UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of a doctoral Dissertation by

Michael Allen Greminger

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Bradley J. Nelson

Signature of Faculty Adviser

Date

GRADUATE SCHOOL

THE INTEGRATION OF MATERIAL MODELS AND COMPUTER VISION FOR FORCE AND DISPLACEMENT SENSING

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Michael Allen Greminger

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Bradley J. Nelson, Adviser

October 2005

Acknowledgments

This research was supported in part by the US National Science Foundation through grant numbers IIS-9996061 and IIS-0208564. I am supported by the Computational Science Graduate Fellowship (CSGF) from the US Department of Energy.

I would like to thank my faculty advisor Prof. Brad Nelson who introduced me to the idea of measuring forces from images and gave me the freedom to explore many different research directions. I would also like to thank Prof. Rajamani for the gracious use of his lab facilities and Prof. Chase for introducing me to the idea of using the BFGS algorithm for parameter estimation. I am grateful to my dissertation committee: Prof. Gini, Prof. Rajamani, and Prof. Chase.

Prof. Yu Sun provided me with robotic cell injection images and Dr. Ge Yang provided me with the use of the one degree of freedom gripper. Finally, I would like to thank Serdar Sezen for teaching me the microfabrication skills needed to fabricate the microgripper presented in this dissertation.

Abstract

The manipulation of deformable objects is an important problem in robotics and arises in many applications including biomanipulation, microassembly, and robotic surgery. In microrobotics and space robotics the robotic manipulator itself is often deformable. This dissertation discusses the use of computer vision to provide feedback for robotic interaction with deformable objects and to provide feedback when the robotic manipulator itself is deformable. Computer vision is a logical sensing choice for working with deformable objects because of its wide availability across many fields and the richness of the data provided by a vision system. A template based deformable object tracking algorithm will be introduced that can be used to provide force and displacement feedback for robotic applications. Various material models are used for modeling the template deformation including the beam equation, the boundary element method, and neural network models. It is important that the tracking algorithm used for feedback is robust to occlusions and spurious edges in the source image. Approaches to handle these potential difficulties are presented. Finally, a compliant four degree of freedom MEMS microgripper is presented. Because of this gripper's compliant design, vision tracking can be used to provide position and force feedback which simplifies its design and fabrication. Vision-based force and displacement sensing is capable of providing accurate and robust feedback where other sensing techniques are not possible or are difficult to implement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Robotics and Deformable Objects

Robotic manipulation of rigid objects is well established. Application areas where robots interact with rigid objects include automated assembly, welding and painting robots, and mobile robots. Additionaly, the robots themselves are designed to be as rigid as possible. One primary reason for robots and the objects they manipulate being rigid is that rigid objects are easier to model than deformable objects. Another reason is that rigid objects can be characterized by fewer degrees of freedom than deformable objects. The increased number of degrees of freedom inherent in deformable structures require more sensors to measure the state of the system. Robots are often able to interact with rigid objects in an open loop fashion but this is not possible for high dimensionality deformable systems.

There are many compelling applications where robots need to interact with deformable objects or where the robot itself is required to be deformable. The richest field is medical robotics where robots must interact with the deformable tissues of the body. There are also situations where the robot itself must be deformable. One example where deformable robots are used is in space robotics where the robot is designed to be compliant in order to conserve weight. Another example where robots are designed to be compliant is in the MEMS domain where it is common to use compliant mechanisms.

1.1.1 Applications Where Robots Must Interact With Deformable Objects

Medical robots are becoming increasingly important for performing tasks that human operators can not perform or to perform current operations in a less invasive manner. One of the first areas where medical robots were used was in the area of orthopedics for tasks such as knee replacements and hip replacements [20] [48]. These applications involve robots interacting with rigid bones so the established rigid body robotics techniques can be used directly. Later on, robotic surgery techniques were applied to other surgical areas. One area where robotic surgery has become more commonplace is in the area of cardiac surgery. Teleoperated surgery systems such as the ZEUS Surgical System from Computer Motion Inc. or the da Vinci Surgical System from Intuitive Surgical Inc. have allowed cardiac surgeons to perform minimumly invasive cardiac operations that would have previously required open heart surgery [30] [28].

When manipulating deformable objects, more feedback data is required to completely define the state of the system. The current teleoperated surgery systems increase the feedback information available by providing haptic force feedback on the ZEUS system and stereo visual feedback, in addition to haptic feedback, on the da Vinci system. There has also been work on performing surgery on a beating heart. Nakamura et al. [34] developed a system to visually track a heart in order to subtract the heart motion from the motion of the robot allowing the surgeon to perform the operation as if the heart is stationary. Future advances in robotic surgery will require more sophisticated feedback for the state of deformable objects.

Another area where robots interact with deformable objects is in the area of biomanipulation. A common biomanipulation task is cell injection. Sun et al. [45] developed a robotic system to autonomously inject mouse egg cells with genetic material for cancer research. As can be seen from figure 1.1, the cell deforms as the injection pipette enters the cell. Measuring the deformation of the cell is important for two reasons: the cell can be damaged if the deformation is too large and the displacements can be used in conjunction with a force sensor to measure the material property parameters of the cell.

The automated assembly of 1D deformable objects including ropes, cables and tubes also requires feedback about the state of the deformable object. The high number of degrees of freedom that it requires to model such objects makes open loop manipulation very challenging. Efforts to automatically manipulate such objects include the work of Acker et

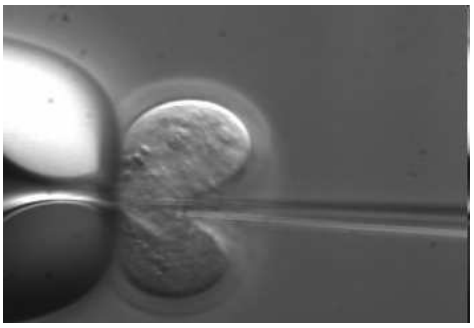


Figure 1.1: A mouse oocyte cell being robotically injected with DNA material.

al. [1] and Luo et al. [31].

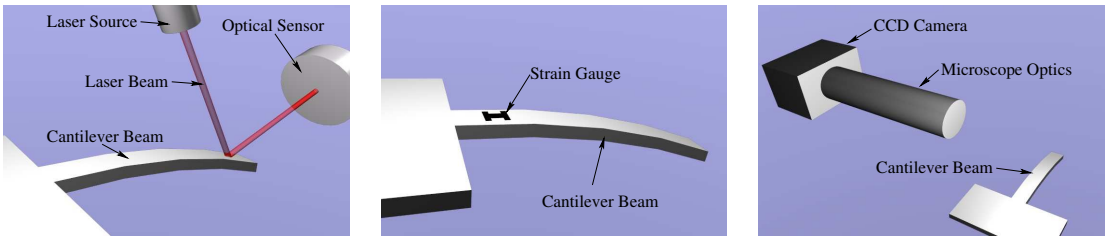
When manipulating deformable objects, feedback information is required to determine the state of the object. Feedback is also an important mechanism for preventing damage to the deformable object being manipulated. In order to be able to predict when the damage will occur, some knowledge of the material model is needed so that the deformations of the object being manipulated can be related to stresses and forces.

1.1.2 Applications Where the Manipulator Itself is Deformable

Traditionaly robotic manipulators have been designed to be rigid. The advantages of this approach are that rigid body kinematic models can be used to accurately model the robot and low frequency resonance modes of the robotic manipulator are avoided which simplifies the control of the manipulator. However, there are situations where a deformable manipulator cannot be avoided and the deformation of the manipulator must be considered in order to control it properly.

Space structures are often not rigid because of the severe weight restrictions that space structures are subject to [6]. Because of this, the deformation of the space structure must be accounted for during any manipulation task to prevent unwanted vibrations. Currently Lichter et al. [29] are working on using range images as a feedback mechanism to control deformable space structures.

At the microdomain, manipulators are often deformable in nature because compliant joints are often preferable to pin joints. One benefit of compliant joints is that they provide backlash free motion. Backlash free motion allows for very high precision positioning. Another benefit of compliant joints is that they are simpler to fabricate using MEMS fab-



(a) (b) (c)

Figure 1.2: Three different feedback mechanisms for monitoring the deflection of a cantilever beam: (a) laser feedback, (b) strain gauge feedback, and (c) vision feedback.

rication techniques than pin joints. The motion of compliant mechanisms is more complicated to model than that of rigid link mechanisms and requires using material models to completely define the motion.

1.2 Robotic Sensors for the Determination of State

Feedback for rigid link mechanisms generally involves linear and rotary position sensors that mirror the prismatic and rotary degrees of freedom of the robotic manipulator respectively. For rigid link mechanisms, these sensors can completely define the state of the manipulator.

For systems with compliance, different techniques need to be employed to measure system state. A common sensing technique is to use strain gauges to capture the deformation of the system. When strain gauges can be employed, they provide good feedback information for a deformable system. Another common approach is to measure the state of deformable object using an optical approach where a laser is reflected off of a deformable object to measure deformations. Figures 1.2(a)-(b) illustrate these two approaches to the measurement of deformation.

There are situations where strain gauges or laser based sensing techniques cannot be used. One situation occurs when the object that is being manipulated is immersed in a fluid where the biomanipulation application is an example. In these situations another feedback approach must be used. The approach that is presented in this dissertation uses vision feedback to measure the state of a deformable object. The next section will overview the use of computer vision as a feedback mechanism for deformable objects.

1.3 Using Computer Vision to Provide Feedback for Deformable Objects

This dissertation presents a new approach for providing feedback for deformable objects. A camera image of the deformed object is processed in order to determine its current shape (see Figure 1.2(c)). From this shape information it will be shown that forces can also be measured. This approach to feedback for deformable objects is non-contact allowing it to be used when it is not possible to use contact methods such as strain gauges. The vision based approach provides feedback for high degree of freedom deformable objects because of the high dimensional nature of images. For example, it will be shown in Chapter 7 that the complete stress field of a 2D linearly elastic object can be obtained from the contour of an object. This entire contour can easily be captured by an image but is challenging to capture by other methods.

For microrobotic and biomanipulation applications there is usually a camera and microscope included as part of the system to provide visual feedback for the operator. Using the approach presented here, the images from this camera or microscope can also be processed by computer to provide displacement and force feedback.

Computer vision provides the high dimensionality information that is needed to track deformable objects with many degrees of freedom. It will be shown in this dissertation that computer vision can be used to perform high precision force and displacement measurements for micromanipulation. It will also be shown that these measurements can be performed in a robust manner that is insensitive to spurious edges in the image or to occlusion of portions of the scene.

1.4 Previous Work in Vision-Based Deformable Object Tracking

The use of elastic models is well established in computer vision. There are two classes of methods that are commonly used. One class provides good tracking results for a wide variety of objects but uses deformation models that are not based on the deformation of physical solids. This class includes the popular active contour model methods. The second class of tracking algorithms uses physics based models to track a smaller class of objects more robustly and to a higher degree of accuracy. The method presented in this dissertation

builds upon the methods in this second class.

1.4.1 Active Countour Model Based Methods

In 1987 Kass et al. [25] proposed a method to track contours in an image using a 2D elastic model called an active contour model or a snake. The snake consists of a 2D spline which has elastic properties and is attracted to edge features within the image. The spline is matched to the image by the minimization of an error function that has terms for internal energy, image energy, and constraint energy. The error function as defined by Kass et al. is

$$E_{\text{snake}} = E_{\text{internal}} + E_{\text{image}} + E_{\text{constraint}} \tag{1.1}$$

The internal energy is a measure of the amount of stretching and bending of the spline, the image energy is a measure of the image intensities and gradient values that the spline passes through, and the constraint energy allows the user to guide the spline externally.

There are many methods that build upon the active countour model framework. Yuille et al. [56] used a deformable template matching algorithm to track facial features. Their splines were defined with degrees of freedom that allowed the splines to take the shape of facial features.

The active contour family of tracking algorithms are efficient at image segmentation and tracking general objects. These methods, however, are not the most robust for less general scenes where there is a priori knowledge about the objects being tracked. For these situations it is more efficient to use a model-based approach as described in the next section.

1.4.2 Model-Based Methods

Metaxas [33] introduced the use of 3D meshes with physics-based elastic properties to track both rigid and non-rigid objects. The primary application of his physics based approach was to track tissues in medical images. The use of physics based models allows for more robust tracking results than can be obtained by using a general active contour model based approach. Metaxas' techniques were used purely for tracking deformable objects and were not used to measure forces or material properties from images.

Prior work also exists in which elastic models are used to derive force or material property information from images. Tsap et al. [50] proposed a method to use nonlinear finite element modeling (FEM) to track non-rigid objects in order to detect the differences in

Table 1.1: Objectives.

Objectives
• Measure forces accurately from images • Perform deformable object tracking that is robust to image noise and occlusion • Design the tracking algorithm to be modular • Design the tracking algorithm to be computationally efficient

elasticity between normal and abnormal skin. They also discussed how their method could be used for force recovery. Kaneko et al. [24] presented a tactile sensor that was able to measure forces visually however their algorithm was limited to wire shaped objects. There has also been work in force measurements at micro and nano-scales using computer vision. Wang et al. [51] used Finite Element Modeling (FEM) techniques to derive the forces that are applied to deformable microparts. Their method is limited by the need to track each FEM mesh point in the image. Danuser et al. [8] proposed the use of statistical techniques along with deformable templates to track very small displacements. They applied their technique to the measurement of strain in a microbar under tension. Dong et al. [11] monitored the tip deflection of an AFM cantilever beam in order to obtain the material properties of multi-walled carbon nanotubes. The force measurement algorithm presented here is unique in using contour data alone, therefore, no feature tracking is required, and the method can be generalized to elastic objects with general geometries.

1.5 Objectives of this Work

The major objectives of this work are listed in Table 1.1. The primary objective is to demonstrate that forces can be measured by tracking structural deformations in images. In order for vision-based force measurement to be a substitute for traditional force sensing technologies, it needs to be shown that forces can be measured to a high level of precision. The algorithm will be validated using a traditional force sensor to insure the accuracy of the vision-based approach.

With this work, an additional goal is to increase the robustness of general deformable object tracking. This will allow the algorithm to be used as a feedback mechanism for the manipulation of deformable objects where occlusions or image noise may arise without

warning. The increase in robustness will come from improvements in the material models, the solver, and the computer vision algorithm.

Algorithms such as active contours tie the model, the solver, and the computer vision algorithm together. The aim of this project is to provide a tracking method that separates the material model, the solver, and the low level image processing. This modularity allows the material model or the image processing algorithm to be swapped out easily, allowing one to use the overall scheme that is best suited to the problem at hand. As an example, an edge detection algorithm that is much more robust to spurious edges than the Canny edge operator will be presented in this dissertation. However, this new edge detection algorithm is more computationaly expensive than the Canny operator so it should only be used when it is absolutely needed. The modular design of the overall algorithm allows this switch to be performed without affecting the other modules of the algorithm.

A final goal is to create an algorithm that is computationally efficient. The goal is to use these measurements as feedback for micromanipulation so they need to be able to be executed in real time. The modularity of the algorithm as addressed above makes the optimization task more straight forward because each of the modules can be optimized separately outside the framework of the overall algorithm.

1.6 Dissertation Organization

Chapter one provides an overview of the role of deformable objects in robotics and the current state of the art in vision-based deformable object tracking. A broad overview of the deformable object tracking algorithm that is presented in this dissertation is described in Chapter 2. The next three chapters go into the details of each of the major modules of the algorithm which are the low level vision algorithm presented in Chapter 3, the error minimization algorithm in Chapter 4, and the boundary element method deformation model in Chapter 5. The tracking algorithm is modified for robustness to spurious edges and occlusions in Chapter 6. Chapter 7 presents the method for vision-based force measurement. An artificial neural network is used to model deformations in Chapter 8 where the neural network is trained directly from images of a deformed object. Chapter 9 introduces a new compliant microgripper with four degrees of freedom. This Chapter also introduces a new bi-directional thermal actuator design. Finally, Chapter 10 summarizes the dissertation. This final chapter also lists the research contributions of this dissertation.

Chapter 2

Overview of the Deformable Object Tracking Algorithm

The applications described in this dissertation, at their most basic level, all involve the measurement of displacements from images. There are three major approaches that are typically used to make these measurements: feature tracking, low level pixel analysis, and template matching.

If a geometric model for the object is available, template matching performs well. In template matching, a representation of the object, the template, is compared to the image. The template has a fixed number of degrees of freedom which usually include rotation and translation. For the applications in this dissertation, the template will also have deformation degrees of freedom. The goal of template matching is to find the set of parameters that minimize the error between the template and the image. The characteristics that define a template matching algorithm are the error measure used, the definition of the degrees of freedom, and the numerical minimization algorithm used. The algorithm presented in this dissertation separates these components into distinct modules. Figure 2.1 lists these modules. These modules will be discussed in the remainder of this chapter.

2.1 Measuring the Error Between the Template and the Image

It was mentioned that an error measure is minimized between the template and the image. The error measure will now be defined. The image will first be preprocessed to reduce the

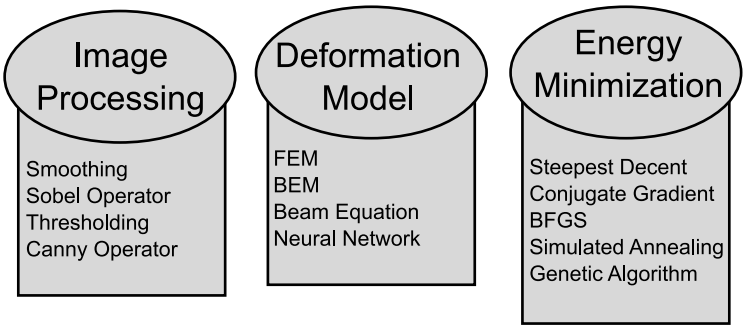


Figure 2.1: The three modules of the deformable object tracking algorithm presented in this dissertation.

image data to a manageable level and then the error will be measured using a least squares error measure.

