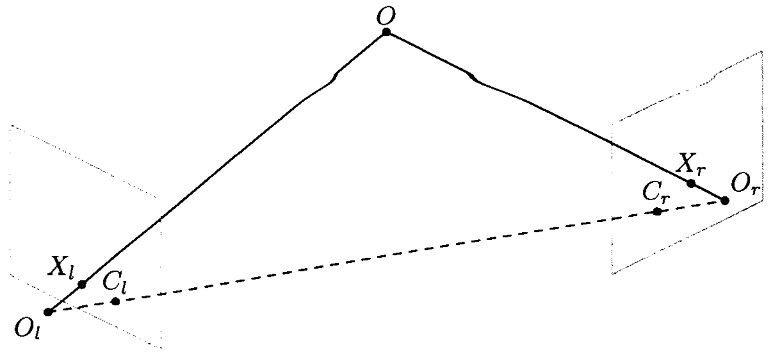


Figure 5.3: Epipolar Geometry

be found by projecting 2 arbitrary points of the line passing C_l and X_l on the right image.

Figure 5.4 shows the epipolar line of the specified point on the left image as a dashed yellow line on the right image. The corresponding point of that particular point on the left image is constrained to be on this line.

What helps us even more in matching the two contours is that we know that corresponding point to each point of one contour is somewhere in the neighborhood of the other contour. Intersecting this region with the epipolar line results in a very small region that needs to be checked to find the matching for any point. This region is shown with black dots around it on the right image in Figure 5.4.

The matching function used to measure the similarity of two patches $\tilde{p}_L$ and $\tilde{p}_R$ of size $h \times w \times 3$ around the points p_L and p_R is defined as

$$x(p_L, p_R) = \sum_{1 \le i \le h, 1 \le j \le w, 1 \le k \le 3} |\tilde{p}_L(i,j,k) - \tilde{p}_R(i,j,k)| \tag{5.3}$$

Kanade and Okutomi in [8] propose an algorithm with an adaptive window size to maximize the performance of stereo matching. However, for the sake of simplicity we decided to use a fixed window size.

5.1.3 Removing the Irrelevant Nodes

The resulting 3D nodes from the previous section form a contour with some big spikes as a result of noise and error. It is helpful to remove or rectify these kind of nodes before starting the final phase. The nodes that are known to be irrelevant beforehand can be detected using several techniques. One simple way of doing so is assuming the nodes are almost on a plane

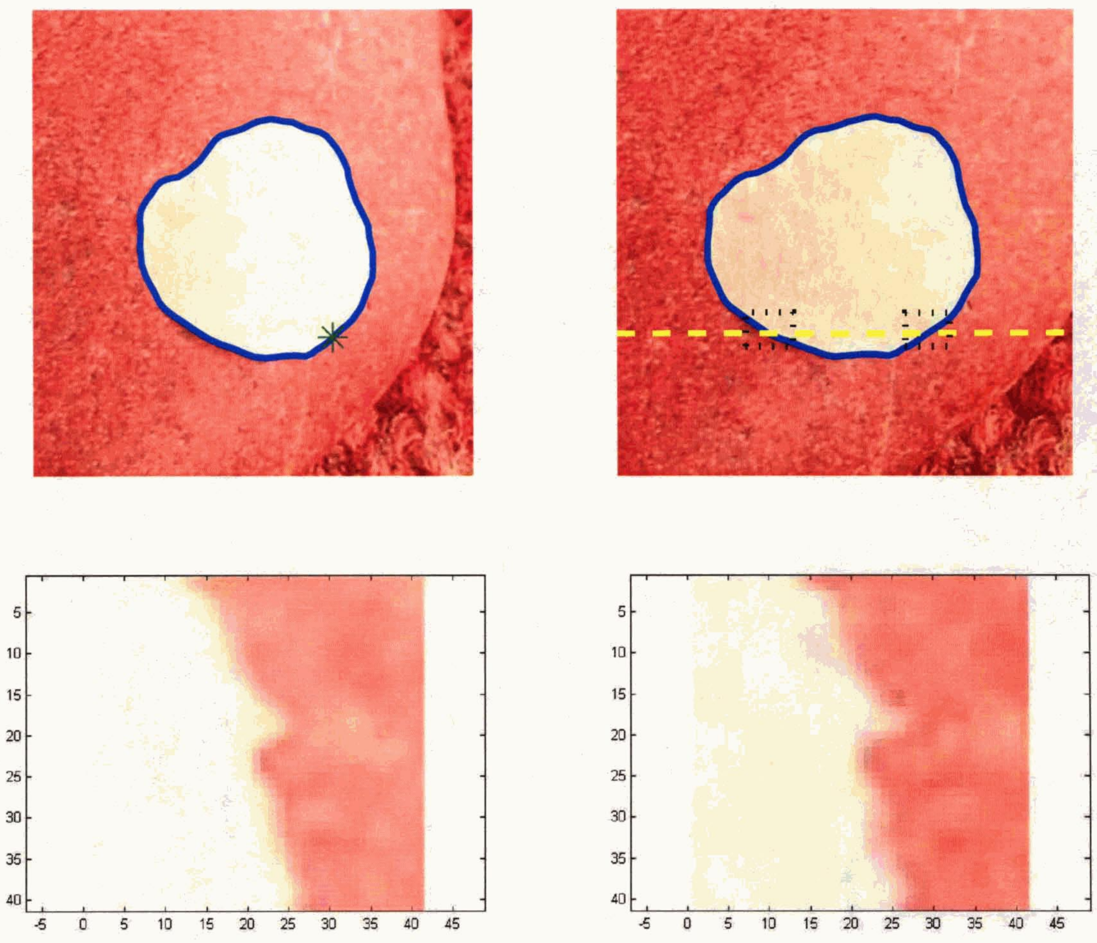


Figure 5.4: Matching left and right contour nodes; The epipolar line on the right image is shown in dashed yellow. The region that is checked for finding the match for the point on the left image, is marked with blue stars on the right image. Images on the bottom are enlargements of sample $\tilde{p}_L$ and $\tilde{p}_R$ patches that have most similarity.

```
% perform a principle component analysis
[coeff, score, latent] = princomp(xyz);

% sc3 is big for “bad” xyz’s. It’s the weight of the “bad” component
% in xyz’s. assuming xyz’s are almost on a plane.
sc3 = score(:,3);

% remove the ones with sc3 > avg+stddev
xyz(:,find(sc3 > (mean(sc3)+std(sc3)))) = [];
```

Program 5.2: Matlab code to remove irrelevant nodes in initialization phase

and removing the ones with enormous distance from the plane that fits to all nodes. We did that by performing a principle component analysis on the coordinates of all the nodes. The first two principle component vectors have the maximum variance and are on plane that best fits the nodes. The third vector is normal to this plane and if we translate the node coordinates to this new coordinate system, the nodes with highest third coefficients are the ones that are farthest from the plane. We chose the ones whose absolute value of coefficients where higher than mean plus standard deviation of all.[4]

The code fragment that performs this removal can be seen on Program 5.2. Figure 5.5 shows the result of running this code on the nodes of the contour displayed in Figure 5.1.