2.1.1 Image Preprocessing

Images contain a large amount of data which makes it desirable to reduce the amount of data without losing the information that is to be extracted from the image. The process of decreasing the amount of data in an image without losing the essential information is called image preprocessing. Common preprocessing algorithms include segmentation, despeckling, and edge detection. Edge detection will be used here to eliminate unneeded data while preserving the edge displacements which contain most of the information in the image.

There are multiple edge detection algorithms available. The Canny edge detection algorithm [5] will be used. The Canny edge detector takes a gray-scale image as input and outputs a binary edge image with each edge represented as a white pixel and all other pixels set to black. The Canny edge operator is chosen because of its ability to preserve continuous edges within an image. Figure 2.2 shows the results of the Canny edge operator applied to an image. Edge detection will be discussed in detail in the next chapter.

2.1.2 The Least Squares Error Measure

The deformable template is registered to a binary edge image using a least square error measure. The template is represented by a list of 2D vertices $\mathbf{r}_i$ and the edge pixels in the current image are represented by the list of 2D vertices $\mathbf{w}_i$. The registration algorithm minimizes the distance squared between the transformed template vertices $\mathbf{r}'_i$ and the nearest

Figure 2.2: Original image on left and Canny edge image on right.

image edge vertices $\mathbf{w}_i$ where the template vertices are transformed by a template transformation T with degrees of freedom represented by the vector $\mathbf{x}$.

$$\mathbf{r}'_i = T(\mathbf{r}_i, \mathbf{x}) \tag{2.1}$$

The error between the transformed template vertices $\mathbf{r}'_i$ and the image vertices $\mathbf{w}_i$ is defined by the following function

$$E(\mathbf{x}) = \sum_{i=1}^{M} \|\mathbf{r}'_i - \mathbf{w}_i\|^2 \tag{2.2}$$

where $\mathbf{r}'_i$ is the position vector of the ith edge pixel of the template transformed by (2.1); $\mathbf{w}_i$ is the position vector of the edge pixel in the image that is closest to the point $\mathbf{r}'_i$; and M is the number of edge pixels in the template. This error function sums the square of the distance between each template vertex and the nearest image edge pixel. Figure 2.3 shows an example of calculating this error with a simple template and a simple edge image. Since the transformed template vertices $\mathbf{r}'_i$ are transformed by the template transform T, E will be a function of the transformation's degrees of freedom $\mathbf{x}$. By minimizing E, the values of $\mathbf{x}$ that best match in the image in a least squares sense will be found.

2.2 The Template Degrees of Freedom

Now the degrees of freedom of the template will be defined. The degrees of freedom will consist of a rigid body motion contribution and a deformable body contribution. The rigid

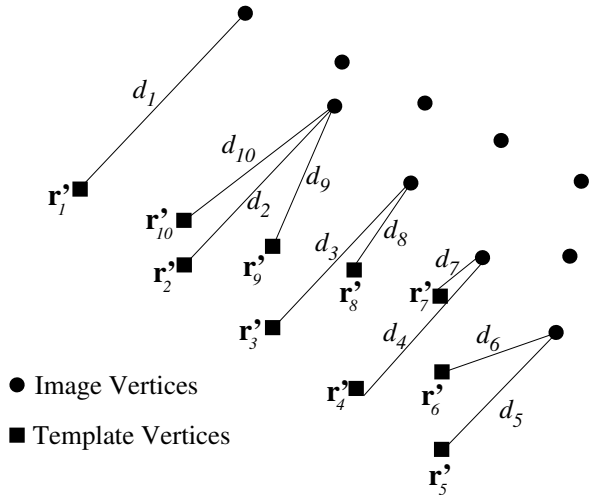


Figure 2.3: Example of measuring error between a template and an image where d_i is the distance from the template vertex $\mathbf{r}'_i$ to the nearest image vertex. The error in this example is $E = \sum_{i=1}^{10} d_i^2$.

body component will be an affine transformation and the deformable body component will be based on elasticity theory.

2.2.1 The Rigid Body Motion Degrees of Freedom

The rigid body portion of the transformation is simply an affine transform. For the rigid body case the template transform is defined as

$$\mathbf{r}' = T(\mathbf{r}, \theta, \mathbf{X}) = A(r) = \mathbf{X} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \mathbf{r} \tag{2.3}$$

where θ is the angle of rotation and $\mathbf{X}$ is the translation vector. In this dissertation, the affine transform is occasionaly referred to as $A(r)$. Figure 2.4 shows how these parameters are defined. The error function between the transformed template vertices $\mathbf{r}'_i$ and the image vertices $\mathbf{w}_i$ can be written as

$$E(\theta, \mathbf{X}) = \sum_{i=1}^{M} \|\mathbf{r}'_i - \mathbf{w}_i\|^2 \tag{2.4}$$

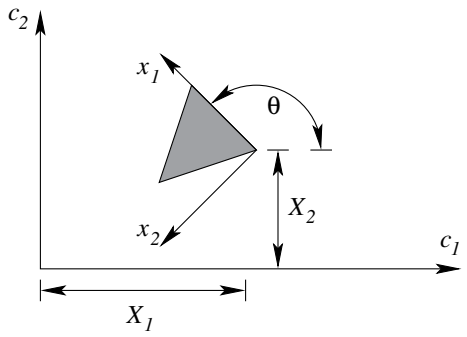


Figure 2.4: Rigid body template degrees of freedom.

Table 2.1: Various methods for modelling the deformations of materials.

Deformable Material Modelling Techniques
• Analytical solutions such as the beam equation
• Boundary element model (BEM)
• Finite element model (FEM)
• Neural network model

2.2.2 The Deformation Degrees of Freedom

Since all of the applications presented in this dissertation involve deformable objects, a template with only rigid body motion degrees of freedom is not sufficient. It is necessary to add non-rigid motion degrees of freedom to the template. Various methods will be used to model this deformation throughout this dissertation, but in all cases the degrees of freedom will be forces applied to the boundary of the object. The deformation model will return the displacement of each point within the object given these applied forces. Table 2.1 lists several available methods to model deformable objects. Analytical, boundary element, and neural network models will be presented in this dissertation.

Deformations will be expressed in general by the following equation

$$\mathbf{u} = D(\mathbf{r}, \{t\}) \tag{2.5}$$

where D is the deformation model and $\{t\}$ is the traction distribution applied to the object. Using the above equation, the template transformation T can be redefined as

$$\mathbf{r}' = T(\mathbf{r}, \theta, \mathbf{X}, \{t\}) = \mathbf{X} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} (\mathbf{r} + \mathbf{u}) \tag{2.6}$$

where each template vertex $\mathbf{r}$ is translated by the vector $\mathbf{u}$ obtained using (2.5) before applying the affine transformation. Since u is a function of $\{t\}$, the new template transformation T is now a function of the applied traction $\{t\}$ in addition to the affine transformation parameters, θ and $\mathbf{X}$. The resulting error function is

$$E(\theta, \mathbf{X}, \{t\}) = \sum_{i=1}^{M} \|\mathbf{r}'_i - \mathbf{w}_i\|^2 \tag{2.7}$$

2.3 Minimizing the Error Function

The error function (2.4) is minimized by a gradient-based multi-variable minimization technique called the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [13]. The BFGS method is a gradient based minimization technique, and differs from the steepest decent method in that it uses information from previous iterations in the choice of a new search direction giving it faster convergence rates. BFGS uses information from previous iterations to approximate the Hessian matrix giving it convergence rates similar to second-order minimization techniques without the overhead of computing a second derivative.

2.4 Extending the Template Matching Algorithm to Three Dimensions

The template matching algorithm described above was defined for two dimensional objects. For example, (2.3) defines a two dimensional affine transformation. For some applications, tracking in three dimensions is required. For these situations, the template matching algorithm can be easily extended to three dimensions. The affine transformation (2.3) needs to be changed to a three dimensional affine transformation with one additional translation degree of freedom and two additional rotational degrees of freedom. The deformation function (2.5) also needs to be updated to a model that supports three dimensions.

Once the three dimensional template is defined, it is necessary to obtain the three dimensional image data. There are two ways to obtain this data. The first is to use multiple camera angles. The second method is to use an imaging technology that is able to obtain

three dimensional data directly such as laser range finders. Medical imaging techniques such as magnetic resonance imaging (MRI) often provide three dimensional information directly as stacks of images.

Chapter 3

Low Level Image Processing

All computer vision techniques at some point need to manipulate individual image pixels. This chapter discusses the low level image processing techniques that are used for the approach presented in this dissertation. The techniques presented in this chapter are widely used in the field of computer vision. Chapter 6 introduces a new low level image processing algorithm that builds upon the algorithms presented here.

3.1 Image Convolution

Convolution is frequently used to filter image data. To perform convolution, a convolution kernel needs to be defined. The convolution kernel is a two dimensional array that generally has a size that is much smaller than that of the image. The equation for image convolution is

$$G(i,j) = I(i,j) \odot K(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i+k-1, j+l-1) K(k,l) \tag{3.1}$$

where I is the source image of size MxN and K is the convolution kernel of size mxn. Table 3.1 shows a symbolic representation of an image and Table 3.2 shows a symbolic representation of a convolution kernel.

There are many convolution kernels that are commonly employed in computer vision. There are kernels that perform image smoothing (low pass filters) and those that enhance edges (high pass filters). An example of a smoothing kernel is the mean kernel which computes the mean value of the local image pixels (see Table 3.3 for an example of a 3x3 mean kernel).

Table 3.1: Image pixels.

I_{11}	I_{12}	I_{13}	$\cdots$	I_{1N}
I_{21}	I_{22}	I_{23}		
I_{31}	I_{32}	I_{33}		
$\vdots$			$\ddots$	
I_{M1}				I_{MN}

Table 3.2: General convolution kernel.

K_{11}	K_{12}	K_{13}	$\cdots$	K_{1n}
K_{21}	K_{22}	K_{23}		
K_{31}	K_{32}	K_{33}		
$\vdots$			$\ddots$	
K_{m1}				K_{mn}

Table 3.3: Mean kernel.

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Table 3.4: Gaussian blur kernel.

0.60653	0.77880	0.60653
0.77880	1.00000	0.77880
0.60653	0.77880	0.60653

Figure 3.1: Original image on left and Gaussian smoothed image on right.

A more commonly used kernel for smoothing an image is the kernel which obtains its values from the shape of the Gaussian function. The two-dimensional zero-mean discrete Gaussian function is [23]

$$G(i,j) = e^{-\frac{i^2+j^2}{2\sigma^2}} \tag{3.2}$$

where σ determines the width of the Gaussian. For a 3x3 kernel with $\sigma^2 = 2$, the Gaussian kernel is given in Table 3.4. An image smoothed by this kernel is shown in Figure 3.1.

The next section introduces the high pass filter Sobel kernels that are used to enhance the edges in the source image.

3.2 Locating Edges in Images

Edges in images correspond to areas in the image where the intensity changes rapidly. All edge detection algorithms apply a high pass filter to the image to enhance edges. Edge detection is frequently used as the first step in image processing because it greatly reduces the amount of data storage required for an image while preserving most of the information available. For the task of deformable object tracking, edges provide all of the information that is necessary. It will be shown in the chaptor on vision-based force measurement that

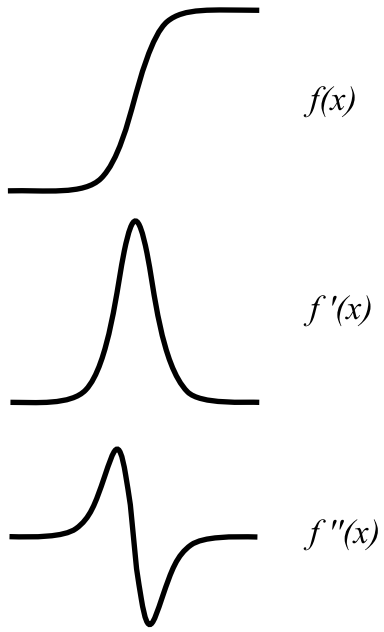


Figure 3.2: The top curve is the functional representation of a one dimensional edge. The bottom two curves show the first and second derivatives of that function.

for linearly elastic objects, the edge of the object completely defines the elastic state of the object.

Figure 3.2 shows a one dimensional function $f(x)$ with an intensity change that represents an edge. The first and second derivatives of this function are also shown in this figure. The first derivative can be used to locate edges by searching for the local maxima and the second derivative can be used to locate edges by locating zero crossings. In practice, the first derivative is generally used because the second derivative has a higher noise level in the resulting edge image when compared to the edge image obtained using the first derivative.

Since images represent discrete samples from a two dimensional function, the image gradient will be used as a measure of edge intensity. Specifically, the magnitude of the image gradient will be used which can be computed directly using

$$|\nabla I(x,y)| = \sqrt{\frac{\partial I(x,y)}{\partial x}^2 + \frac{\partial I(x,y)}{\partial y}^2} \tag{3.3}$$

In practice, the image gradient is calculated using a convolution kernel. A common kernel

Table 3.5: x Sobel kernel.

-1	0	+1
-2	0	+2
-1	0	+1

Table 3.6: y Sobel kernel.

+1	+2	+1
0	0	0
-1	-2	-1

for edge detection is the Sobel kernel which calculates the first derivative in the x and y directions using separate kernels. These kernels are shown in Tables 3.5 - 3.6 respectively [23]. Figures 3.3(b) and (c) show the results of applying the x and y sobel kernels to the source image shown in Figure 3.3(a) (note that the absolute value of the convolution result $G(i,j)$ is shown in the figures).

The magnitude of gradient is calculated from the results of the Sobel operator using

$$|G(i,j)_{\text{Sobel}}| = \sqrt{G_x(i,j)^2 + G_y(i,j)^2} \tag{3.4}$$

where $G_x(i,j)$ and $G_y(i,j)$ represent the results of the x and y Sobel convolutions respectively. The square root calculation is usually removed in order to make the calculation more efficient. The resulting approximate to the magnitude of gradient is

$$|G(i,j)_{\text{Sobel}}| = |G_x(i,j)| + |G_y(i,j)| \tag{3.5}$$

The angle of the edge can be calculating using:

$$\theta(i,j) = tan^{-1}\left(\frac{G_y(i,j)}{G_x(i,j)}\right) \tag{3.6}$$

A thresholding technique is used to detect edges based on the approximation of the magnitude of gradient (3.5). If the magnitude of gradient is greater then a threshold value than the pixel is set to an edge. Figure 3.3(d) shows the final binary edge image obtained by thresholding the magnitude of the gradient obtained using the Sobel convolution kernels.

(a) (b) (c) (d) (e)

Figure 3.3: (a) Original image, (b) Sobel operator in x direction, (c) Sobel operator in y direction, (d) threshold based Sobel edge detector, and (e) Canny edge detector.

3.3 The Canny Edge Operator

The Canny edge operator is an improvement upon the threshold-based Sobel edge detection method and was designed with the goal of obtaining an optimal edge detector. The Canny edge operator is a multi-step algorithm with one of the steps including the use of the Sobel operator to calculated the magnitude of the image gradient. Because of its increased complexity, the Canny edge operator does take more computation time than the Sobel edge operator. However, this complexity is justified by the results obtained and modern computer hardware can compute the Canny edge operator very quickly. The Canny edge operator's unique features include its ability to preserve continuous edges through the use of dual threshold values and the thinning of edges to a thickness of one pixel by only preserving the local maximum edge through the use of the edge angle information. The steps to perform the Canny edge detection are [23]:

1. Smooth the image using a Gaussian convolution kernel.
2. Calculate the magnitude of the image gradient and the edge normal direction using (3.5) and (3.6) respectively.
3. Suppress the nonmaxima gradient magnitude values.
4. Link continuous edges by using two threshold values.

The first step of the Canny edge operator is to blur the image slightly by convolving the image with a Gaussion kernel. This step is done to minimize the noise that is amplified by the application of the Sobel edge operators used to calculate the image gradient. The next step is to apply the Sobel operator in both the x and y directions using the Sobel kernels presented in the previous section. Equation (3.5) is used to approximate the magnitude of the gradient and (3.6) is used to calculate the angle of the edge.

Each edge pixel is assigned to one of four sectors based on the edge angle. The four sectors are shown in Figure 3.4. The edge candidate is assigned to the sector to which its edge normal lies. These sectors are used to perform the non-maxima suppression which eliminates edge pixels that are not the maximum for the edge in question. Each pixel magnitude of gradient is compared to each of its neighbors in the same sector. The pixel that is the maximum is kept and the magnitude of the others is set to zero. The nonmaxima suppression step helps to insure that edges are only one pixel wide.

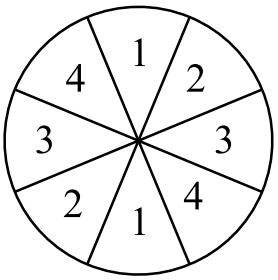


Figure 3.4: Diagram representing the four possible Canny edge directions.

Finally, edge thresholding by a hysteresis technique is preformed. Unlike the Sobel edge operator, the Canny edge operator uses two threshold values instead of one. If the magnitude of gradient of an edge is greater than the larger of the two thresholds, the edge is automatically accepted. If the magnitude of gradient is below the lower threshold, the edge is automatically rejected. Edges between the two threshold values are accepted if they form part of a continuous chain of edges and rejected otherwise. This hysteresis technique helps the Canny edge operator preserve continuous edges in images even if the edge is weak at points.