5.2 Refinement Phase

After initializing the contour to a location very close to final answer, it is time to “refine” it to get the best result. In order to perform this refinement we define the external and internal forces affecting a snake in 3D space. As before [9] we have the following formula for computing the energy of an snake:[5]

[4]As we will see in Section 6.3 this approach cannot be used when the contour is not somewhat planar. Other approaches such as performing local PCA for each nodes neighborhood can be used in such cases. Because this step does not play a significant role in our work we can skip it in non-planar cases. Internal forces of the snakes will cancel out the negative effect of these bad initializations.

[5]At the early stages of this work, when we did not have the *contour initialization phase*, we were not successful in absorbing the snake toward the lesion being segmented. The contour was very unstable in 3D space when it was far from the final answer. In order to make it more stable we tried adding more complicated internal forces to Equation 4.1 such as a force that resisted twisting in the snake. Also our first approach to 3D snake was based on pressure forces (inflation) to push the snake toward the lesion. These forced proved to be very problematic for a 1D contour in 3D space. Later we decided to use *gradient vector*

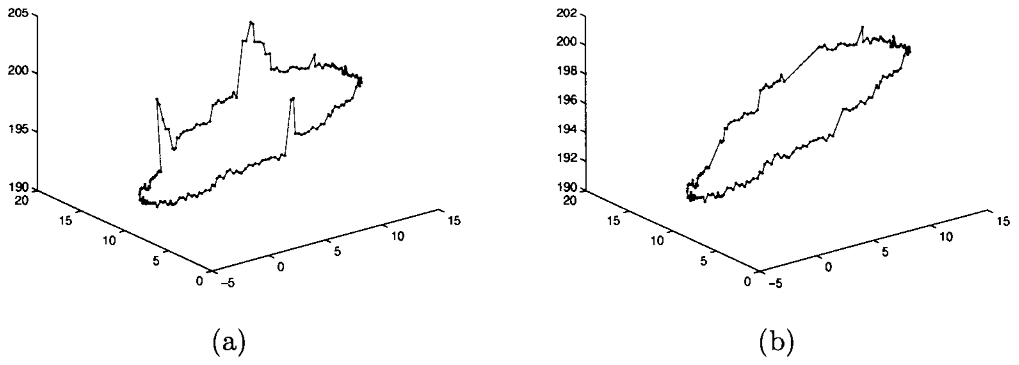


Figure 5.5: Before (a) and after (b) cutting irrelevant nodes in initialization phase

$$E = \int_0^1 \frac{1}{2}(\alpha|\mathbf{x}'(s)|^2 + \beta|\mathbf{x}''(s)|^2) + E_{\text{ext}}(\mathbf{x}(s))\, ds \tag{5.4}$$

The difference between a 2D and a 3D snake (3D in the context of this work) is in the E_{ext} factor. As previously mentioned in Chapter 4, the formula usually used as E_{ext} is

$$E_{\text{ext}}(x, y) = -|\nabla(G_\sigma(x, y) * I(x, y))|^2 \tag{5.5}$$

as introduced in [9]. Xu and Prince [20] used the *generalized diffusion equations* known to arise in such diverse fields as heat conduction, reactor physics, and fluid flow [6] to calculate a gradient vector flow to apply the external forces on the snake.

These forces work perfectly well for planar snakes, but cannot be used directly when the snake is in 3D space and the forces are defined on planes of projected images. In order to apply the external forces, we need to calculate a 3D gradient vector flow. We will not actually calculate the vectors for all the points in 3D space, but will provide a function to calculate the direction and strength of the vector for any given point. In each iteration this function is applied on each node of the contour to calculate the external force.

5.2.1 Applying External Forces

In addition to internal snakes forces, external forces from the image still need to be applied to the contour. One obvious force needed in 3D space is a force to push the contour in a

flows for that purpose. After adding the initialization phase, the contour was pretty stable near the final result and we decided not to change the original snake energy formula.

way that its projections move towards the edges of the region of interest on both left and right images. The other force needed is one that keeps the nodes on the surface of the sub-manifold. This force tries to make the projections of the nodes on the images correspond to each other.

Image Gradient Force

In order to move the nodes in 3D space we extended the gradient vector flow field from 2D images to 3D space. Before explaining the details we define the following notation:

$\pi_L(\mathbf{p})$ **and** $\pi_R(\mathbf{p})$ are the projections of the 3D point $\mathbf{p}$ on left and right images respectively.

$\Pi(p_L, p_R)$ is the back-projection of two corresponding points p_L and p_R into the 3D space.

$\overrightarrow{\text{gvf}}_L(p_L)$ **and** $\overrightarrow{\text{gvf}}_R(p_R)$ are unit vectors in the direction of the gradient vector flow at points p_L and p_R in left and right images respectively.

For any given point $\mathbf{p}$ in the space we define $\overrightarrow{\text{gvf}}(\mathbf{p})$ as

$$\begin{aligned} \overrightarrow{\text{gvf}}(\mathbf{p}) &= \frac{1}{|\vec{v}|}\vec{v} \\ \vec{v} &= \Pi(p_L + \overrightarrow{\text{gvf}}_L(p_L), p_R + \overrightarrow{\text{gvf}}_R(p_R)) - \mathbf{p} \\ p_L &\equiv \pi_L(\mathbf{p}) \\ p_R &\equiv \pi_R(\mathbf{p}) \end{aligned} \tag{5.6}$$

This vector field applies the 2D vector fields of the images on the projections of a 3D field and then back-projects the result into the 3D space. Projection of $\mathbf{p}$ on the left image is p_L. This point wants to move in the direction of $\overrightarrow{\text{gvf}}_L(p_L)$ on the left image and go to $p_L + \overrightarrow{\text{gvf}}_L(p_L)$. Similarly, p_R on the right image wants to go to $p_R + \overrightarrow{\text{gvf}}_R(p_R)$. If we assume that the resulting points correspond to each other, back-projection of them is a point in 3D space that moves both projections of $\mathbf{p}$ on the left and right images according to their own GVF fields. If we push $\mathbf{p}$ towards the $\Pi(p_L + \overrightarrow{\text{gvf}}_L(p_L), p_R + \overrightarrow{\text{gvf}}_R(p_R))$ we are satisfying both left and right potential forces. The difference between $\mathbf{p}$ and this new point is called $\vec{v}$ and $\overrightarrow{\text{gvf}}(\mathbf{p})$ is defined as normalization of $\vec{v}$. The vector $\vec{v}$ does not have a meaningful length. Only the direction of it is important, which is why we normalize it.