Figures 3.3(d)-(e) compare the Sobel and Canny edge operators for the same source image. It can be seen that the Canny operator not only reduces the noise level in the resultant edge image, but it also helps to prevent breaks from forming in continuous edges. The local nonmaxima filtering also helps insure that the Canny edges are only one pixel wide. Compare this characteristic to the Sobel edges which are often many pixels wide. The canny edge operator will form the starting point for all of the image processing algorithms presented in this dissertation.

Chapter 4

Minimization of the Error Function

After the edge detection algorithm and the appropriate deformation model are implemented the final module to implement is the module to minimize the error function. The error function module is used to minimize (2.7). This chapter will present the algorithm used to minimize this error function.

The goal of the minimization algorithm is to find the $\mathbf{x}_{\min}$ that minimizes the error function

$$E(\mathbf{x}_i) = \sum_{j=1}^{M} \left\| \mathbf{r}'_j - \mathbf{w}_j \right\|^2 \tag{4.1}$$

where

$$\mathbf{x}_i \equiv \begin{bmatrix} \theta_i \\ \mathbf{X}_i \\ \{t\}_i \end{bmatrix} \tag{4.2}$$

and $\mathbf{x}_0$ is the initial guess for the solution of the minimization problem.

4.1 Approaches to Function Minimization

There are numerous approaches to the numerical minimization of functions. In general, if the error function to be minimized has a first derivative, the gradient descent based methods have the best convergence performance. For gradient based methods, the solution $\mathbf{x}$ is incremented by

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha \mathbf{S}_i \tag{4.3}$$

where $\mathbf{S}_i$ is the search direction for the ith iteration and α is a scalar indicating the distance to increment in the direction $\mathbf{S}_i$. The value is α is determined using a line search algorithm which will be discussed below. For the steepest decent algorithm $\mathbf{S}_i$ is given by

$$\mathbf{S}_i = - \bigtriangledown E(\mathbf{x}_i) \tag{4.4}$$

The value for α is usually computed using a one-dimensional search algorithm such as Newton's method.

4.2 The BFGS Minimization Algorithm

The BFGS method is a variable metric method and as such it uses information from previous information when it updates the search direction $\mathbf{S}_i$. Because of this, variable metric methods converge faster then a simple steepest descent search.

4.2.1 Calculation of Search Direction

The BFGS method uses the following equation to obtain the search direction for each iteration [13]

$$\mathbf{S}_i = -\mathbf{H_i} \bigtriangledown E(\mathbf{x}_i) \tag{4.5}$$

where $\mathbf{H}_i$ is calculated by the following equation

$$\mathbf{H}_i = \mathbf{H}_{i-1} + \mathbf{D}_i \tag{4.6}$$

$\mathbf{D}_i$ is called the symmetric update matrix and it is defined by the following equation

$$\mathbf{D}_i = \frac{\sigma + \tau}{\sigma^2} \mathbf{p}\mathbf{p}^{\mathrm{T}} - \frac{1}{\sigma} \left[\mathbf{H}_{i-1}\mathbf{y}p^{\mathrm{T}} + p\left(\mathbf{H}_{i-1}\mathbf{y}\right)^{\mathrm{T}} \right] \tag{4.7}$$

where $\mathbf{p}$, $\mathbf{y}$, σ, and τ are calculated by the following equations

$$\mathbf{p} = \mathbf{x}_i - \mathbf{x}_{i-1} \tag{4.8}$$

$$\mathbf{y} = \bigtriangledown E(\mathbf{x}_i) - \bigtriangledown E(\mathbf{x}_{i-1}) \tag{4.9}$$

$$\sigma = \mathbf{p} \cdot \mathbf{y} \tag{4.10}$$

$$\tau = \mathbf{y}^{\mathrm{T}} \mathbf{H}_{i-1} \mathbf{y} \tag{4.11}$$

$\mathbf{H}_i$ is an approximation to the inverse Hessian matrix which allows the BFGS method to have convergence properties very similar to second order minimization methods. Methods that approximate the Hessian matrix are called quasi-Newton methods.

4.2.2 The Line Search

A line search algorithm is used to determine the α value used in (4.3). The choice of line search algorithm is very important to the performance of the gradient search method. There are many possible line searches that can be used including Newton's method and the golden section method. When using one of these methods, α is often chosen so that $\mathbf{x}_{i+1}$ in (4.3) gives the minimum error function value. When using quasi-Newton methods this may not be the best approach because it may actually slow down the convergence of the algorithm and require an unnecessary number of error function evaluations in order to perform the line search. For this reason, the backtracking algorithm is used to solve for α [10]. The back-tracking line search algorithm initially attempts $\alpha = 1$ and uses $\alpha = 1$ if the error function is decreased sufficiently. If the error function is not decreased sufficiently, α is decreased or "backtracked" until a value of α is found that sufficiently decreases the error function. Quadratic interpolation of the error function along the search vector is used for the first backtrack and cubic interpolation is used for subsequent backtracking. The use of the backtracking line search algorithm helps to dramatically reduce the number of error function evaluations.

4.3 Calculating the Analytical Gradient of the Error Function

In order to execute the BFGS method, the gradient of the error function must be calculated in order to obtain the new search direction for each iteration (see (4.5)). The gradient can be calculated numerically or explicitly. This section will point out the drawbacks of calculating the gradient numerically and will show how to compute the explicit gradient in such a way that allows the deformation model or the error measure to be changed without having to rewrite the entire explicit gradient formula.

The gradient of E is defined as

$$\bigtriangledown E(\mathbf{x}) = \begin{bmatrix} \frac{\partial E}{\partial \mathbf{x}(1)} \\ \frac{\partial E}{\partial \mathbf{x}(2)} \\ \vdots \\ \frac{\partial E}{\partial \mathbf{x}(n)} \end{bmatrix} \tag{4.12}$$

where E is a function of n variables.

4.3.1 Numerical Calculation of Gradient

As can be seen from (4.12), the calculation of the gradient of E requires the computation of each of the partial derivatives. Each of these partial derivatives can be approximation by the following symmetrical finite difference equation

$$\frac{\partial E}{\partial \mathbf{x}(i)} = \left(E\left(\begin{bmatrix} \mathbf{x}(1) \\ \vdots \\ \mathbf{x}(i)+h \\ \vdots \\ \mathbf{x}(n) \end{bmatrix}\right) - E\left(\begin{bmatrix} \mathbf{x}(1) \\ \vdots \\ \mathbf{x}(i)-h \\ \vdots \\ \mathbf{x}(n) \end{bmatrix}\right) \right) / (2h) \tag{4.13}$$

where h is chosen to be small. The optimal value of h depends on the machine precision of the variables (see [37] for guidelines for choosing an optimal value for h). Note that the calculation of the gradient will require $2n$ evaluations of the function E since two function evaluations are required for each of the n partial derivatives.

There are two major drawbacks to this approach of computing the error function gradi-

ent. The first drawback is that this calculation is an approximation to the gradient. Because of this, it will affect the convergence characteristics of the BFGS algorithm. The second drawback, and the more serious drawback for the deformable object tracking application, is that the approximation to the gradient requires $2n$ evaluations of the error function. This large number of functional evaluations will limit the achievable performance of the deformable object tracking algorithm especially for templates with a large number of degrees of freedom or for templates with deformation models that are computationally expensive to evaluate.

Even with the limitations of the numerical approximation to the gradient it is very useful for an initial run at minimizing an error function because defining the gradient of the error function explicitly is time consuming. The numerical approximation to the gradient also provides a good method for checking the accuracy of the explicit gradient since the explicit expressions for the gradient of the error function are generally complex and prone to error.

4.3.2 Explicit Calculation of Gradient

As pointed out in the previous section, the explicit computation of the gradient is both more accurate and more efficient than computing it numerically. The downside of computing the gradient explicitly is that the explicit form can be complex because it depends on the deformation model used, the error measure used, and the affine transformation of the template. This can be seen by reviewing how the error function is calculated. The error function is

$$E(\mathbf{x}) = \sum_{j=1}^{M} \rho(d) \tag{4.14}$$

where d is the distance between the jth template pixel and the nearest image pixel and is defined as $d \equiv \left\| \mathbf{r}'_j - \mathbf{w}_j \right\|$, ρ is the error measure being used, and $\mathbf{x}$ is defined as

$$\mathbf{x} \equiv \begin{bmatrix} \theta \\ \mathbf{X} \\ \{t\} \end{bmatrix} \tag{4.15}$$

Recall that the $\mathbf{w}_j$ values are constants that represent edge vertices in the current image. The $\mathbf{r}'_j$ values represent the deformed pixel locations and are calculated using

$$\mathbf{r}'_j = \mathbf{X} + \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} (\mathbf{r}_j + \mathbf{u}) \tag{4.16}$$

where $\mathbf{r}_j$ is the undeformed template location and $\mathbf{u}$ is computed using

$$\mathbf{u} = \mathbf{D}\left(\mathbf{r}_j, \{t\}\right) \tag{4.17}$$

where $\mathbf{D}$ is the deformation model.

From the above equations, the gradient can be explicitly calculated. The chain rule will be used so the explicit gradient can be easily updated if the functions ρ or D are changed. The explicit gradient is

$$\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}(i)} = \sum_{j=1}^{M} \left(\frac{\partial \rho(d)}{\partial d} \left(\frac{\partial d}{\partial r'_{jx}} \frac{\partial r'_{jx}}{\partial \mathbf{x}(i)} + \frac{\partial d}{\partial r'_{jy}} \frac{\partial r'_{jy}}{\partial \mathbf{x}(i)} \right) \right) \tag{4.18}$$

where the partial of d with respect to the x component of $\mathbf{r}_j$ is

$$\frac{\partial d}{\partial r'_{jx}} = \frac{r'_{jx} - w_{jx}}{\left\| \mathbf{r}'_j - \mathbf{w}_j \right\|} \tag{4.19}$$

and similarly, the partial with respect to the y component is

$$\frac{\partial d}{\partial r'_{jy}} = \frac{r'_{jy} - w_{jy}}{\left\| \mathbf{r}'_j - \mathbf{w}_j \right\|} \tag{4.20}$$

Finally, the partials of r'_{jx} and r'_{jy} with respect to the degrees of freedom are computed. The partials with respect to θ are

$$\frac{\partial r'_{jx}}{\partial \theta} = -sin(\theta)(r'_{jx} + u_x) - cos(\theta)(r'_{jy} + u_y) \tag{4.21}$$

and

$$\frac{\partial r'_{jy}}{\partial \theta} = cos(\theta)(r'_{jx} + u_x) - sin(\theta)(r'_{jy} + u_y) \tag{4.22}$$

The partials with respect to the components of $\mathbf{X}$ are

$$\frac{\partial r'_{jx}}{\partial X_1} = 1 \tag{4.23}$$

$$\frac{\partial r'_{jy}}{\partial X_1} = 0 \tag{4.24}$$

$$\frac{\partial r'_{jx}}{\partial X_2} = 0 \tag{4.25}$$

$$\frac{\partial r'_{jy}}{\partial X_2} = 1 \tag{4.26}$$

The partial of $\mathbf{r}'_j$ with respect to the kth traction degree of freedom $\{t_k\}$ is

$$\frac{\partial r'_{jx}}{\partial \{t_k\}} = cos(\theta)\frac{\partial D_x}{\partial \{t_k\}} - sin(\theta)\frac{\partial D_y}{\partial \{t_k\}} \tag{4.27}$$

and

$$\frac{\partial r'_{jy}}{\partial \{t_k\}} = sin(\theta)\frac{\partial D_x}{\partial \{t_k\}} + cos(\theta)\frac{\partial D_y}{\partial \{t_k\}} \tag{4.28}$$

Using the above equations, the gradient of the error function can be calculated explicitly which efficiently provides an accurate derivative when compared to the derivative obtained using the numerical gradient approximation. Also, if the error measure ρ is changed or the deformation model $\mathbf{D}$ is changed, the explicitly gradient can be easily updated by substituting the new value for $\frac{\partial \rho}{\partial d}$ into (4.18) or the new values for the partials of $\mathbf{D}$ into (4.27) and (4.28).

4.4 Optimizing the Calculation of the Error Function

An efficient minimization algorithm has been presented in this section. Even with an efficient minimization algorithm, the efficiency of the overall minimization problem is still limited by the computation time required to evaluate the error function itself. This section presents approaches to minimize the computation time needed to evaluate the error function.

4.4.1 The KD-Tree Datastructure

The solution of the error function (2.7) requires a nearest neighbor search for each template pixel in order to find the nearest image edge pixel. This nearest-neighbor search performed for each template pixel is where the tracking algorithm presented in this dissertation spends most of its computation time. In the simplest solution to the nearest-neighbor search, one simply calculates the distance to every image vertex and selects the image vertex with the shortest distance. This algorithm works but makes no use of the spatial structure that exists within the image pixels. If the image vertex data is organized in a spatial data structure, the nearest image pixel can be found without having to measure the distance to every image pixel. The data structure employed to organize the pixel data is the KD-Tree [41].

A two dimensional KD-Tree is a binary tree data structure where each node corresponds to a data point. Each new node partitions the data space by either a vertical line or a horizontal line. Figure 4.1 shows a set of points that are partitioned into a KD-Tree. A KD-Tree can be used to find the nearest image vertex with $O(logN)$ operations as opposed to $O(N)$ operations needed to find the nearest pixel without using a spatial data structure (where N is the number of vertex points in the image).

The order in which data points are added to the tree is important for achieving the $O(logN)$ performance. In order to achieve the $O(logN)$ performance, the KD-Tree must be balanced. However, there is a computational cost to building a balanced tree that must be considered when choosing the best method to construct the KD-Tree. The time required to create the KD-Tree becomes important because for every new image frame that is read in from the camera, a new KD-Tree must be constructed. The order the data points are added to the KD-Tree determines how balanced the final KD-Tree will be. Three approaches were implemented for adding the points to the KD-Tree: the no-preprocessing approach where the data points are added in the same order that they are received from the camera (an example of such a tree is shown in Figure 4.2), the balanced approach where the data points are ordered so that a balanced tree is created (see Figure 4.1), and the randomized approach where the points are placed in random order before they are added to the tree.

The results for the three methods of adding nodes to the KD-Tree are summarized in Table 4.1. The table shows the total time for tree creation and error function minimization. It is desireable to use the algorithm with the lowest total time. It can be seen from this data that the balanced tree is the most efficient for the nearest neighbor searches since it has the fastest minimization time, however, it takes the longest time to create. The tree with no preprocessing can be created very quickly but it has slow minimization performance. The random insertion order algorithm provides the lowest total time even though it doesn't provide the lowest minimization times. The reason that the no preprocessing algorithm performs so poorly is because the data is coming from image scan lines so consecutive data points placed into the KD-Tree will have come from nearly the same location on the image (see Figure 4.2). This will lead to a tree that is very poorly balanced and little advantage will be gained by using the data structure. Randomizing the order of the data entered into the KD-Tree leads to a balanced tree without the computation cost required to generate an optimal tree.

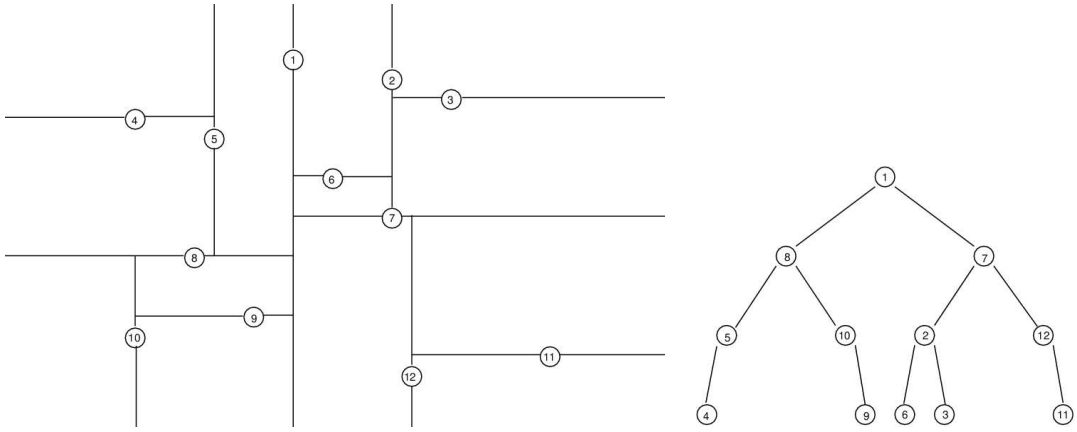


Figure 4.1: Partitioned data on left and KD-Tree on right for data that was added to tree in optimal order to create a balanced tree.

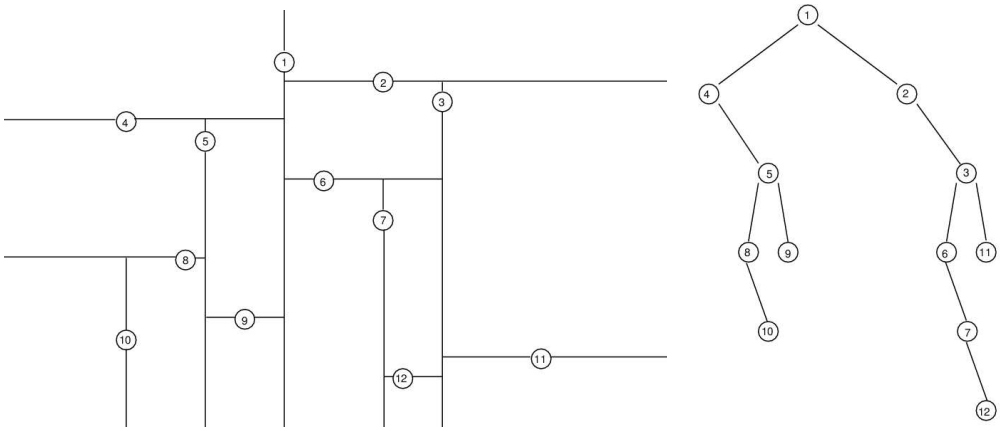


Figure 4.2: Partitioned data on left and KD-Tree on right for data that was added to tree in scan line order.