Note that the direction we are choosing for $\overrightarrow{\text{gvf}}(\mathbf{p})$ does not necessarily maintain the node on the surface of the object. It merely tries to satisfy left and right potential forces and may push the node off the surface. We define another external force (correspondence force) responsible for maintaining the nodes on the surface. Since each iteration moves the nodes a short distance in the space, eventually the correspondence force compensates for the occasional bad maneuvers of the image gradient force.

Correspondence Force

In order to keep the points on the surface of the sub-manifold we define an energy function $m : \mathbb{R}^3 \longrightarrow \mathbb{R}$ for all points in the space that is minimal on the surface and increases as we move farther. This function only needs to work properly near the surface as the initialization phase has already placed the 3D contour close to it. The matching function defined in (5.3) has the same properties. So we can define $m_1(\mathbf{p})$ as

$$m_1(\mathbf{p}) = x(\pi_L(\mathbf{p}), \pi_R(\mathbf{p}))$$

To take care of the discontinuous and noisy nature of the m_1 function, we convolve it with a Gaussian filter to smooth out the noise.

$$G_\sigma(x, y, z) = \frac{e^{-\frac{x^2+y^2+z^2}{2\sigma^2}}}{2\pi\sigma^2}$$

$$m(\mathbf{p}) = (G_\sigma * m_1)(\mathbf{p}) \tag{5.7}$$

The gradient of such an energy function pushes the snake nodes towards the surface.

$$\overrightarrow{\text{cor}}(\mathbf{p}) = (-\nabla m)(\mathbf{p}) \tag{5.8}$$

In order to calculate $\overrightarrow{\text{cor}}(\mathbf{p})$, $m_1(\mathbf{p})$ needs to be calculated not only at $\mathbf{p}$ but in its neighboring points in 3D space as well. This enable us to perform the convolution and get a stable result for $\overrightarrow{\text{cor}}(\mathbf{p})$.

Combined External Force

The two defined 3D forces (image gradient and correspondence) are added together each with a weighting coefficient and are used as the external force affecting the snake nodes.

$$\mathbf{F}_{\text{ext}}(\mathbf{p}) = \kappa_g \overrightarrow{\text{gvf}}(\mathbf{p}) + \kappa_c \overrightarrow{\text{cor}}(\mathbf{p}) \tag{5.9}$$

5.2.2 Calibration Problems

One of the problems that arises while trying to calculate and apply external forces is that in order to get feedback from 2D images with regard to the quality of the contour we need to project the 3D contour onto the two images. As we mentioned in Section 3.3 small calibration errors (which always occur) will result in back-projection lines missing each other. In that case the point with minimum total distance to the lines is chosen instead of the intersection point. The problem is later, when we project that 3D point onto the images, it will not map exactly on the same pixels that were used to calculate its coordinates. Our studies showed that this problem can be modeled as a shift in 2D coordinates.

We use a translation vector for each of the left and right images to compensate for the effect of inaccurate calibration. Assuming $\pi_L(\mathbf{p})$ is the projection of point $\mathbf{p}$ on the left image and p_L is the point on the left image used to calculate the external force imposed on $\mathbf{p}$ from that image

$$p_L = \pi_L(\mathbf{p}) + \vec{c}_L$$

where $\vec{c}_L$ is the compensating vector for the left image. For the right image a different vector should be used.

$$p_R = \pi_R(\mathbf{p}) + \vec{c}_R$$

It is easy to see $\vec{c}_L \approx -\vec{c}_R$.[6] Hence, assuming $\vec{c} = c_L$

$$p_L = \pi_L(p) + \vec{c}$$

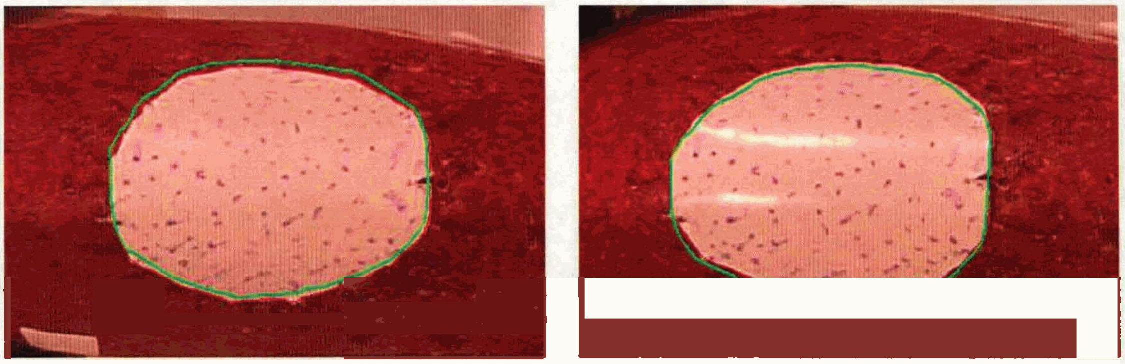
$$p_R = \pi_R(p) - \vec{c}$$

Figure 5.6 illustrates the effect of applying the compensation vector when projecting the contour on left and right images.

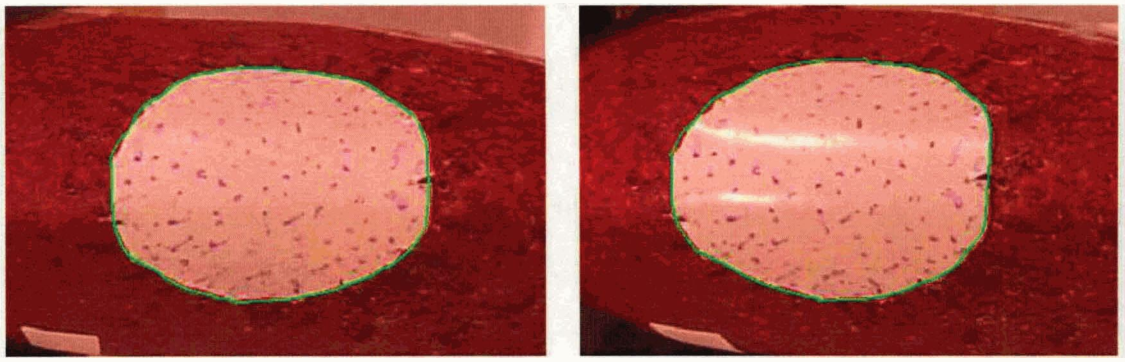
Calculating $\vec{c}$ is as simple as comparing the projection of the initial contour with the actual 2D contours used in back-projection and subtracting node locations on the plane. It is shown in Program 5.3.