Table 4.1: Comparison of KD-Tree construction algorithms.

	Random	Balanced	No Preprocessing
Average Tree Creation Time (sec)	0.0082	0.0305	0.0033
Average Minimization Time (sec)	0.1217	0.1129	0.1897
Total Time (sec)	0.1299	0.1434	0.1930

4.4.2 Under-Sampled Template

The error function can be evaluated more quickly if the number of template pixels is reduced. This reduces the number of nearest neighbor searches that need to be performed. To reduce the number of template pixels, a certain percentage of pixels are removed randomly from the template. Figure 4.3 shows a cantilever template with all of the edge pixels along with an under-sampled template retaining ten percent of its original edge pixels (see Section 7.3 for a discussion of this template and the associated tracking problem). Figure 4.4 shows the average time for force calculation versus the percentage of template pixels that are retained. It can be seen from this plot that there is a linear relationship between the percent of template pixels and the speed of the algorithm. A linear relationship is expected because each nearest neighbor search takes approximately the same amount of time, so if there are half the number of nearest neighbor searches the time to perform the minimization should be cut in half. Also, it can be seen from the plot that the y-intercept of the line is not zero. This non-zero intersect is due to the time that is required to construct the KD-Tree before the nearest neighbor search actually begins (the time to perform the Canny edge operator was not included in the results). By reducing the number of template pixels, the time per iteration can be reduced significantly. In this case with 100 percent of the template pixels the average iteration time was approximately 0.12 sec while with 20 percent of the template pixels the average iteration time was reduced to 0.037

Reducing the number of template pixels, however, is not without drawbacks. By having less data points in (2.7), it would be expected that the accuracy of the force result obtained would be decreased. Figure 4.5 summarizes this by showing the cantilever deflection confidence intervals versus the percentage of template pixels retained. It can be seen that the confidence intervals become larger when fewer pixels are used in the template. Below 20 percent of pixels, the confidence intervals begin to grow more dramatically. Even when only one percent of the template pixels are retained (such a template would consist of approximately nine pixels where the original template consisted of 920 pixels) sub-pixel deflection confidence intervals of 0.4 m are still achieved (the pixel size is approximately 1 m). When determining the percentage of template pixels to use in a particular tracking problem it is important to balance the needs of accuracy and that of fast calculations.

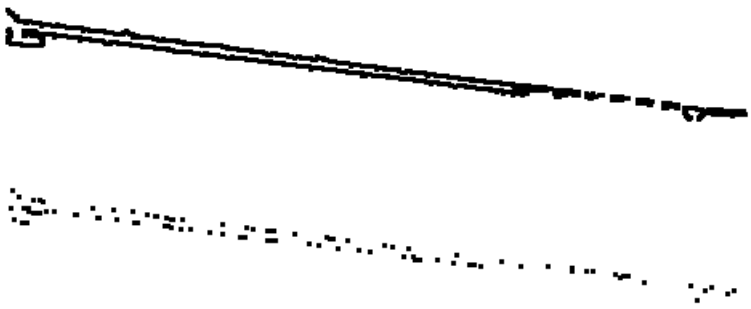


Figure 4.3: Full template from edge pixels of image on top and under-sampled template on bottom retaining only 10% of the original edge pixels.

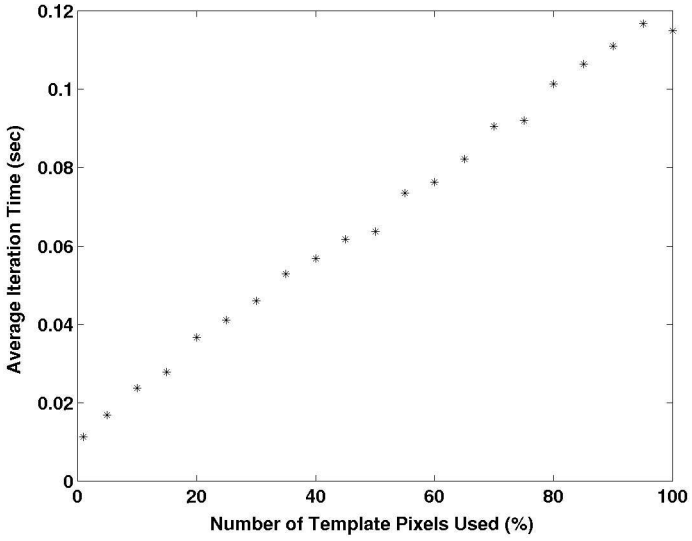


Figure 4.4: Under-sampled template performance.

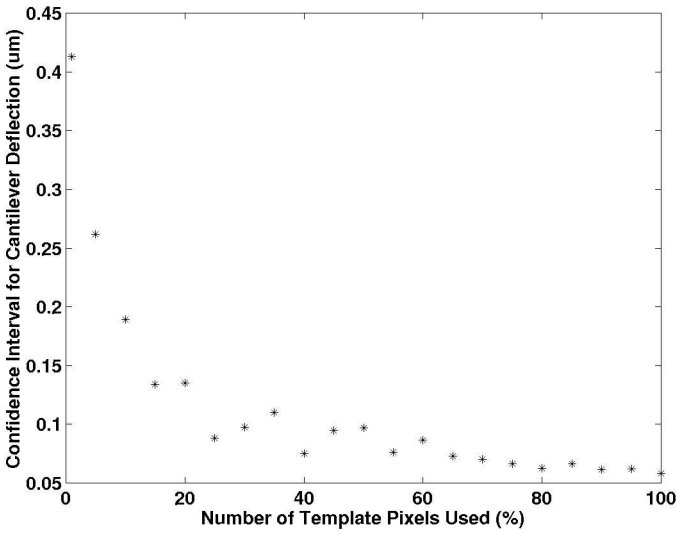


Figure 4.5: Under-sampled template effect on system accuracy.

4.4.3 Potential for Parallization Using Multiple Processors

Performance improvements similar to those achieved by decreasing the number of template pixels can be achieved by making use of multiple processors rather than removing template pixels. This can be done by separating the evaluation of the error function (2.7) into a series of partial sums where the number of partial sums is the same as the number of processors that are available. Each processor then evaluates its own partial sum and returns the result to the main processor. This method has the potential provide performance benefits similar to those provided by the under-sampled template. For example, if two processors are used the performance should be similar to the performance of the under-sampled template with 50 percent of the original pixels (ignoring the communication cost between processors). With multiple processors, the performance improvements come without the cost of decreased accuracy because all of the template pixels are retained. Even further performance enhancements could be obtained by combining the use of multiple processors with the use of an under-sampled template.

Chapter 5

A Deformable Object Tracking Implementation Using the Boundary Element Method[1]

This chapter brings together the concepts of the previous chapters and presents a complete deformable object tracking algorithm. The boundary element method (BEM) is used to model the deformations of the template. The implementation of the algorithm is discussed and the tracking results are also presented.

5.1 Introduction

Deformable object tracking has many important applications. Application areas include medical imaging [33][32], robotic manipulation of deformable objects [34][45], and vision-based force measurement [16][51]. For robust deformable object tracking, it is essential that the tracking algorithm has some knowledge about the behavior of deformable objects. Algorithms without inherit material models, such as active contour models [25], can perform a wide variety of tracking tasks but their generality sacrifices robustness when used for tracking deformable objects.

Existing model based tracking algorithms have used the finite element method (FEM) to model material deformations [33][50]. BEM, like FEM, is a method to model an elastic

[1] ©2004 IEEE. Reprinted, with permission, from "Boundary element deformable object tracking with equilibrium constraints," Greminger, M.A., Sun, Y., Nelson, B.J., IEEE International Conference on Robotics and Automation, 2004.

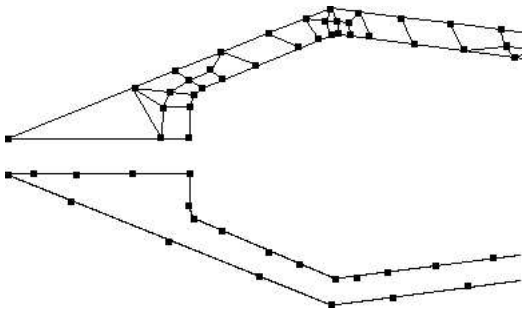


Figure 5.1: Comparison between 2D boundary element and finite element meshes.

object. BEM differs from FEM in the way the object is meshed. BEM only requires the boundary of the object to be meshed while FEM requires the interior and the boundary of the object to be meshed. Figure 5.1 shows a microgripper with an FEM mesh on the top jaw and a BEM mesh on the bottom jaw. The boundary mesh property of BEM makes it uniquely suited to computer vision problems because edge detection can be used to easily locate the boundary of an object. An even more important benefit of a BEM mesh is that it can readily handle large deformations without the need for mesh refinement. The elements of FEM meshes may become ill-conditioned or turn inside-out if they experience a large deformation.

This chapter is organized as follows: Section 5.2 presents the boundary element method and the method used to numerically solve the Neumann Problem. The actual deformable template matching algorithm that makes use of the solution to the Neumann problem is presented in Section 5.3. Section 5.4 presents tracking results for the deformable object tracking algorithm. Strain energy regularization is presented in Section 5.5 and includes tracking results which make use of this technique. Finally, a summary and concluding comments are presented in Section 5.6.

5.2 The Boundary Element Method

The boundary element method is a technique to solve partial differential equations by reformulating the original PDE into an integral equation over the boundary of an object. The solution to this boundary integral equation (BIE) is the solution to the original PDE. Because the integral equations are over the boundary of the object, only the boundary of the object needs to be partitioned.

The linearly elastic 2D plane stress model will be used to model the deformation of the deformable template. The PDE for the 2D plane stress elasticity problem can be expressed in terms of displacements $u_\alpha(x)$ by [43]

$$\mu\nabla^2 u_\alpha + \mu\left(\frac{1+\nu}{1-\nu}\right)\frac{\partial}{\partial x_\alpha}\left(\frac{\partial u_1}{\partial x_1}+\frac{\partial u_2}{\partial x_2}\right)+F_\alpha = 0 \tag{5.1}$$

where $\alpha = 1, 2$. $F_\alpha(x)$ is a body force applied to the object, such as gravity or a force due to acceleration. This equation is known as Navier's equation of plane stress and is defined over a 2D domain R with a boundary ∂R. The shear modulus μ and Poisson's ratio ν completely define the material properties of an isotropic linearly elastic object. The shear modulus can be expressed in terms of the modulus of elasticity E by the following equation

$$\mu = \frac{E}{2(1+\nu)}$$

The boundary conditions for (5.1) can be expressed as a prescribed displacement vector over the boundary ∂R. This is known as the Dirichlet problem. The boundary condition can alternatively be expressed as a prescribed traction vector over the boundary where, in 2D, the traction vector has the units of force per length. This is known as the Neumann problem.

In order to convert Navier's equation (5.1) into a boundary integral equation, it is first necessary to obtain the fundamental solutions for (5.1). The fundamental solutions can then be used to construct the boundary integral equation.

5.2.1 Fundamental Solutions

The fundamental solutions for the plane stress elasticity problem are the solutions to (5.1) for a point load F of unit magnitude applied to a point p in an infinite 2D medium of unit thickness. The fundamental solutions are sometimes referred to as Kelvin solutions, Green's functions, or singular solutions. The displacement of a point q in an infinite medium with a unit load applied at p is known as the displacement fundamental solution and is given by [2]

$$U_{lk}(p,q) = C_1\left[C_2 \ln\frac{1}{r}\delta_{lk} + \frac{(p_l - q_l)(p_k - q_k)}{r^2}\right] \tag{5.2}$$

where

$$\begin{aligned} r &= \left[(p_1 - q_1)^2 + (p_2 - q_2)^2\right]^{\frac{1}{2}} \\ C_1 &= \frac{1}{8\pi\mu(1-\nu)} \\ C_2 &= 3 - 4\nu \end{aligned}$$

and $l, k = 1, 2$. U_{lk} corresponds to the displacement of the point q in the kth direction due to a unit load in the lth direction. The displacement fundamental solution is shown graphically in Figure 5.2 for the unit load $F = (1, 0)$.

There is also a fundamental solution that gives the traction at a point q in an infinite medium due to a unit load at p. The traction vector must be defined in reference to a line l that cuts through the material. The traction vector is the force distribution that would need to be applied to the object in order to maintain the same state of stress if it were to be cut by the line l. The traction fundamental solution can be written as

$$\begin{aligned} T_{lk}(p, q) = \frac{C_3}{r}\Bigg[\frac{\partial r}{\partial n}\left(C_4\delta_{lk} + 2\frac{(p_l - q_l)(p_k - q_k)}{r^2}\right) \\ + C_4\left(\frac{n_l(p_k - q_k)}{r} - \frac{n_k(p_l - q_l)}{r}\right)\Bigg] \end{aligned} \tag{5.3}$$

where

$$\begin{aligned} \frac{\partial r}{\partial n} &= \frac{n_1(p_1 - q_1)}{r} - \frac{n_2(p_2 - q_2)}{r} \\ C_3 &= \frac{1}{4\pi(1-\nu)} \\ C_4 &= 1 - 2\nu \end{aligned}$$

and n_l is the outward normal vector to the line l at the point q. The traction fundamental solution is shown graphically in Figure 5.2 for the unit load $F = (1, 0)$. Note that the fundamental solutions are singular when $p = q$.

5.2.2 Boundary Integral Equations

The fundamental solutions (5.2) and (5.3) are used to construct the boundary integral equations for the elasticity problem (5.1). The integral equation that relates interior displacements to boundary displacements and boundary tractions is known as Somigliana's identity

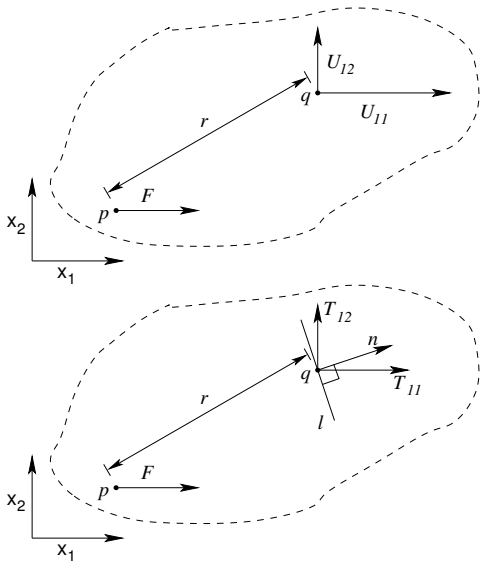


Figure 5.2: Displacement fundamental solution on top and traction fundamental solution on bottom.

and is as follows [40]

$$u_i(p) = \int_{\partial R} \left[U_{ij}(p,q) t_j(q) - T_{ij}(p,q) u_j(q) \right] \mathrm{d}\partial R(q) \tag{5.4}$$

for $\forall p \in R \setminus \partial R$, $i, j = 1, 2$, u_j is a displacement vector, and t_j is a traction vector. The fundamental solutions act as the kernel functions in Somigliana's identity. The singularities in these kernel functions do not affect the evaluation of the integral equation because p cannot be on the boundary ∂R so p and q cannot coincide. Somigliana's identity gives the displacement of any point p within a body but it requires the knowledge of the displacements and tractions on the boundary. In general only the displacements on the boundary are known in the case of the Dirichlet problem or only the tractions on the boundary are known in the case of the Neumann problem. In order to solve the elasticity problem it is necessary to have an integral equation where the point p is on the boundary of the object. This equation is known as Somigliana's boundary identity and is [40]

$$\begin{aligned} c_{ij} u_j(p) = \mathrm{PV} \int_{\partial R} & \left[U_{ij}(p,q) t_j(q) \right. \\ & \left. - T_{ij}(p,q) u_j(q) \right] \mathrm{d}\partial R(q) \end{aligned} \tag{5.5}$$

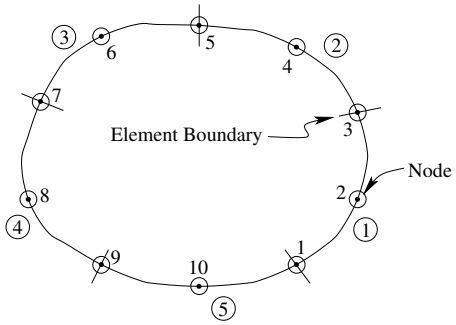


Figure 5.3: Boundary mesh.

for $\forall p \in \partial R$. The term c_{ij} is a constant that is $\frac{1}{2}\delta_{ij}$ if the boundary ∂R is smooth at the point p. If a corner exists at p then the value of c_{ij} depends on the angle formed by the corner. It will be seen in the next section that the explicit calculation of c_{ij} will be unnecessary in the numerical solution of the problem. The PV indicates that the integral exists only in the sense of a Cauchy principal value because the integrand becomes unbounded when $p = q$ due to the singular kernel functions. The numerical treatment of these singularities will be addressed in Section 5.2.3.