After several iterations of applying external and internal forces, the 3D contour finally converges to a smooth and curved one that lies on the surface of the object. An example of such a contour is displayed in Figure 5.7 in blue together with the initial contour in red. All the coordinates are in millimeters in left camera reference frame.

[6]The reason is that the point selected instead of the intersection point is of equal distance to both lines coming from the left and right images. Furthermore, the shortest path from that point to each line is perpendicular to it. This results in equal amounts of shift in opposite directions on left and right projections.



Contour shift after projection



Compensation for contour shift

Figure 5.6: Compensating for contour shift after projection

```
% "xyL" and "xyR" are the left and right contours.
% "env" contains camera information.
xyz = backProject(xyL, xyR, env);
xyL2 = mapOnLeftImage(xyz, env);
xyR2 = mapOnRightImage(xyz, env);
comp = round(mean(xyR - xyR2 - xyL + xyL2) / 2);
```

Program 5.3: Matlab code to calculate compensation vector

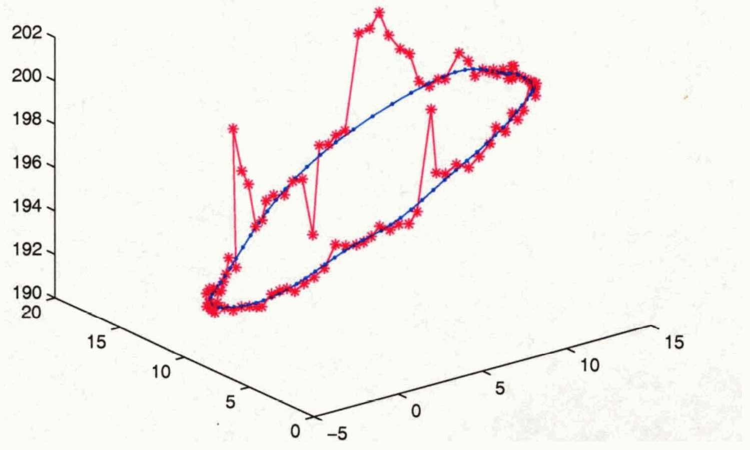


Figure 5.7: Final result after refinement in blue together with the initial contour in red

5.3 Summary

The method proposed in this chapter can be summarized as follows:

1. Segment left and right images independently. (results in two 2D contours)
2. Establish correspondence between left and right contours.
3. Back-project matching point into 3D space.
4. Remove obvious noise (irrelevant nodes) from the 3D contour if possible.
5. Project the 3D contour onto the left and right images and compare with original 2D contours to calculate the compensation vector.
6. Move the snake in 3D by applying internal, GVF and correspondence forces. Apply the compensation vector when projecting from 3D to 2D.

Chapter 6

Experiments and Results

6.1 Hardware Specifications

We used two Sony Cyber-shot DSC-V1 cameras bound together in a specially made frame to make a stereo camera. Cameras were used in 5 megapixel, macro mode with high JPEG quality. We used a LANC Shepherd remote control to synchronize the cameras. Even though the shutters were synchronized it was not possible to use the flash. The cameras were neither synchronized enough to use a single flash nor out of sync enough to use both flashes. The stereo system is shown in Figure 6.1.

The programs were run in Matlab 7.0.1 on a PC with a Pentium 4 1.7GHz CPU and 2GB of RAM. The amount of RAM on this machine was not relevant to the measured times as the program was non memory intensive. (30MB was used for keeping the left and right images in memory and the rest was insignificant.)

6.2 Plain Patch on a Smooth Rock Surface

This was the simplest case used in writing and debugging the code. Stereo images of a rock with a white patch on it where taken. These two pictures are shown in Figure 6.2. Cameras were calibrated using a set of 12 pictures taken from a 7×9 checkerboard with square size of 3cm $\times$ 3cm.

All the figures presented in previous chapters for illustrating the algorithms (except Figure 5.6) are the actual results of this case. Figure 5.1 displays the result of performing the initialization phase (before removing irrelevant nodes) on this case. Figure 5.4 illustrates

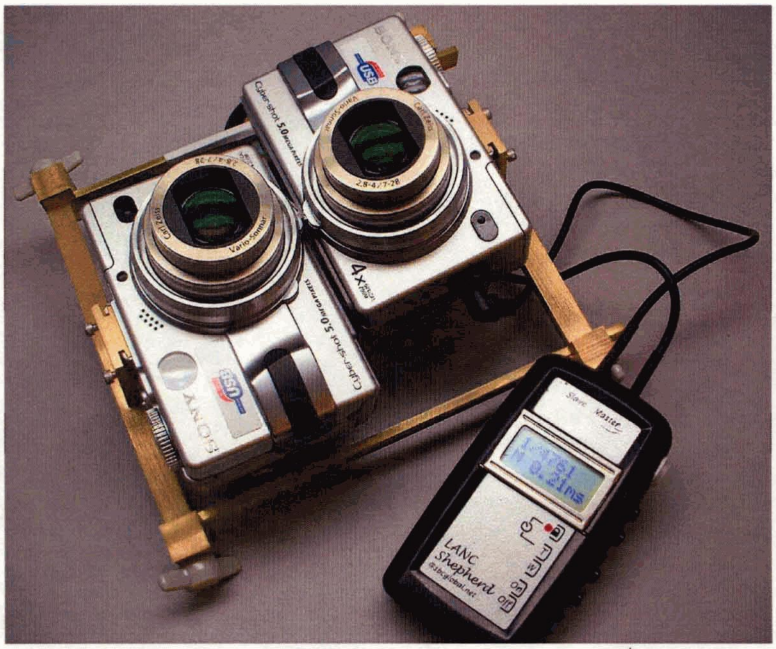
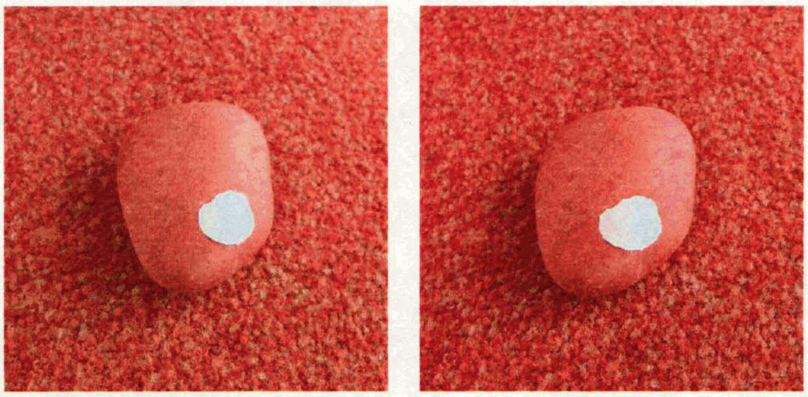