5.2.3 Partitioning of the Boundary Integral Equations

In order to numerically solve the boundary integral equation (5.5) it is necessary to partition the boundary of the object. The partitioned boundary is shown in Figure 5.3. The boundary is partitioned into elements with each element having three nodes. The ith node of the mesh is at the position x^i. The values of displacement u and traction t are defined at each node and interpolation is used to evaluate the displacement and traction values between the nodes. Quadratic interpolation is used with the following shape functions [2]

$$\begin{aligned} \phi_1(s) &= \frac{1}{2}s\,(s-1) \\ \phi_2(s) &= \left(1-s^2\right) \\ \phi_3(s) &= \frac{1}{2}s\,(s+1) \end{aligned} \tag{5.6}$$

where s is a parameter that varies from -1 to 1 along the length of an element as shown in

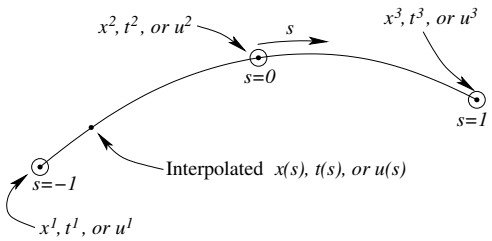


Figure 5.4: Elemental interpolation of geometry x, displacements u, and tractions t.

Figure 5.4. The value of u at any point along an element is given by

$$u_i(s) = \sum_{j=1}^{3} \phi_j(s) u_i^j \tag{5.7}$$

where the superscript j indicates the node number. An interpolation equation analogous to (5.7) can be written for the object geometry x and the surface tractions t. Now that it has been defined how x, u and t are interpolated within an element, it is necessary to decide what continuity will be enforced between elements. For the boundary geometry x and the boundary displacements u at least $\mathcal{C}^0$ continuity is needed between elements, otherwise there could be gaps in the boundary of the object. It could be required that x and u also have continuous derivatives between boundaries but this requirement would not allow ∂R to have corners. Because of the need to model objects with corners, only $\mathcal{C}^0$ continuity will be enforced on x and u between elements. In order to model objects with corners, it is necessary to place an element boundary at the location of the corner.

The continuity requirement between elements for boundary tractions will be more relaxed because it is possible to have surface loadings that are discontinuous. Figure 5.5 shows a portion of a partitioned boundary that has a corner between elements 1 and 2. In this figure the geometry x has $\mathcal{C}^0$ continuity between the two elements while the traction t is discontinuous between the two elements. Since continuity is not required for the traction vector between elements, there will be more traction degrees of freedom then there are displacement degrees of freedom. In general for a boundary element model with N elements, there are $2N$ nodes, $4N$ displacement degrees of freedom (each node has an u_1 and an u_2 degree of freedom) and $6N$ traction degrees of freedom.

Next, (5.5) will be expressed in a form that can be numerically computed. The first step is to break the integral in (5.5) into N integrals, one for each element. Integrating over each

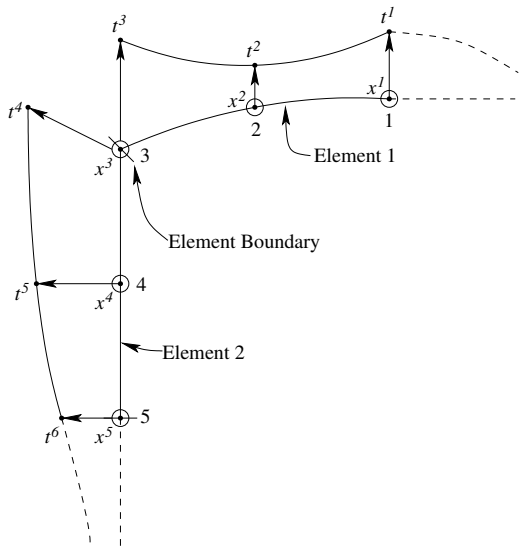


Figure 5.5: Boundary tractions t can have discontinuities between elements while boundary geometry x must have $\mathcal{C}^0$ continuity between elements.

element the following equation is obtained

$$c_{ij}u_j(p) = \sum_{k=1}^{N} \int_{-1}^{1} \left[U_{ij}(p, q(s)) \left(\sum_{n=1}^{3} \phi_n(s) t_j^n \right) - T_{ij}(p, q(s)) \left(\sum_{n=1}^{3} \phi_n(s) u_j^n \right) \right] J(s)\, \mathrm{d}s \tag{5.8}$$

where

$$\begin{aligned} q_i(s) &= \sum_{n=1}^{3} \phi_n(s) x_i^n \\ J(s) &= \frac{\mathrm{d}\partial R}{\mathrm{d}s} \\ &= \sqrt{\left(\sum_{n=1}^{3} \frac{\partial \phi_n(s)}{\partial s} x_1^n \right)^2 + \left(\sum_{n=1}^{3} \frac{\partial \phi_n(s)}{\partial s} x_2^n \right)^2} \end{aligned}$$

The Jacobian term $J(s)$ is required because the integration is now over the local parameter s. Since the nodal values u^n and t^n are constant over each element, they can be taken out

of the integral yielding

$$c_{ij}u_j(p) = \sum_{k=1}^{N}\sum_{n=1}^{3} t_j^n \int_{-1}^{1} U_{ij}(p,q(s))\phi_n(s)J(s)\,\mathrm{d}s - \sum_{k=1}^{N}\sum_{n=1}^{3} u_j^n \int_{-1}^{1} T_{ij}(p,q(s))\phi_n(s)J(s)\,\mathrm{d}s \tag{5.9}$$

Switching to matrix notation

$$\mathbf{cu}(p) + \sum_{k=1}^{N}\sum_{n=1}^{3} \Delta\mathbf{T}_n\mathbf{u}_n = \sum_{k=1}^{N}\sum_{n=1}^{3} \Delta\mathbf{U}_n\mathbf{t}_n \tag{5.10}$$

where

$$\Delta\mathbf{U}_n = \int_{-1}^{1} \phi_n(s)\mathbf{U}(p,q(s))J\,\mathrm{d}s$$
$$\Delta\mathbf{T}_n = \int_{-1}^{1} \phi_n(s)\mathbf{T}(p,q(s))J\,\mathrm{d}s \tag{5.11}$$

and $\mathbf{c}$, $\Delta\mathbf{T}_n$, and $\Delta\mathbf{U}_n$, are 2×2 matrices and $\mathbf{u}_n$ and $\mathbf{t}_n$ are 2×1 vectors. The boundary integral equation has now been partitioned into sums of integrals over each element. This equation applies for all points p on the boundary of the object including the nodal points. Next, (5.10) is applied to all 2N nodes in the boundary element mesh and obtain 2N vector equations (4N scalar equations) that can be used to solve the elasticity problem. These equations are

$$\mathbf{cu}(p_i) + \sum_{k=1}^{N}\sum_{n=1}^{3} \Delta\mathbf{T}_n\mathbf{u}_n = \sum_{k=1}^{N}\sum_{n=1}^{3} \Delta\mathbf{U}_n\mathbf{t}_n \tag{5.12}$$

where $i = 1, 2, ..., 2N$. The equation can now be written in a matrix form that can be used to solve the elasticity problem as follows

$$[I]\{cu\} + [F]\{u\} = [G]\{t\} \tag{5.13}$$

where $\{u\}$ is a $4N\times 1$ vector containing all of the nodal displacements and $\{t\}$ is a $6N\times 1$ vector containing all of the nodal tractions. $[F]$ and $[G]$ are $4N\times 4N$ and $4N\times 6N$ matrices respectively which consist of the element integrals (5.11). It should be noted that for the diagonal entries of these matrices, the point p_i will be coincident with one of the nodes in the element making the kernel functions singular. Because of this, special care needs to be taken with these integrals. The term $[I]\{cu\}$ can be combined with the $[F]\{u\}$ to obtain

the following equation

$$[F']\{u\} = [G]\{t\} \tag{5.14}$$

where the matrix $[F']$ contains the c term information along its diagonal. The matrices $[F']$ and $[G]$ depend completely on the geometry of the object and can be computed in advance for a given object. The off diagonal terms of these matrices can easily be computed using a standard quadrature routine such as Gauss integration. The singularity in the diagonal elements of $[G]$ is due to the singularity in the displacement fundamental solution (5.2). This singularity is of the form $\ln r$ which can be computed numerically if the appropriate quadrature rule is used. The package used to calculate the singular integrals was the GNU Scientific Library (GSL) which uses an adaptive routine specifically designed to handle singularities [14].

The singularity on the diagonal terms of the matrix $[F']$ is more difficult to evaluate because the singularity in these terms is due to the singularity in the traction fundamental solution (5.3) which is of the form $(1/r)$. This singularity cannot be readily computed numerically but it turns out that the explicit calculation of these integrals can be avoided by making the following observation [2]. If the vector $\{u\}$ consists of only a rigid body motion, the product $[F']\{u\}$ will be zero. This is true because a pure rigid body motion of an elastic object must lead to a stress free state constraining $\{t\}$ to be zero. If $\{t\}$ were not zero, there would some deformation in the object and the displacement would not be pure rigid body displacement. From this observation, the following equations can be generated

$$\begin{aligned} [F']\left\{ \begin{array}{cccccccc} 1 & 0 & 1 & 0 & \cdots & 1 & 0 \end{array} \right\}^{\mathrm{T}} &= 0 \\ [F']\left\{ \begin{array}{cccccccc} 0 & 1 & 0 & 1 & \cdots & 0 & 1 \end{array} \right\}^{\mathrm{T}} &= 0 \end{aligned} \tag{5.15}$$

Using these equations, the two singular entries in each row of the matrix can be expressed in terms of the other non-singular entries in the row. Note that the c term from (5.13) was also contained in the diagonal of the matrix $[F']$ so it is not necessary to explicitly calculate that term.

Now that the system matrices $[F']$ and $[G]$ are defined, a solution to the elasticity problem can be obtained. For the Dirichlet problem, the nodal displacements $\{u\}$ are known and (5.14) can be used to calculate the nodal tractions $\{t\}$. For the Neumann problem, the nodal tractions $\{t\}$ are known and (5.14) can be used to calculate the nodal displacements $\{u\}$. For the deformable template application the Neumann problem is the problem of interest because it is necessary to apply a force distribution to the template and then calculate

the resulting nodal displacements.

5.2.4 Solution of the Neumann Problem

The Neumann problem is defined as the case where the surface tractions $\{t\}$ are known and the surface tractions $\{u\}$ need to be computed using (5.14). It can be shown [22] that a solution to the Neumann problem exists if and only if

$$\int_{\partial R} t_i(q)\,\mathrm{d}\partial R(q) = 0 \tag{5.16}$$

where $i = 1, 2$. An object that satisfies (5.16) is said to be in a state of static equilibrium. If (5.16) is satisfied then there are an infinite number of solutions $\{u\}$ to (5.14). All of the solutions differ only by a rigid body displacement. Because there are an infinite number of solutions, the matrix $[F']$ is not invertible. The rank of $[F']$ is $4N-3$. There is one linearly dependent row in $[F']$ for each of the 3 possible 2D rigid body motions. Since $[F']$ is not invertible, singular value decomposition (SVD) is used to solve (5.14). SVD decomposes $[F']$ into the following form [37]

$$[F'] = [U][W][V]^{\mathrm{T}} \tag{5.17}$$

where $[W]$ is a diagonal matrix consisting of the singular values of $[F']$ and $[U]$ and $[V]$ have columns that are orthonormal. Because the rank of $[F']$ is $4N-3$, three of the singular values will be zero or very nearly zero. A unique solution can be obtained to the Neumann problem by using the following equation [37]

$$\{u\} = [V][\mathrm{diag}\,(1/w_j)][U]^{\mathrm{T}}[G]\{t\} \tag{5.18}$$

where w_j are the singular values and the value $(1/w_j)$ is set to zero for the three singular values near zero. This method finds the solution with the smallest magnitude $\|\{u\}\|^2$. This has the effect of removing the rigid body motion component from the solution. Defining $[A] = [V][\mathrm{diag}\,(1/w_j)][U]^{\mathrm{T}}[G]$, (5.18) can be rewritten as

$$\{u\} = [A]\{t\} \tag{5.19}$$

where $[A]$ only depends on the object's geometry so it can be computed off-line.

Once the nodal displacements $\{u\}$ are found by solving the Neumann problem, the

displacement at any point on the boundary can be found by using the interpolation functions (5.6).

5.3 The BEM Deformable Template Matching Algorithm

The deformable template is registered to a binary edge image using the template matching algorithm described in Chapter 2.

5.3.1 The Error Function for BEM Deformable Object Tracking

BEM will be used to model the non-rigid portion of the object's motion. To do this, the template is deformed according to the BEM model before performing the affine transformation. The deformation transformation (2.6) becomes

$$\mathbf{r}' = T(\mathbf{r}, \theta, \mathbf{X}, \{t\}) = \mathbf{X} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} (\mathbf{r} + \mathbf{u}) \tag{5.20}$$

where $u(\mathbf{r}, \{t\})$ is the displacement of the template edge pixel $\mathbf{r}$ due to the applied traction distribution $\{t\}$ on the boundary of the object. The displacement vector $u(\mathbf{r}, \{t\})$ is obtained from the solution to the Neumann problem (5.19). The error function (2.4) becomes

$$E(\theta, \mathbf{X}, \{t\}) = \sum_{i=1}^{M} \|\mathbf{r}'_i - \mathbf{w}_i\|^2 \tag{5.21}$$

Minimizing the above error function will give the traction distribution $\{t\}$ that best fits the image in a least squares sense. As mentioned previously, in order for there to be a solution to the Neumann problem, $\{t\}$ must satisfy (5.16). Therefore, the minimization of (5.21) is a constrained minimization problem. The next section demonstrates how to convert this constrained minimization problem into an unconstrained minimization problem of reduced dimensionality.

5.3.2 Constrained Minimization

The Neumann problem can only be solved if the equilibrium condition (5.16) is satisfied. Therefore, when (5.21) is minimized, $\{t\}$ must be constrained to satisfy (5.16). A method is presented in this section that solves the constrained minimization problem efficiently.

The first step is to express (5.16) in discrete form. The equilibrium condition can be partitioned into sums of integrals over each of the N elements as follows:

$$\sum_{k=1}^{N}\sum_{n=1}^{3}\Delta\mathbf{P}_n\mathbf{t}_n = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5.22}$$

where

$$\Delta\mathbf{P}_n = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\int_{-1}^{1}\phi_n(s)J(s)\,\mathrm{d}s \tag{5.23}$$

This discrete form can now be written as a matrix equation.

$$[E]^{\mathrm{T}}\{t\} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5.24}$$

where $[E]$ is a $6N \times 2$ matrix. The above constraint equation is a linear equality constraint. The Generalized Elimination Method as described by Fletcher [13] will be used to eliminate two variables from the optimization problem by applying a linear transformation. Applying this transformation, the minimization problem (5.21) can be recast in the form:

$$E(\theta, \mathbf{X}, [Z]\{y\}) = \sum_{i=1}^{M}\left\|\mathbf{r}_i' - \mathbf{w}_i\right\|^2 \tag{5.25}$$

where $\{Z\}$ is a $6N \times (6N-2)$ matrix. Minimizing (5.25) gives values for θ, $\mathbf{X}$ and $\{y\}$ that best match the image in a least squares fashion. The traction distribution can then be obtained by the transformation $\{t\} = [Z]\{y\}$. The matrix $[Z]$ acts as a transformation that maps any $\{y\} \in \Re^{6N-2}$ to a traction $\{t\} \in \Re^{6N}$ which satisfies equilibrium. As suggested by Fletcher, QR decomposition can be used to find a $[Z]$ satisfying this property (the choice of $[Z]$ is not unique). The QR decomposition for $[E]$ can be written as

$$[E] = [Q]\begin{bmatrix} R \\ 0 \end{bmatrix} = [Q_1 Q_2]\begin{bmatrix} R \\ 0 \end{bmatrix} = [Q_1][R] \tag{5.26}$$

where $[Q]$ is a $6N \times 6N$ orthogonal matrix, $[R]$ is a 2×2 upper triangular matrix, and $[Q_1]$ and $[Q_2]$ are $6N \times 2$ and $6N \times 6N-2$ matrices, respectively. The transformation $[Z]$ is set to $[Q_2]$.

The template matching algorithm has now been recast into an unconstrained form of reduced dimensionality, (5.25), that can be minimized using standard gradient based mini-

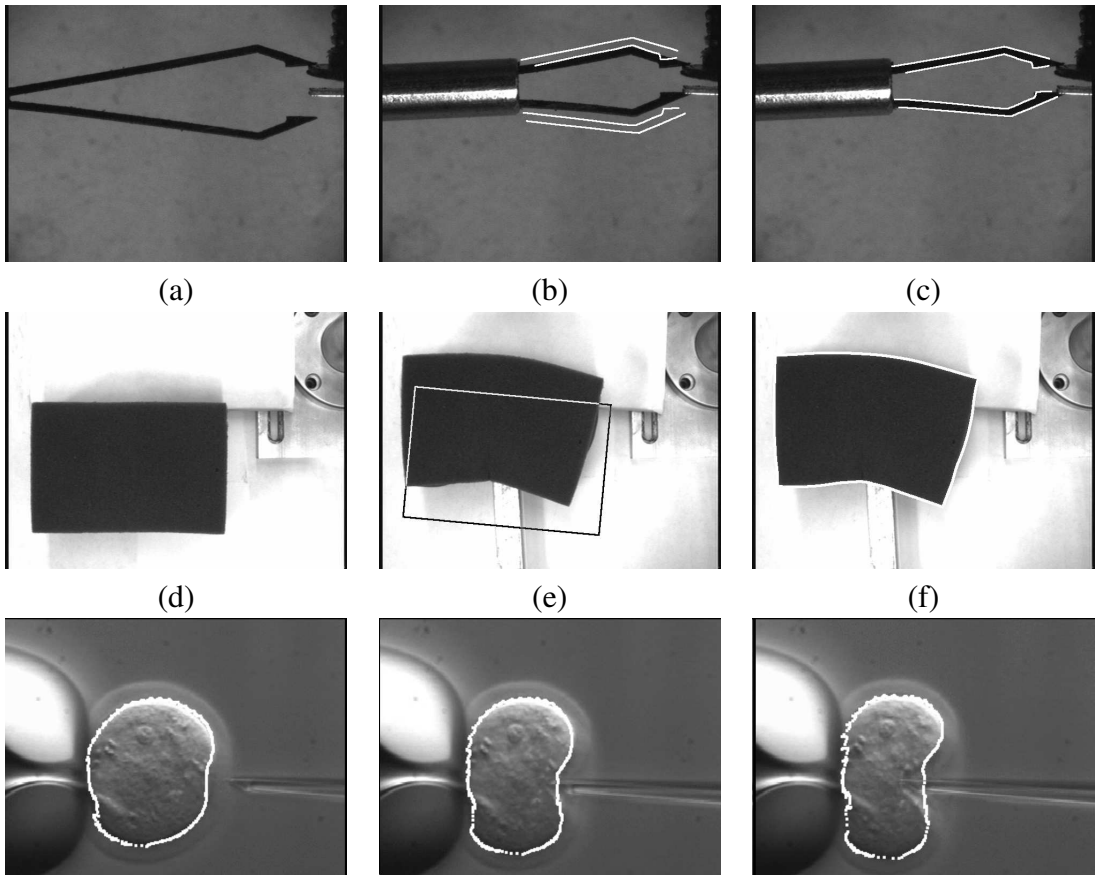


Figure 5.6: (a)-(c) BEM deformable template results when applied to a microgripper, (d)-(f) to a foam block, and (g)-(i) to a mouse oocyte (approximately 50 μm in diameter in the undeformed state).

mization techniques.

5.4 Initial BEM Template Matching Results

The results of the BEM template matching algorithm applied to a microgripper, foam block and a mouse embryo cell are shown in Figure 5.6. The microgripper in Figures 5.6(a-c) is used for robotic micro-assembly tasks [52]. A 90 node BEM mesh is used to model the gripper and is shown in Figure 5.7. Only a portion of the gripper's contour was used in tracking. This is necessary because much of the gripper is occluded by the tube that closes it (Chapter 6 describes a method to handle this occlusion in a robust way without

alternating the template). The modulus of elasticity used by the BEM model was set to that of spring steel, the material used to construct the gripper.

Unlike the gripper, which satisfies the assumptions for linear elasticity, the foam block and the cell exhibit deformations that do not satisfy these assumptions. The BEM model used to deform the template assumes linearly elasticity, but, as can be seen from Figures 5.6(d-i), the deformable template was able to track the non-linear objects. In order to allow the template to track these large deformations, the modulus of elasticity used in the BEM model was decreased to make the template softer.

Figures 5.6(g-i) show three frames from the tracking of a mouse oocyte. The cell is being robotically injected [45]. Tracking deformations of the cell can provide useful feedback for the robotic injection process and can be used in combination with a force sensor to learn more about the material properties of cells. Note that the horizontal edges were removed from the image during the tracking process in order to prevent the template from being attracted to the pipette.

If the objects being tracked are linearly elastic, the material properties used in the BEM model and the traction distribution $\{t\}$ applied to the template will corresponding to real world values. For non-linearly elastic objects such as the foam block or the cell, the material properties and the forces applied to the template will have no relation to real world values.

Figure 5.8(a) shows the a template matched to a computer generated image of a deformable gripper (the template is shown in black on the top jaw of the gripper). The figure also shows the undeformed BEM template, the forces applied to the template in order to match the image and the stress distribution within the matched template.

5.5 Strain Energy Regularization

The BEM tracking algorithm as described above only constrains the forces to satisfy static equilibrium and contains no regularization terms. The lack of regularization can lead to overfitting when the image contains noise or the object in the image does not match the template's geometry exactly. To prevent overfitting, a regularization term can be added to the energy function (5.25). The minimization problem becomes

$$E(\theta, \mathbf{X}, [Z]\{y\}) = \sum_{i=1}^{M} \left\|\mathbf{r}'_i - \mathbf{w}_i\right\|^2 + \alpha SE(\{t\}) \tag{5.27}$$

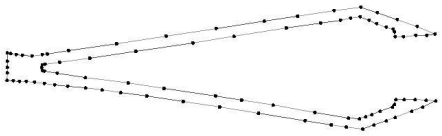


Figure 5.7: BEM mesh used to track microgripper.

where $SE(\{t\})$ is the strain energy of the BEM model as a function of the traction distribution applied to the model and is defined as

$$SE(\{t\}) = \int_R \left[\frac{1}{2E} \left(\tau_{11}^2 + \tau_{22}^2 \right) - \frac{\nu}{E} \tau_{11} \tau_{22} + \frac{1}{2G} \tau_{12}^2 \right] \mathrm{d}R \tag{5.28}$$

The α term in (5.27) determines how much the model can deform. A large value of α will prevent the model from deforming at all while a small value of α will permit overfitting to occur.

The definition of strain energy is the amount of work that must be applied to the object in order to deform it from its undeformed shape to its deformed shape. Using strain energy for regularization leads to smoother tracking results because it takes a large amount of work to deform the template to a shape that is not smooth. Strain energy measures have been used for regularization in other tracking algorithms [54] [4] [32].

5.5.1 Numerical Calculation of Strain Energy

In order to evaluate the integral (5.28) efficiently, the domain R must be divided into area elements. Two-dimensional Gauss integration is used to evaluate the integral numerically over each of the area elements. Since the strain energy is a function of internal stress, the internal stress must be computed at the Gauss points as a function of the applied tractions $\{t\}$. This can be done using the following boundary integral equation derived from Somigliana's identity (5.4) using the stress-displacement relationship for plane stress [15]

$$\begin{aligned} \tau_{ij}(p) = \int_{\partial R} & \left[S_{ijk}(p,q) t_k(q) \right. \\ & \left. - R_{ijk}(p,q) u_k(q) \right] \mathrm{d}\partial R(q) \end{aligned} \tag{5.29}$$

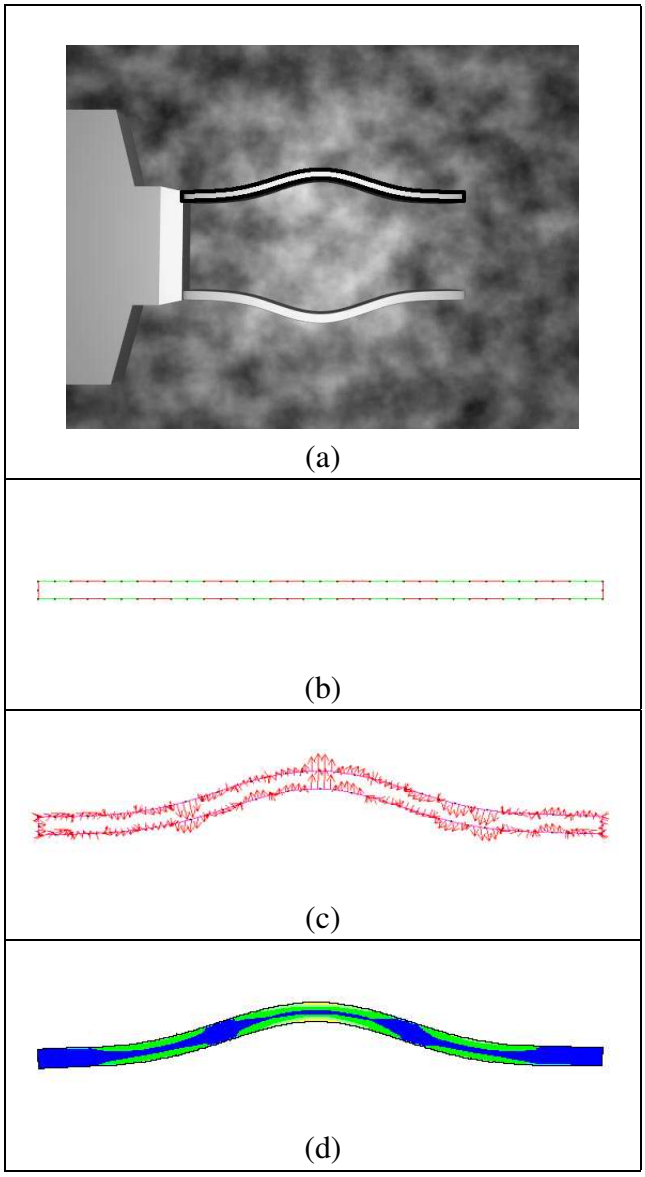


Figure 5.8: (a) Gripper jaw tracking solution, (b) undeformed mesh, (c) the traction distribution $\{t\}$, and (d) the interior stress field where lighter colors indicate higher stresses.

where $p \in R \setminus \partial R$ and S_{ijk} and R_{ijk} can be computed in terms of U_{ij} and T_{ij} using the stress-displacement relationship for plane stress which results in the following equations

$$S_{ijk}(p,q) = \left(\frac{\nu E}{1-\nu^2}\right)\delta_{ij}U_{mk,m} + G\left(U_{ik,j} + U_{jk,i}\right) \tag{5.30}$$

and

$$R_{ijk}(p,q) = \left(\frac{\nu E}{1-\nu^2}\right)\delta_{ij}T_{mk,m} + G\left(T_{ik,j} + T_{jk,i}\right) \tag{5.31}$$

Stress can only be calculated within the interior of the object using (5.29) because the kernel functions are highly singular and cannot be integrated when $p \in \partial R$ [15]. However, this is acceptable since all of the Gauss points will lie within the interior of the object.

The equation for stress (5.29) can be expressed in matrix form in a manner analogous that used to obtain (5.13). Doing this, the following is obtained

$$\begin{Bmatrix} \tau_{11} \\ \tau_{22} \\ \tau_{12} \end{Bmatrix} = [S]\{t\} - [R]\{u\} \tag{5.32}$$

where $[S]$ and $[R]$ are $3 \times 6N$ and $3 \times 4N$ matrices respectively. The solution of the Neumann problem (5.19) can be used to express the above equation as a function of boundary tractions alone giving the equation

$$\begin{Bmatrix} \tau_{11} \\ \tau_{22} \\ \tau_{12} \end{Bmatrix} = \left([S] - [R][A]\right)\{t\} \tag{5.33}$$

For each Gauss point used to calculate the strain energy over the entire domain R, the matrices $[S]$ and $[R]$ can be stored so that future computations of the strain energy can be computed efficiently.

5.5.2 Strain Energy Regularization Results

Figure 5.9 shows the results of applying strain energy regularization to the boundary element tracking algorithm. The algorithm is applied to the tracking of a rubber torus. The rubber torus is about 1 cm in diameter. Figures 5.9(a,b) show the tracking of the torus without using strain energy regularization and Figure 5.9(c,d) show the tracking solution

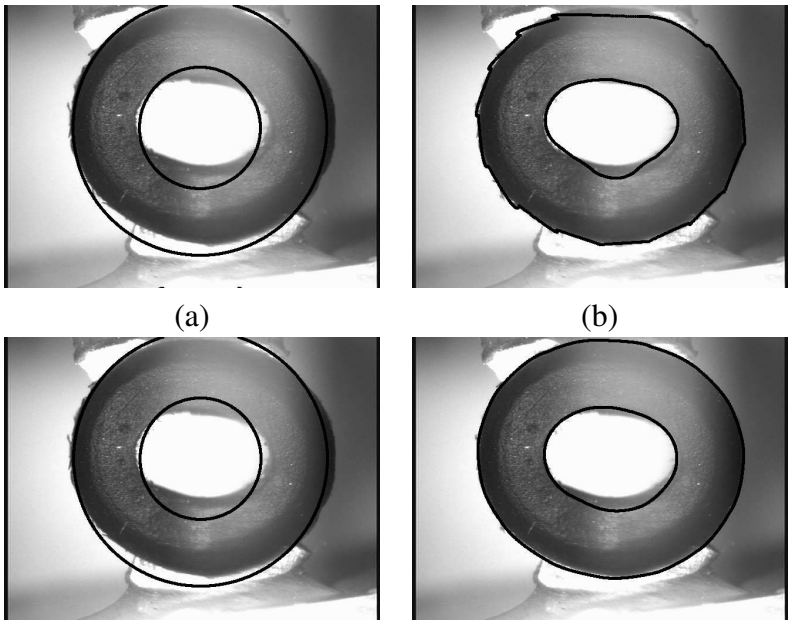


Figure 5.9: Results obtained tracking a rubber torus when (b) strain energy regularization is not used and when (d) strain energy regularization is used. The initial template locations are shown to the left of each solution and the templates are shown in black.

using strain energy regularization. The initial template location and shape are shown on the left of the figure and the tracking solutions are shown on the right. It can be seen that without strain regularization the template overfits the image and giving a non-smooth tracking result.

5.6 Summary and Conclusions

This Chapter has presented a deformable object tracking algorithm based on the boundary element method that is capable of tracking general two dimensional objects. Even though the model is based on linear elasticity, it was shown that the algorithm is able to track objects with non-linear material properties. Chapter 8 will present an application that relies on this property to generate training data for neural networks.

Strain energy regulation was presented as a add-on to the BEM tracking algorithm to help insure smooth tracking results. This is important for tracking objects undergoing large deformations or when there is a large amount of image noise.

BEM deformable object tracking is a valuable tool for the manipulation of deformable objects or for learning the material models of elastic objects.

Chapter 6

Deformable Object Tracking Robust to Occlusions and Spurious Edges

6.1 Introduction

In order for the deformable object traking algorithm to be useful for robotics applications, it must be robust. Common causes of tracking errors are the presence of spurious edges and occlusions of the object being tracked. Spurious edges are edges from objects other than those from the object of interest. These edges act to attract the deformable template away from the object of interest. Spurious edges must be eliminated in order to obtain accurate results. Occlusions occur when an object in the scene blocks the view of the object of interest. Occlusions can occur frequently in robotic applications and the tracking algorithm must be able to provide useful results with the information available even when there is significant occlusion.

This chapter presents modifications to the BEM deformable object tracing algorithm presented in the previous chapter that make it robust to spurious edges and occlusions (see Figure 6.1 for a BEM tracking example). It is first shown how this algorithm is modified to include a robust error measure that addresses the problem of occlusion. Next, a modification to the edge detection algorithm is introduced that rejects spurious edges. Two methods are used to reject spurious edges, an interior intensity based edge classifier and a neural network edge classification method. The performance of each of these methods is demonstrated for the problem of tracking a gripper in a highly cluttered environment.

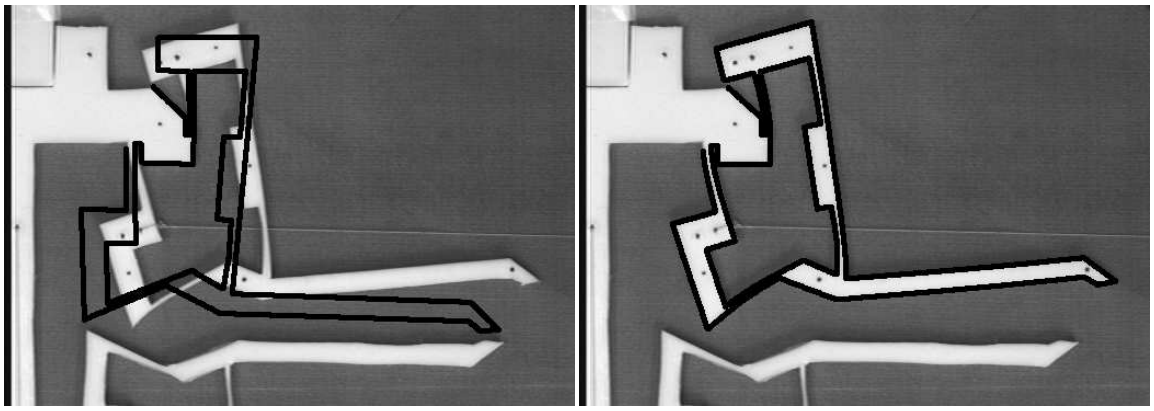


Figure 6.1: Deformable object tracking example using BEM to model deformations. The template's initial condition is shown on the left and the tracking solution is shown on the right.

6.2 Robustness to Occlusion

The least squares error measure used to calculate (2.7) works well in many situations, however, use of this error measure implicitly assumes that the errors between the template and the edge image are normally distributed [12]. Situations where the errors in the image are not normally distributed often occur. One common case where errors are not normally distributed occurs when there is occlusion. A least squares error measure does not provide the correct solution when there is occlusion as can be seen in Figure 6.2(b).

A measure not affected by occlusion is required. The Cauchy error measure given by [44]

$$\rho(r) = log\left(1 + \frac{1}{2}r^2\right) \tag{6.1}$$

where r is the error, has robust properties particularly relevant to this problem. For the template matching application, r is defined to be the distance from each template vertex to the nearest image edge vertex. Using this error function, the minimization problem (5.21) becomes

$$E(\theta, \mathbf{X}, \{t\}) = \sum_{i=1}^{M} log\left(1 + \frac{1}{2}\left\|\mathbf{r}'_i - \mathbf{w}_i\right\|^2\right) \tag{6.2}$$

Using this robust error measure, a deformable object can be successfully tracked even when a large portion is occluded. A tracking result using the Cauchy error measure is shown in Figure 6.2(c).