Figure 6.1: The stereo camera used in our experiments



(L) (R)

Figure 6.2: Left (L) and right (R) shots of the plain patch on smooth rock

Operation	**Time**	**Parameters**
Smoothing/Edge Detection	3.6s	$\sigma = 3$
GVF Calculation	2.1s	$\mu = 0.2$, 80 iterations
2D Deformation	4.7s	$\alpha = \beta = 0.05$, $\kappa = 0.6$ 150 iterations $d_{\text{min}} = 1$, $d_{\text{max}} = 3$
Contour Matching	14.6s	100 nodes 40×40 Patch Size Epipolar Neighborhood = 5
Refinement	1.0s	$\alpha = \beta = 0.05$, $\kappa = 0.1$ 50 iterations
Total Time	26.0s	

Table 6.1: Runtime details for plain patch case

the matching process. Figure 5.5 demonstrates the effect of removing the irrelevant nodes. The final result together with initial status of the contour is also displayed in Figure 5.7.

Table 6.1 has a summary of parameters and running times of each stage of the algorithm for this case.

d_{min} **and** d_{max} are the parameters used in snake interpolation for adjusting minimum and maximum distance between the nodes.

Patch Size is the size of the neighborhood used for matching patches around the nodes. (in pixels)

Epipolar Neighborhood specifies how far from the epipolar line we look for finding the best match. (in pixels)

κ is the weight assigned to the external forces.

6.3 Bent Patch on Rock's Edge

This case was designed to investigate the performance of the system on non-planar curves. The same rock used in 6.2 was used again, but the patch was put on the edge of the rock and it was bent to a great extent. As it can be seen in Figure 6.3 the difference between left and right shots is very small around the patch. Surprisingly the system was again successful in measuring the size and reconstructing the 3D shape of the patch's boundary.

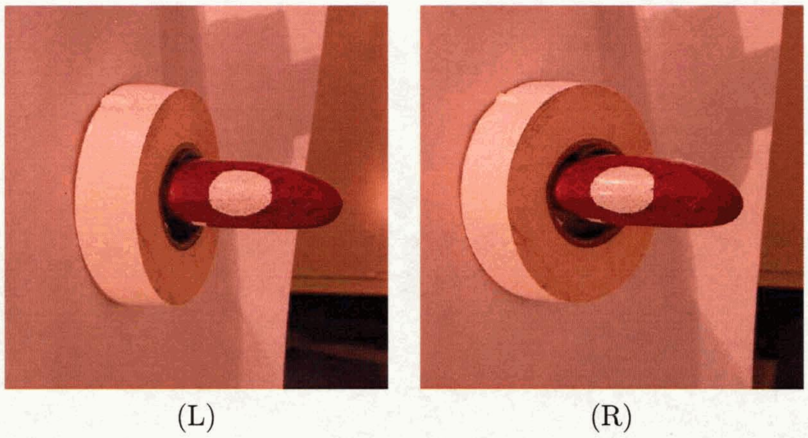


Figure 6.3: Left (L) and right (R) shots of the bent patch on rock's edge

Calibration was done with a smaller checkerboard with square size of 1cm × 1cm which resulted in slight inaccuracy in calibration demonstrated in Figure 5.6.

The initialization phase as shown in Figure 6.4, was able to come up with a good estimate for the contour. Because of the non-planar shape of the patch, the simple method described in 5.1.3 for removing irrelevant nodes can no longer be used and we simply skipped that step. By performing more iterations in the refinement phase, the location of these nodes will be rectified.

After initializing the contour the refinement phase was done. As it can be seen in Figure 6.4, this phase compensates for the errors made in initializing the contour. The initial curve is displayed in red with stars representing the nodes and the final result is in blue with dots as nodes.

Table 6.2 contains the runtime details for this case.

We also compared the size of the final contour with physical measurements we performed using a 0.1mm vernier caliper. Two lengthes were compared; l_1 and l_2 as it is shown in Figure 6.5.

The error, even though small, is mostly caused by the following reasons

- Inaccuracy of the calibration causes errors in projections and back-projections. The compensation vectors are helpful for reducing this error but are not successful in removing it completely.
- Inaccuracy of snakes in detecting edges causes the left and right contours not to be completely corresponding.

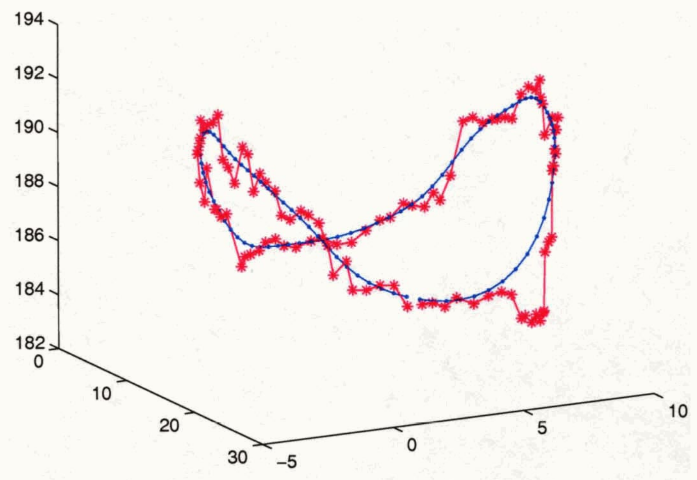


Figure 6.4: Final result for the bent patch in blue; together with the initial contour in red

Operation	**Time**	**Parameters**
Smoothing/Edge Detection	2.8s	$\sigma = 3$
GVF Calculation	1.5s	$\mu = 0.2$, 80 iterations
2D Deformation	4.8s	$\alpha = \beta = 0.05$, $\kappa = 0.6$ 180 iterations $d_{\text{min}} = 1$, $d_{\text{max}} = 3$
Contour Matching	16.8s	100 nodes 40×40 Patch Size Epipolar Neighborhood = 5
Refinement	0.9s	$\alpha = \beta = 0.05$, $\kappa = 0.1$ 50 iterations
Total Time	26.8s	