The Cauchy error measure has better performance in this situation because the Cauchy

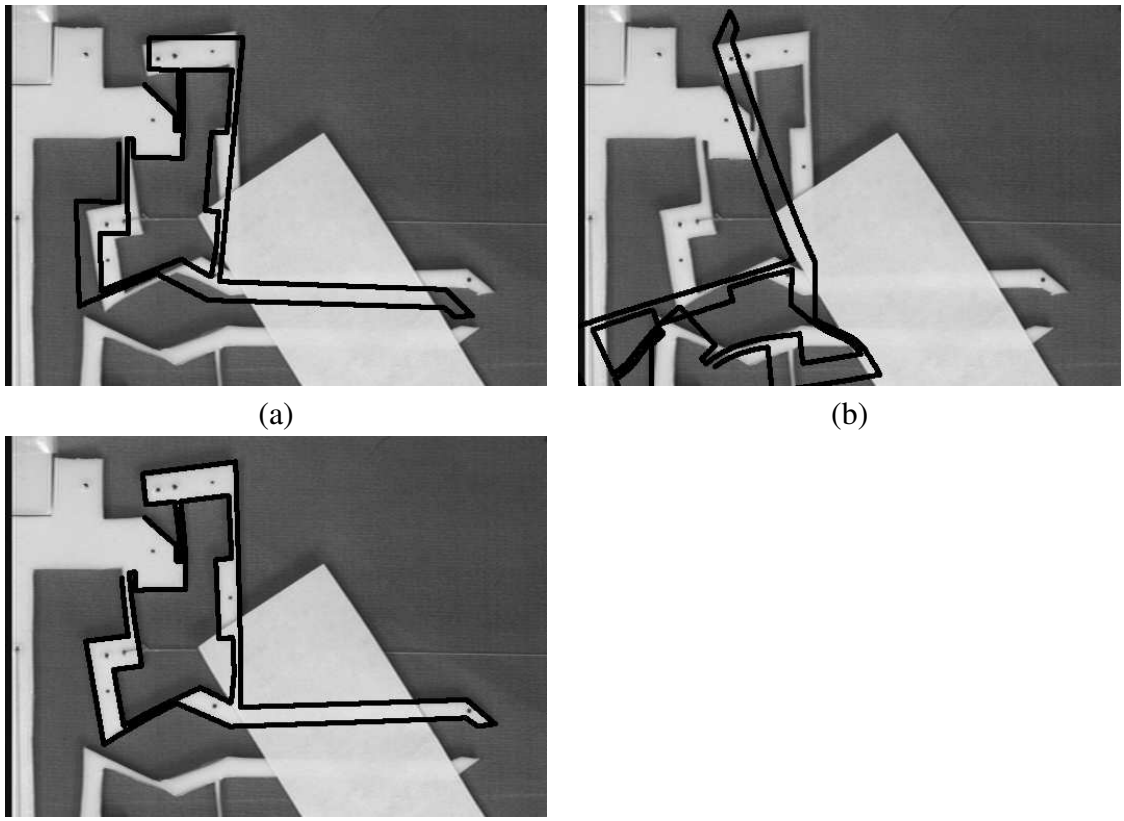


Figure 6.2: Effect of error measure on tracking results when tracking a partially occluded object. (a) The initial template location and shape for both algorithms, (b) the tracking solution when using the least squares error measure, and (c) the tracking solution when using the Cauchy error measure.

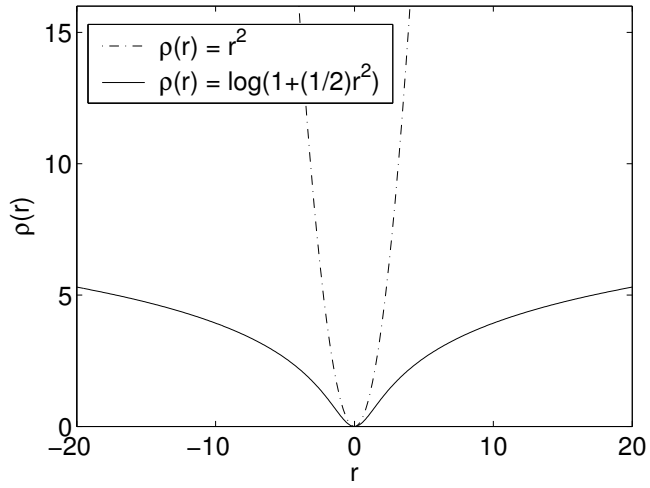


Figure 6.3: Comparison between least squares and Cauchy error measures.

error measure penalizes template pixels that are far from image edge pixels less than the least squares error measure does. This can be seen from Figure 6.3 which shows both the least squares error measure and the Cauchy error measure. This property of the Cauchy error measure causes the error function to ignore the occluded portions of the object being tracked.

6.3 Robustness to Spurious Edges

When tracking objects in real world scenes there are often edges present that are not from the object being tracked. These spurious edges can draw the template away from the object of interest. In this section, a modified Canny edge operator is presented that only preserves edges from the object being tracked by using information known about the object. Figure 6.4 illustrates the problem. Figure 6.4(a) shows an image containing the object to be tracked (the upper left gray rectangle). All of the edges in the original image are shown in Figure 6.4(b) and Figure 6.4(c) shows only the edges from the object that is to be tracked. In order to obtain the edge image shown in Figure 6.4(c) it is necessary to probe the pixel values on either side of the edges shown in Figure 6.4(b) to determine of the edge in question bounds the object being tracked. For example, in Figure 6.4, if an edge has the dark gray color of the rectangle being tracked on one side, then the edge should be preserved. In this section a modification to the Canny operator that suppresses unwanted edges is presented.

The Canny edge detector uses the image gradient and the continuity of edges to determine whether an edge candidate should be accepted or rejected. This algorithm works well to find general edges, however, it does not consider information that may be available about the object being tracked. By using some information about the object of interest, the edges in the scene that are not from this object can be removed. The first step is to perform a Canny edge detection operation on the image. The ith edge pixel found by the Canny operator is labeled w_i. For each edge pixel, w_i, samples are taken in the positive direction of the edge normal and in the negative direction of the edge normal. The edge normal $\mathbf{n}_i$ is defined by:

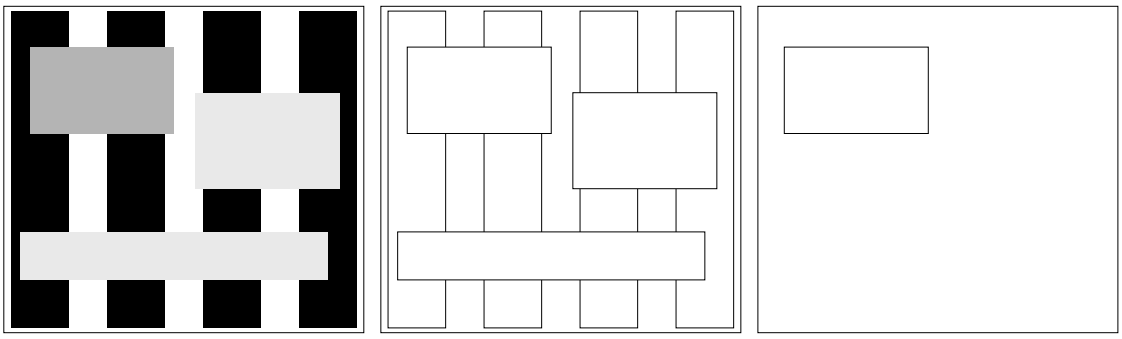
$$\mathbf{n}_i = \frac{\nabla \mathbf{I}_i}{\|\nabla \mathbf{I}_i\|} \tag{6.3}$$

where $\nabla \mathbf{I}_i$ is the gradient of the image at the location of edge vertex w_i. The edge normal sample vector in the positive direction is referred to as $\mathbf{S}_i^+$ and in the negative direction is referred to as $\mathbf{S}_i^-$. Figure 6.5 illustrates the edge normal samples for a representative edge pixel w_i. The samples are computed using bilinear interpolation and the samples are spaced one pixel apart. The number of samples taken in each direction depends on the application. Five samples are used for the results presented in this chapter.

Once the edge normal sample vectors have been obtained, they are used to determine whether the edge pixel should be accepted or rejected. This is a classification problem. Let f be the classifier function where f is defined so that an edge is accepted if $f\left(\mathbf{S}_i^+, \mathbf{S}_i^-, \omega\right) = 1$ and is otherwise rejected. The vector ω represents the parameters that define the classifier. Two forms for f are used in this chapter: one compares the edge normal sample vector to the known intensity of the interior of the object being tracked and the other uses a neural network model to evaluate whether an edge normal sample vector represents the desired object or not.

6.3.1 Interior Intensity Based Classifier

The first approach to classification compares the edge normal sample vectors, $\mathbf{S}_i^+$ and $\mathbf{S}_i^-$, to the known intensity $\mathbf{C}$ of the interior of the object. The error between the edge normal sample vectors and the interior intensity is computed for each edge pixel in the image. If the error is below a threshold value, H, then the edge pixel is accepted. The classifier f is



(a) (b) (c)

Figure 6.4: Edge detection in a cluttered scene: (a) original scene, (b) edges in the scene, and (c) edges from the object of interest.

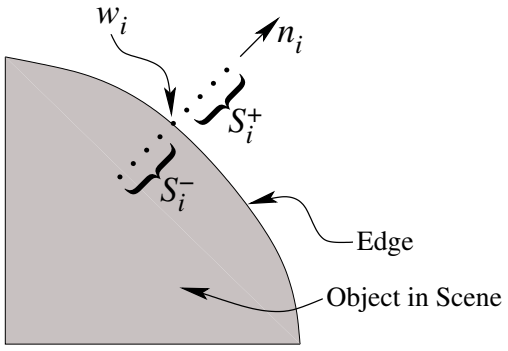
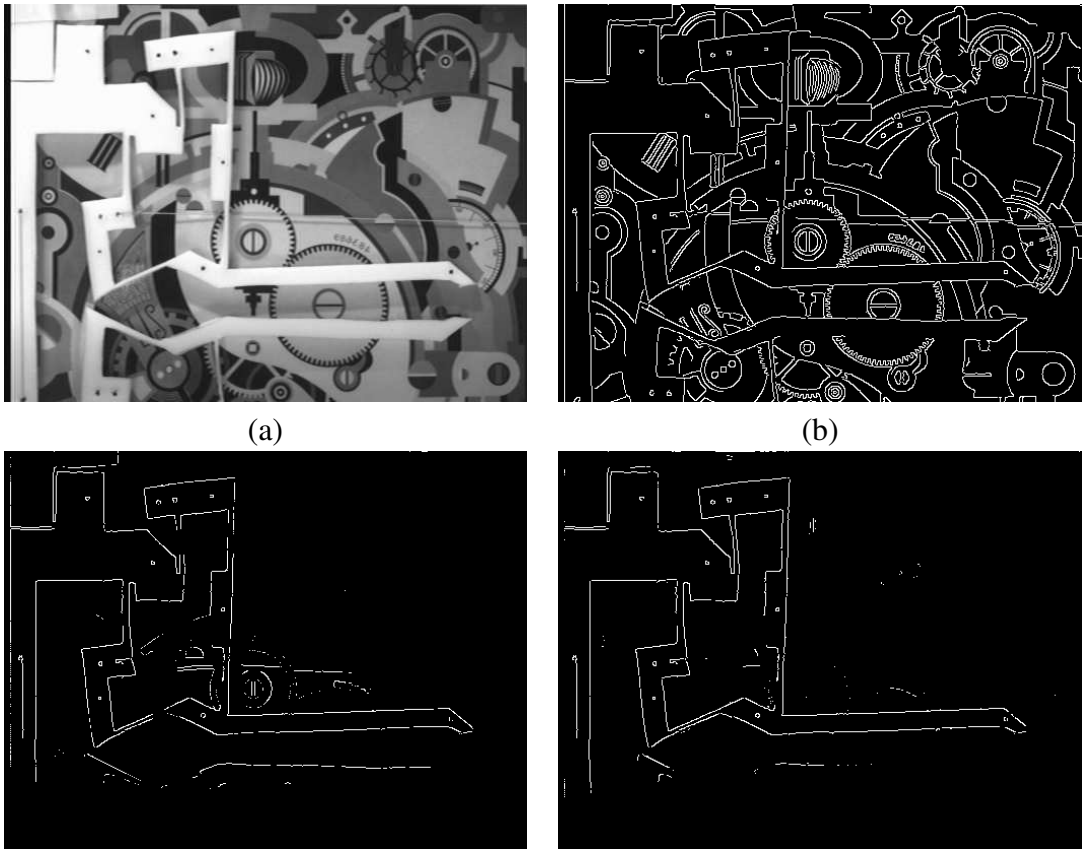


Figure 6.5: Edge samples used to test whether or not to keep an edge pixel.



(a) (b)

(c) (d)

Figure 6.6: Comparison of edge detection techniques discussed in the text. (a) The original image, (b) Canny edge detection algorithm, (c) edge detection example using only object intensity to classify edges, and (d) edge detection example using the neural network edge classifier.

defined as

$$f\left(\mathbf{S}_i^+, \mathbf{S}_i^-, \omega\right) = \begin{cases} 1 & \text{If} \quad \left\|\mathbf{S}_i^+ - \mathbf{C}\right\| < H \\ & \text{or} \quad \left\|\mathbf{S}_i^- - \mathbf{C}\right\| < H \\ 0 & \text{Otherwise} \end{cases} \tag{6.4}$$

For this classifier, the parameter vector ω is defined to be the interior intensity of the object and the threshold: $\{\mathbf{C}, H\}$.

An example of this method applied to an image of the compliant gripper is shown in Figure 6.6(c). The original Canny edges are shown in Figure 6.6(b). It can be seen from Figure 6.6(c) that using the interior intensity of the object to classify edges does not eliminate all spurious edges (false positives) and some of edges from the object being tracked are lost.

6.3.2 Neural Network Classification Method

It was shown above that using the object's interior intensity to classify edge candidates leads to false positives and to the loss of some of the edges from the object being tracked. To create a more robust classifier algorithm it is necessary to define a more general classifier than one simply based the threshold of an error measure. A neural network classifier is used in order to decrease the number of false positives. The neural network takes as input the edge normal samples and outputs a value between 0 and 1. If the output is greater than 0.5, the edge is accepted, otherwise the edge is rejected. The classifier f becomes

$$f\left(\mathbf{S}_i^+, \mathbf{S}_i^-, \omega\right) = \begin{cases} 1 & \text{If} \quad NeuralNet(\mathbf{S}_i^+) > 0.5 \\ & \text{or} \quad NeuralNet(\mathbf{S}_i^-) > 0.5 \\ 0 & \text{Otherwise} \end{cases} \tag{6.5}$$

For the neural network edge classifier, the parameter vector ω represents the weights of the trained neural network model. The neural network is a standard feed-forward neural network trained by error back propagation. In order to train the neural network, training pairs need to be obtained. The training pairs are obtained from representative images of the scene. From the training images, the positive and negative edge normal sample vectors are calculated for all of its edge pixels. The edge normal sample vectors that correspond with the interior of the object of interest are paired with a neural network output of 1.0 and the normal sample vectors that do not correspond to the interior of the object of interest are paired with the neural network output of 0.0. These training pairs are used to train the

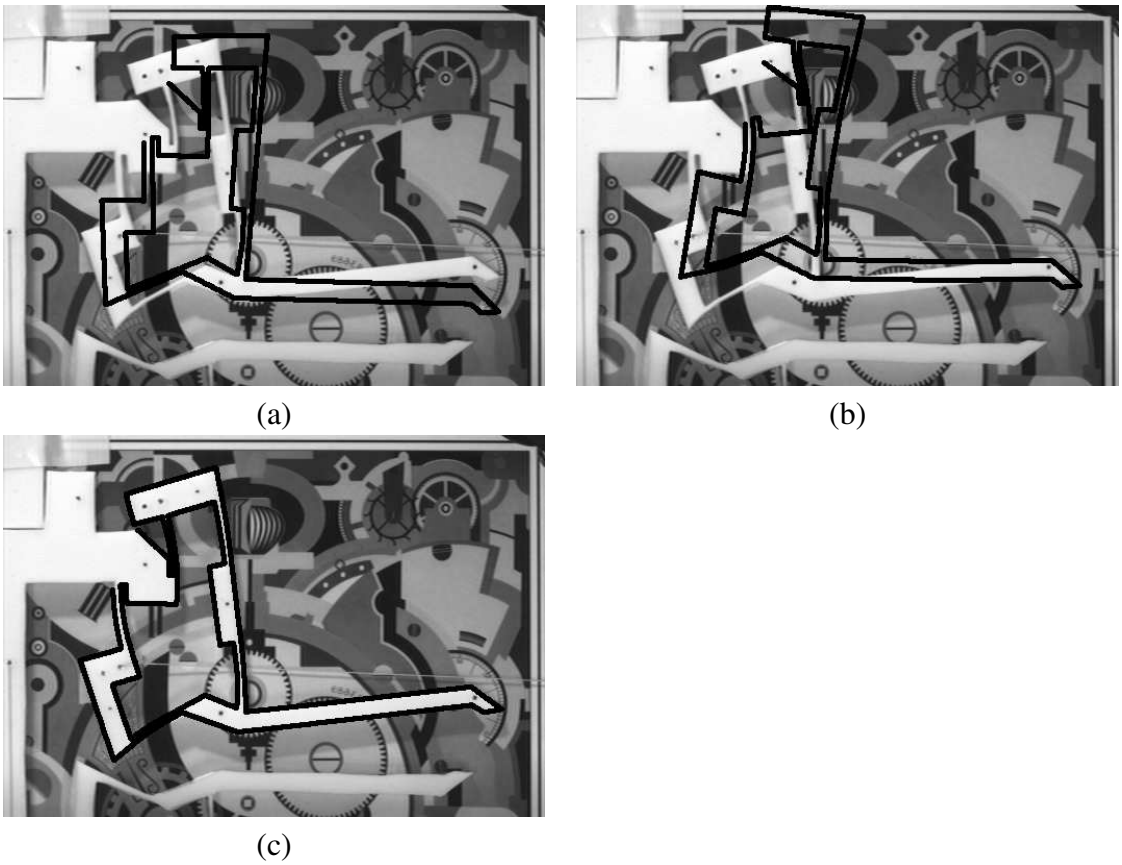


Figure 6.7: Effect of edge detection algorithm on tracking results for a cluttered scene. (a) The initial template shape and location for both methods, (b) the tracking solution when using the Canny edge detector, and (c) the tracking result when using the neural network edge classifier.

neural network model. For the results presented here, the neural network model has 10 hidden nodes and 6180 training pairs were used (2060 sample vectors corresponding to an actual edge and 4120 samples corresponding to a false edge).