Table 6.2: Runtime details for the bent patch case

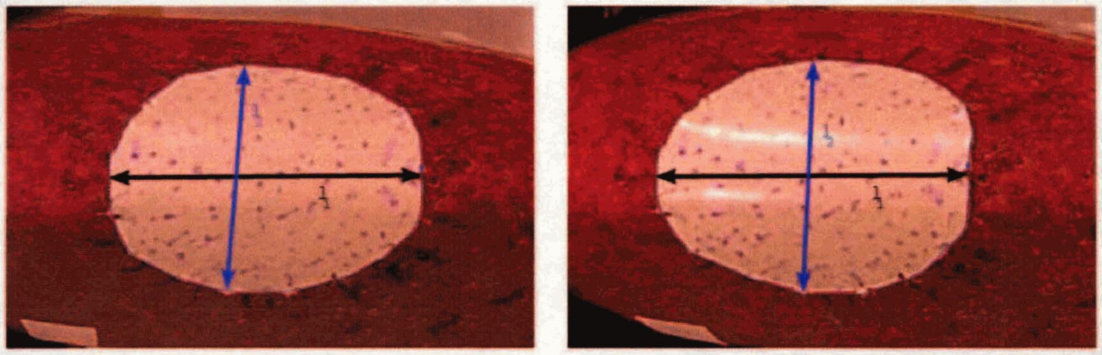


Figure 6.5: Physical measurements performed on the contour

	Calculated	**Measured**	**Difference**
l_1	17.0mm	16.9mm	%0.6
l_2	12.9mm	12.7mm	%1.6

Table 6.3: Calculated and measured lengths in bent patch case

In order to demonstrate the benefits of stereo imaging in measuring lengths, we also measured the perimeter of the patch using 3 different approaches.

- *Physical measurement.* We wrapped a string around the patch and then measured the length of the string using a vernier caliper. We tried to be as precise as possible. It's fair to say that we had a precision of 1mm, so we rounded all the results to the nearest millimeters.

- *3D contour measurement.* We measured the length of the final contour displayed in Figure 6.4. Since all the coordinates were already in millimeter, this task was straightforward.

- *2D contour measurement.* We measured the length of 2D contour segmenting the patch in the left image in pixels. Then we divided this number by the length of the l_1 in the image in pixels. This ratio was then multiplied by the actual length of l_1 measure using a caliper. This was to simulate the effect of having a probe with know length in the image. This length is best estimate we can get from a 2D image for the length of the perimeter.

The results of these measurements are in Table 6.4. Whether the physical measurement is

Physical measurement	56mm	
3D contour measurement	55mm	2% error
2D contour measurement	47mm	16% error

Table 6.4: Measuring the perimeter of the patch using different approaches

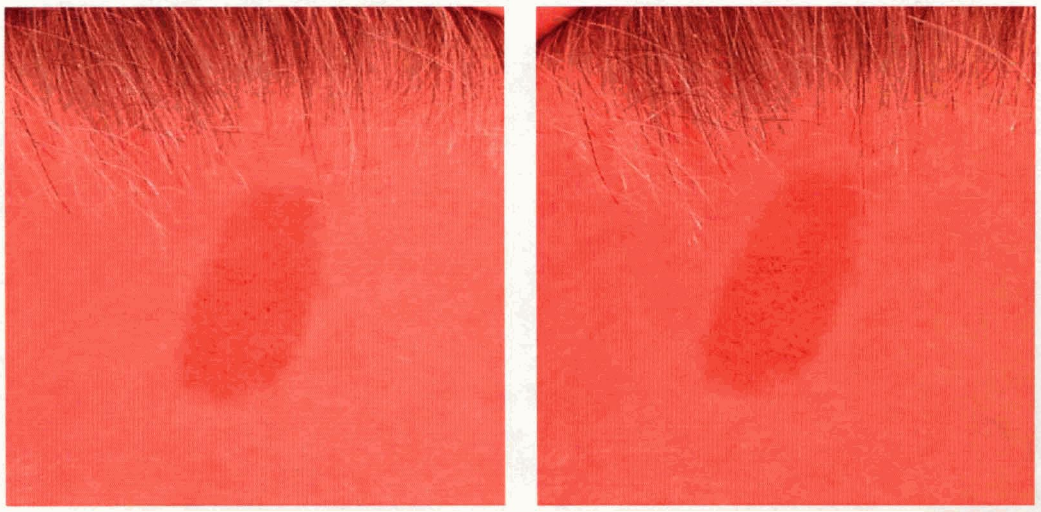


Figure 6.6: Left and right shots (cropped) in the skin texture case

more accurate or the 3D contour technique is debatable, but it can be seen that the 2D measurement is obviously far from accurate. Without having information about the depth of an image, it is not possible to judge distances. In this case, the 2D measurement has a 16% error.

6.4 Skin Texture

This case was designed to evaluate the performance of the system when the edges are not clear. This is usual in medical images. The region of interest is a birthmark on the forehead of an individual. As it can be seen in Figure 6.6, the edges are blurry and vague. Due to large amount of noise in the images, larger values were selected for α and β compared to the previous cases. ($\alpha_{\text{old}} = \beta_{\text{old}} = 0.05$, $\alpha_{\text{new}} = \beta_{\text{new}} = 0.35$)

Back-projecting the two 2D contours into 3D space results in the contour shown in Figure 6.7. After deforming this contour in 3D space for several iterations, we will reach the final result as shown in Figure 6.8.

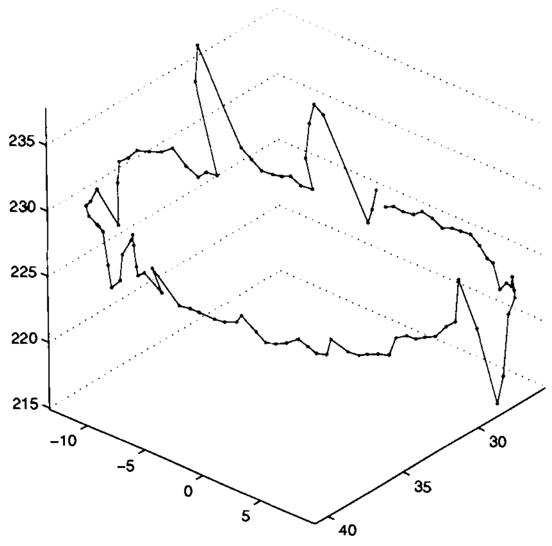


Figure 6.7: Result of the initialization phase for the skin texture

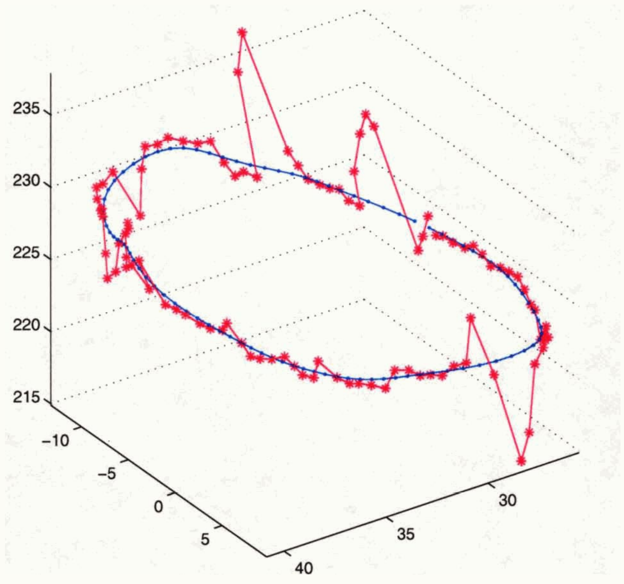


Figure 6.8: Final result for the skin texture in blue; together with the initial contour in red

Projecting this final contour onto the left and right images, results in 2 different 2D contours that are illustrated in Figure 6.9. As it can be seen, despite the poor contrast between the birthmark and the forehead's skin, the final result is satisfactory.

6.5 Stereo Camera as a Measurement Device

Observing the accuracy of our stereo system in measuring lengths, we decided to study its performance in doing so independent of the segmentation part. We used the stereo system to shoot pictures of a metal object. The pictures are displayed in Figure 6.10. In order to get the best results, the cameras were calibrated right before shooting the pictures.

We measured the width of the object near its head, middle and tail using both vernier caliper and the stereo system. Matching points were provided to the system manually. Table 6.6 shows the results of our study. As it can be seen, as we get farther from the camera the accuracy is reduced.

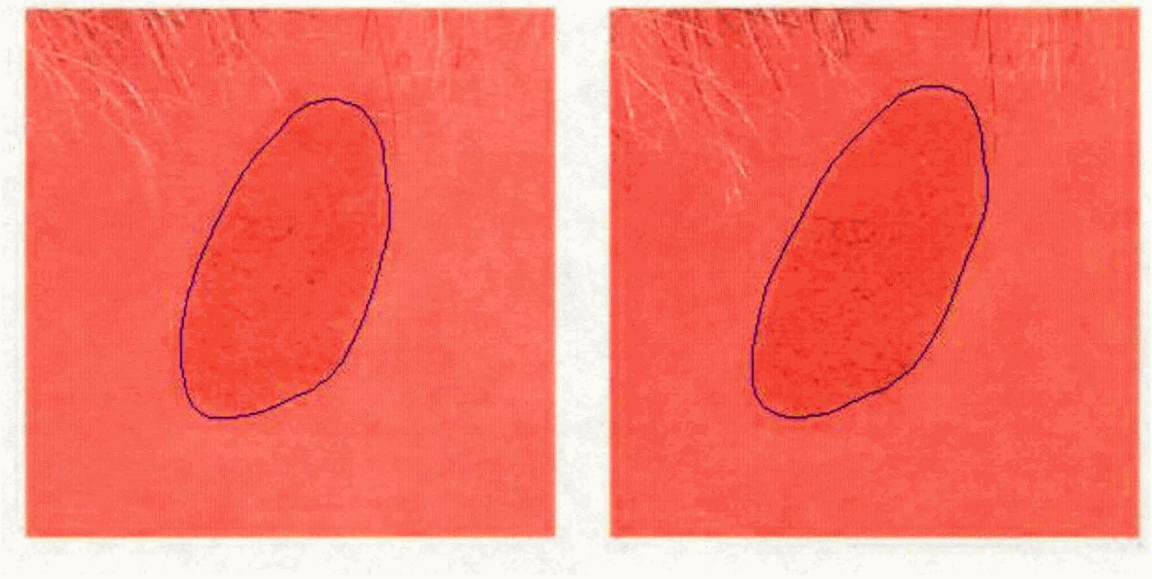


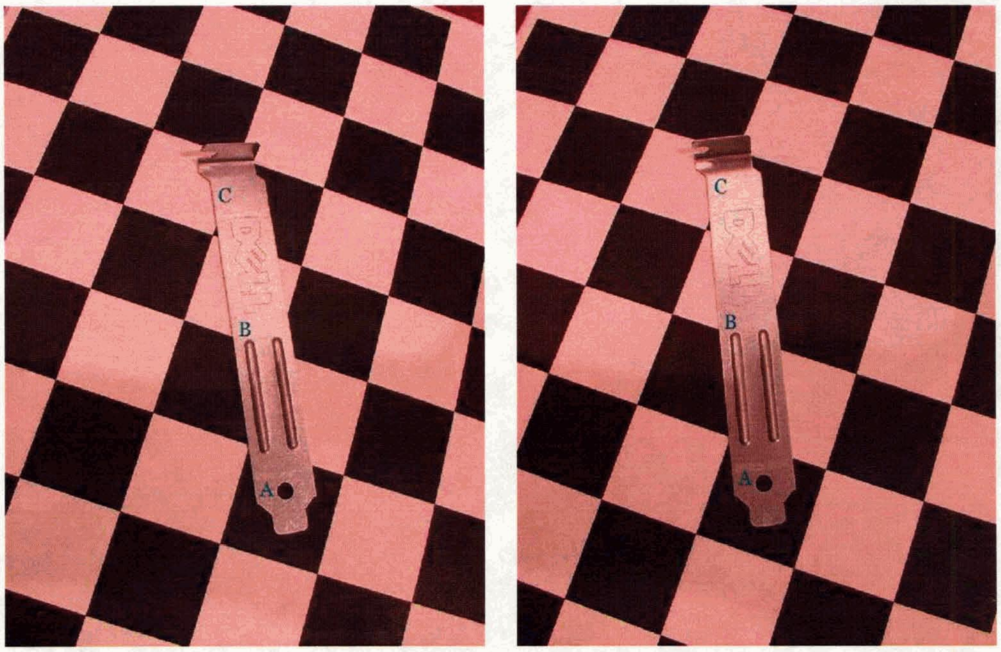
Figure 6.9: Projection of the final contour on skin images

Operation	Time	Parameters
Smoothing/Edge Detection	2.6s	$\sigma = 3$, window size: 400×400
GVF Calculation	1.9s	$\mu = 0.2$, 80 iterations
2D Deformation	2.7s	$\alpha = \beta = 0.35$, $\kappa = 0.6$ 180 iterations $d_{\text{min}} = 2$, $d_{\text{max}} = 6$
Contour Matching	18.8s	100 nodes 40×40 Patch Size Epipolar Neighborhood = 5
Refinement	1.5s	$\alpha = \beta = 0.35$, $\kappa = 0.1$ 50 iterations
Total Time	27.5s	

Table 6.5: Runtime details for the textured skin case

	Calculated	Measured	Difference
Near the tail (A)	18.3mm	18.2mm	%0.5
Near the middle (B)	18.3mm	18.2mm	%0.5
Near the head (C)	18.5mm	18.2mm	%1.6

Table 6.6: Measurements performed at different parts of metal object on its width



Left shot (L) Right shot (R)

Figure 6.10: Left and right shots of metal object used in measurement

Chapter 7

Conclusion

We applied snakes (active contour models) on non-planar surfaces of 3D objects. We also used color information to improve segmentation. In addition, we used camera characteristics and stereo calibration data to perform physical measurements. Combining active contours with stereo imaging improved both techniques.

Segmentation was improved because of the extra information available from a pair of images instead of one. These two images, each taken from a slightly different angle, can compensate for each other in the areas where noise and specularity cause problems in segmentation.

Stereo matching was improved because of internal forces of the contour that try to maintain a smooth shape. When the matching process makes mistakes the internal energy of the contour increases and causes a resistance force which eliminates the error.

7.1 Contributions

Our research in this thesis covered a wide area in the subject of computer vision. Stereo vision, camera calibration and deformable models are each of interest in the computer vision field. However, our contributions in this work can be summarized into the following:

- *Design of an effective method for initializing a contour around a region on a 3D volume.* We proposed a technique for initializing the active contour without reconstructing the depth map for the whole image. Also the need for image rectification, which can be a time consuming process, was eliminated.

- *Extension of snakes (active contours) from planar surfaces to non-planar surfaces of 3D volumes.* We designed external forces to deform the active contour in 3D space. Different approaches were investigated and the most robust/effective one was proposed. We also investigated more sophisticated internal forces for 3D snakes (such as twisting forces) and other external forces (such as inflation).

- *More effective segmentation using active contours.* Using complementary information available from stereo images we reduced the effect of noise and specularity in segmentation.

- *Measurement in physical units.* In addition to segmenting regions of interest we provide real dimension of the area of interest. This information can also be used to calculate the area and the perimeter of regions. We also showed that stereo imaging can be used as a means of measuring lengths and dimensions with relatively high precision.

7.2 Future Work

This work can be extended in different ways and directions. We currently have the following ideas that can be explored:

- *Better matching in the 3D initialization phase.* The current system tries to match left and right contour nodes independently (node by node). Information about neighboring nodes' matches can be used to improve the accuracy of search for the corresponding node.

- *Better removal of irrelevant nodes.* Currently we use a simple method for removing irrelevant nodes after initialization. More sophisticated algorithms can be employed to find these nodes. Furthermore, instead of removing such nodes, we can try to adjust their position based on neighborhood nodes.

- *Better matching functions.* More advanced matching functions can be employed to find the corresponding points. Edge map information can be used to improve the quality of this function. Also adaptive window sizes as suggested by [8] can be used to increase the performance and accuracy of this process.

- *Applying different kinds of filters to improve edge detection and matching.* Various kinds of filters can be applied to the original images to increase the quality of edge detection and matching functions. Also different transformations of the original images may be used for different purposes. (edge detection or matching)

- *Runtime user interaction.* Currently we could put marks on the original images to guide the snakes (even though no such marks were necessary in either of the cases presented in Chapter 6). These marks can emphasize the edges that are not visible to the program. By marking the corresponding edges on both images, we can also help the matching process. However, more dynamic interactions can be made possible. It is also possible to give the user the ability to modify the contour in 3D and see the projections.

- *Color Correction.* In some case, left and right images have different color themes. For example one picture might be generally brighter than the other or more saturated or more reddish. This causes problems in matching process. It also may cause initial left and right contours to be completely different. For example in the case of medical images, one contour may capture a wider range than the other. Performing a color correction beforehand, may reduce this sort of difficulties.

This work can be used as a precise measuring tool. Measurements performed by our algorithm in cases with high contrast edges is comparable with a vernier caliper. We also showed that a calibrated stereo system with human interaction can be used as a tool for measuring lengths. This can be very helpful in cases where using tools like vernier calipers or micrometers is not practical. Impracticality can be either due to lack of access or for sanitary reasons in case of medical imaging.

Bibliography

[1] Jean-Yves Bouguet. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`, October 2004. Accessed Feb. 2005.

[2] D. C. Brown. Decentering distortion of lenses. *Photometric Engineering*, 37(3):444–462, 1966.

[3] Tat-Jen Cham and Roberto Cipolla. Stereo coupled active contours. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, pages 1094–1099. IEEE Computer Society, 1997. ISBN 0-8186-7822-4.

[4] Laurent D. Cohen. On active contour models and balloons. *CVGIP: Image Understanding*, 53(2):211–218, Mar. 1991.

[5] R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume 1. Interscience Publishers, Inc., 1953. ISBN 0-471-50447-5.

[6] Charles A. Hall and Thomas A. Porsching. *Numerical Analysis of Partial Differential Equations*. Prentice Hall, Englewood Cliffs, NJ, 1990. ISBN 013626557X.

[7] Janne Heikkilä and Olli Silvén. A four-step camera calibration procedure with implicit image correction. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 1106. IEEE Computer Society, 1997. ISBN 0-8186-7822-4.

[8] Takeo Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932, September 1994.

[9] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. In *International Journal of Computer Vision*, volume 1(4), pages 321–331, 1987.

[10] Danijela Markovic and Margrit Gelautz. Video object segmentation using stereo-derived depth maps. *27th Workshop of the AAPR/ÖAGM*, pages 197–204, 2003.

[11] D. Marr and T. Poggio. A computational theory of human stereo vision. In *Proc. Royal Society*, pages 301–328, May 1979.

[12] William K. Pratt. *Digital Image Processing.* John Wiley & Sons Inc., 1978. ISBN 0-471-01888-0. 514 pp.

[13] Guillermo Sapiro. Vector (self) snakes: a geometric framework for color, texture, and multiscale image segmentation. In *Proc. of International Conference on Image Processing*, volume 1, pages 817–820, 1996.

[14] J. Serra. Image analysis and mathematical morphology. *London: Academic*, 1982.

[15] Changming Sun. Fast stereo matching using rectangular subregioning and 3d maximum-surface techniques. *International Journal of Computer Vision*, 47(1–3):99–117, May 2002.

[16] Demetri Terzopoulos and Kurt Fleischer. Deformable models. *The Visual Computer*, 4:306–331, 1988.

[17] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214. ACM Press, 1987. ISBN 0-89791-227-6.

[18] Demetri Terzopoulos and Richard Szeliski. Tracking with kalman snakes. *Active Vision*, pages 3–20, 1992.

[19] Patrick H. Winston. *Artificial Intelligence, Second Edition.* Addison-Wesley Publishing Company, 1984. ISBN 0-201-08259-4. 351–355 pp.

[20] Chenyang Xu and Jerry L. Prince. Gradient vector flow: A new external force for snakes. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 66–71. IEEE Computer Society, 1997.

[21] Chenyang Xu and Jerry L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, pages 359–369, Mar. 1998.

[22] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *ICCV*, pages 666–673, 1999.

Index