Figure 6.6(d) shows the performance of the neural network classification scheme applied to a test image (this image was not used in the training of the neural network). It can be seen that the neural network edge classifier eliminates almost all of the spurious edges without losing any of the edges from the object of interest.

The elimination of spurious edges greatly improves the robustness of the deformable object tracking algorithm. Figure 6.7(c) shows the successful tracking solution for a scene with a large number of spurious edges.

Table 6.1: Quantitive measure of the performance of the algorithms introduced in this chapter. Error values are measured in mm.

	Control Image Set	Occlusion Image Set	Spurious Edge Image Set
Least Squares Error - Canny (LSE-C)	0.249	44.011	9.2814
Cauchy Error - Canny (CE-C)		0.270	
Least Squares Error - Canny + Neural Net (LSE-C+NN)			0.293

This method for the suppression of spurious does have the drawback of increased computation time over the standard Canny edge detection algorithm. For the image shown in Figure 6.6(a), the Canny edge detector takes approximately 0.02 seconds to process the whole 640x480 image. The use of the neural network edge classifier to suppress the spurious edges adds approximately 0.75 seconds to the computation time (these times were obtained using a 2.66 GHz Intel processor). The edge suppression algorithm spends most of its computation time interpolating the image to obtain the sample vectors, $\mathbf{S}_i^+$ and $\mathbf{S}_i^-$. The neural network edge classifier could be made to run in real-time by optimizing the interpolation routine. One option for speeding up the interpolation calculations would be to make use of the image interpolation functionality built into modern graphics hardware.

6.4 Performance Analysis

The performance of the above algorithms was evaluated quantitively in order to measure how much the tracking algorithm is improved by the modifications that have been presented. Five fiducial marks where placed on the gripper. These marks were used to measure the quality of the tracking solution by measuring the error between the tracking algorithm's predicted location of the marks and the actual location of the marks. Table 6.1 summarizes the results that were obtained. The columns represent the three different image sequences that were used. The first set is a control set, the second set introduces occlusion, and the third set introduces spurious edges (see Figures 6.1, 6.2, and 6.7, respectively, for example images from the sets). The rows indicate the error measure and edge detector used. The table entries show the average error between the template fiducial marks and the image

fiducial marks and are measured in the units of mm.

It can be seen from the table that the performance of the least squares error measure and Canny edge detector algorithm (LSE-C) is not satisfactory for the occlusion and spurious edge image sets. For the occlusion image set, the Cauchy error measure and Canny edge detector algorithm (CE-C) has performance that is nearly as good as the LSE-C algorithm for the control image set. For the spurious edge image set, the least squares error measure and neural network edge classifier algorithm (LSE-C+NN) again performs almost as well as the LSE-C algorithm does for the control set.

6.5 Summary and Conclusions

A deformable object tracking algorithm that is robust to occlusion and to spurious edges was presented. A robust error measure was used to handle the problem of occlusion and a modification of the Canny edge operator was used to eliminate spurious edges. The enhanced performance resulting from these modifications was demonstrated by tracking a four degree-of-freedom compliant gripper.

Chapter 7

Vision-Based Force Measurement[1]

Microassembly is becoming increasingly important, because it enables the creation of MEMS devices with greater functionality. Through an assembly process, MEMS devices can be created with three dimensional features and can consist of structures created by incompatible microfabrication processes. Force sensing is important for microassembly because the objects involved are often fragile. Force sensing is also essential for biomanipulation where the biological cells and tissues being handled are easily damaged. Currently, force measurement at the micro scale is usually done using laser-based optical techniques [35] or using piezoresistive material embedded in an elastic part [49]. Both of the methods are difficult to implement because they require a specially designed elastic part. For example, a laser-based optical force sensor requires precise alignment of laser optics with respect to the elastic part, and a piezoresistive force sensor requires that a piezoresistive layer be embedded within the part during its manufacture. Vision-based force measurement (VBFM) has the advantage that it can be used with existing elastic parts. It also has the advantage that it makes use of the microscope optics and cameras that are already present in a micromanipulation or biomanipulation workstation.

This chapter describes a deformable template matching approach that is used to recover the force applied to an elastic object, where the template deforms according the governing equations of elasticity. The deformable template registers to a Canny edge image [5] of a deformed object using an error minimization approach.

This chapter is organized as follows. Section 7.1 formulates the elasticity problem and concludes by showing how the Dirichlet to Neumann map can be used to recover the force

[1] ©2004 IEEE. Reprinted, with permission, from "Vision-based force measurement," Greminger, M.A., Nelson, B.J., IEEE Transactions on Pattern Analysis and Machine Intelligence, no. 3, vol. 26, 2004.

distribution applied to an elastic object using the displacement field of the contour. The problem of recovering the displacement field of the contour of an elastic object is discussed in Section 7.2, where it is shown that the recovery of the contour displacement field can be solved using deformable templates. Section 7.3 formulates the deformable template matching algorithm for a cantilever beam. The application of VBFM to a micro-gripper device is discussed in Section 7.4. A concluding discussion is given in Section 7.6.

7.1 Linear Elasticity Theory

In this chapter only deformable objects that are linearly elastic are considered. Linear elasticity theory assumes that the object's strains are infinitesimal and that the object's stress-strain relationship is linear. Both of these assumptions are satisfied for the materials to be considered here: silicon and steel. Both silicon and steel exhibit a linear stress-strain relationship for a large working range. It is also assumed the object is in a state of two dimensional stress referred to as plane stress.

7.1.1 The Formulation of the Plane Stress Elasticity Problem

The plane stress assumption assumes a state of stress where there is no stress in the x_3 direction of an object. Therefore, the stress components τ_{13}, τ_{23}, and τ_{33} will have a value of zero (see Figure 7.1). The remaining stress components, τ_{11}, τ_{22}, and τ_{12}, are functions of x_1 and x_2 only. Consider the bounded two dimensional domain R shown in Figure 7.2. It is assumed that R is defined such that the divergence theorem applies. Two dimensional domains where the divergence theorem applies are those that are bounded by a finite number of piece-wise smooth curves [18] and are known as normal domains. The elastic body defined by R is governed by the equations of elasticity which are simplified for the plane stress case. The equations of elasticity can be expressed in terms of displacements $u(x)$ by [43]:

$$\mu \nabla^2 u_\alpha + \mu \left(\frac{1+\nu}{1-\nu} \right) \frac{\partial}{\partial x_\alpha} \left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} \right) + F_\alpha = 0 \tag{7.1}$$

where $\alpha = 1, 2$, μ is the shear modulus, and ν is the Poisson's ratio. The material properties of an isotropic, elastic material are completely defined by the shear modulus and the Poisson's ratio.

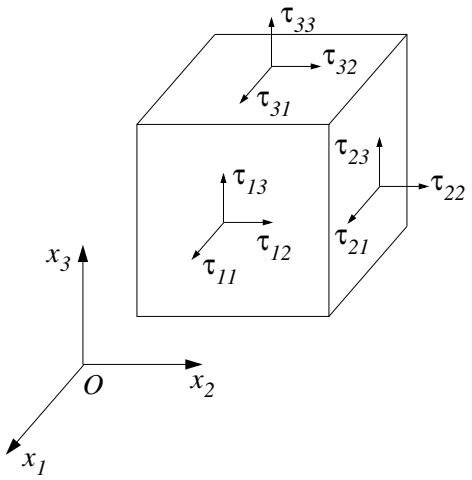


Figure 7.1: Cube in three dimensional state of stress.

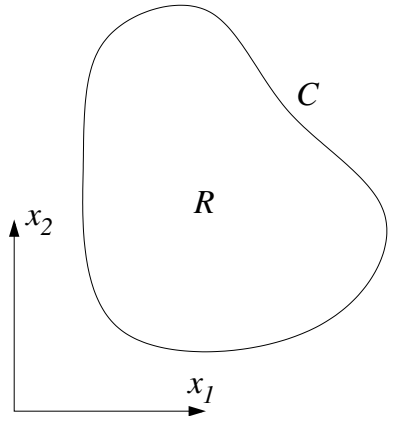


Figure 7.2: Elastic body R and its associated contour C.

The boundary conditions for the elasticity problem can be expressed as a prescribed displacement vector f_i on the contour C, known as a Dirichlet boundary condition:

$$u_i|_C = f_i \tag{7.2}$$

or a prescribed traction vector T_i on C, known as a Neumann boundary condition:

$$\tau_{ij} n_j|_C = T_i \tag{7.3}$$

where n_j is the outward unit normal of C and the stress tensor τ_{ij} can be expressed in terms of displacements by [43]:

$$\tau_{11} = \frac{2\lambda\mu}{\lambda + 2\mu}\left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right) + 2\mu\frac{\partial u_1}{\partial x_1} \tag{7.4}$$

$$\tau_{22} = \frac{2\lambda\mu}{\lambda + 2\mu}\left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right) + 2\mu\frac{\partial u_2}{\partial x_2} \tag{7.5}$$

$$\tau_{12} = \mu\left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}\right) \tag{7.6}$$

$$\tau_{13} = \tau_{23} = \tau_{33} = 0 \tag{7.7}$$

The traction vector is the force per unit length applied to the contour of the object. It is assumed throughout this discussion that f_i and T_i have piecewise continuous derivatives.

7.1.2 The Dirichlet to Neumann Map

The Dirichlet to Neumann map Λ [34][47] is a mapping from the surface displacements f_i to the surface tractions T_i and can be expressed as:

$$\Lambda(f_i) = \tau_{ij} n_j|_C = T_i \tag{7.8}$$

In order for Λ to be defined it is necessary that for each f_i there exists a unique T_i. The existence and uniqueness theorems of linear elasticity are sufficient to show that the Dirichlet to Neumann map does exist. The uniqueness theorem for the Dirichlet plane stress problem is due to Kirchoff [27]. Kirchoff's proof shows that the solution to the Dirichlet problem is unique as long as the following conditions on the shear modulus and Poisson's ratio are

satisfied:

$$\mu \neq 0 \qquad -1 < \nu < \frac{1}{2} \tag{7.9}$$

which are satisfied for the materials being considered. The existence of solutions for the Dirichlet plane stress problem can also be shown [43]. The existence of the Dirichlet to Neumann map shows that the traction distribution on the contour of a linearly elastic body can be uniquely determined from the displacement field of its contour.

In general, the existence of the Dirichlet to Neumann map cannot be proven for nonlinear problems. However, the concept of the Dirichlet to Neumann map can be applied to nonlinear problems where the mapping can often be computed using numerical modeling techniques.

7.2 Recovery of Boundary Displacement Field From Contour Data

The vision-based force measurement problem is reduced to that of finding the displacement field of the contour of an object. Figure 7.3 shows an undeformed contour C_1 along with a deformed contour C_2. In general, the problem of determining the displacement field that leads to the deformed contour C_2 does not have a unique solution. Figure 7.4 illustrates this point by showing how two distinct displacement fields can lead to two identical deformed contours, C_2 and C_3. Three points, P_1, P_2, and P_3, are shown on the undeformed contour C_1 as well as on the deformed contours C_2 and C_3. Experience with elastic objects suggests that the displacement field that created the contour C_2 is the correct one, however, both displacement fields are equally valid.

In order to find an unique displacement field for a given deformed contour it is necessary to make some assumptions about the object that is being deformed. It is assumed that it behaves according to (7.1) and that the strain components are infinitesimal. With the assumption of infinitesimal strains it becomes clear that the displacement field that led to contour C_3 is not physically possible because it would require large strain values. Since the strains are assumed to be small, the deformed contour will not differ much from the undeformed contour. With these assumptions in mind, a logical way to recover the displacement field is to perturb the undeformed contour C_1 (see Figure 7.5) by small amounts until it matches the deformed contour C_2. The undeformed contour will be perturbed by

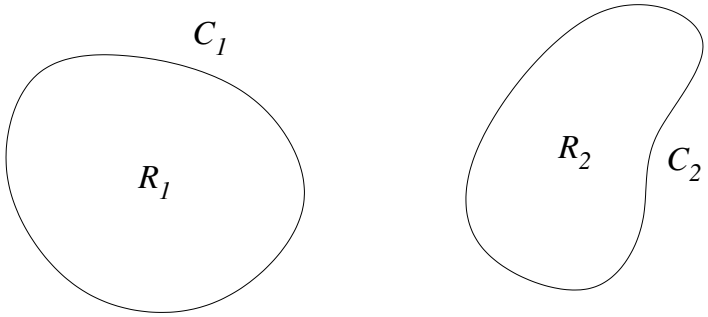


Figure 7.3: Undeformed contour C_1 and deformed contour C_2.

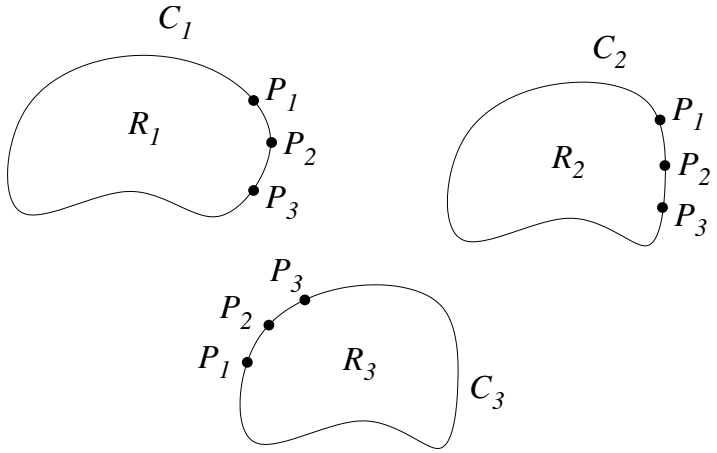


Figure 7.4: Illustration of two distinct displacement fields that lead to identical deformed contours, C_2 and C_3.

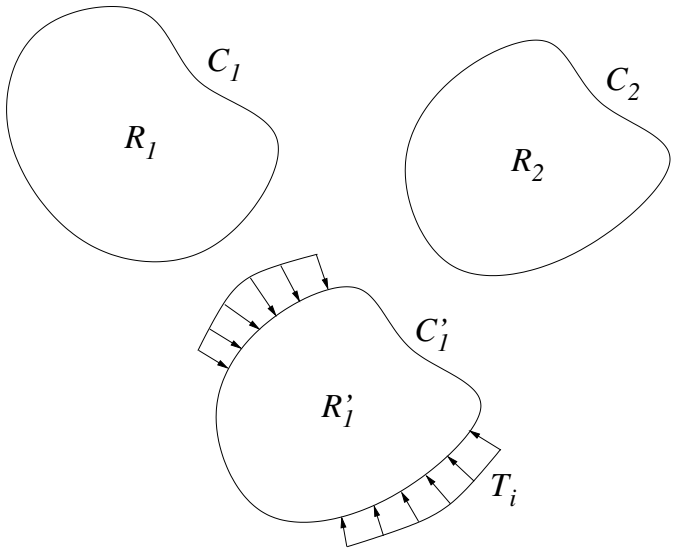


Figure 7.5: The contour C_1 is perturbed by the traction distribution T_i so that it matches the deformed contour C_2.

assuming a traction distribution T_i and then deforming the contour according to (7.1). Figure 7.5 shows a perturbed contour C_1', with a traction distribution T_i applied to it, which matches the deformed contour C_2.

The perturbation approach just described can be implemented with a deformable template matching algorithm where the undeformed template is the undeformed contour of the object, C_1, and the template is perturbed by a traction distribution T_i according to the governing equations of elasticity to obtain the deformed contour C_1'. If the elastic model used to deform the template accurately models the elastic object shown in the image, the solution to the Dirichlet to Neumann Map is given by the traction distribution T_i applied to deform the template. Therefore, the Dirichlet to Neumann map can be evaluated for a particular deformed object through the use of deformable template matching provided that the template is deformed according the equations of elasticity. The following section demonstrates how this deformable template matching technique can be applied to a cantilever beam.

7.3 Force Recovery with Deformable Templates

As mentioned above, the forces can be recovered by perturbing a deformable template. The first object tested with this approach was a silicon cantilever beam 450 μm long. An image

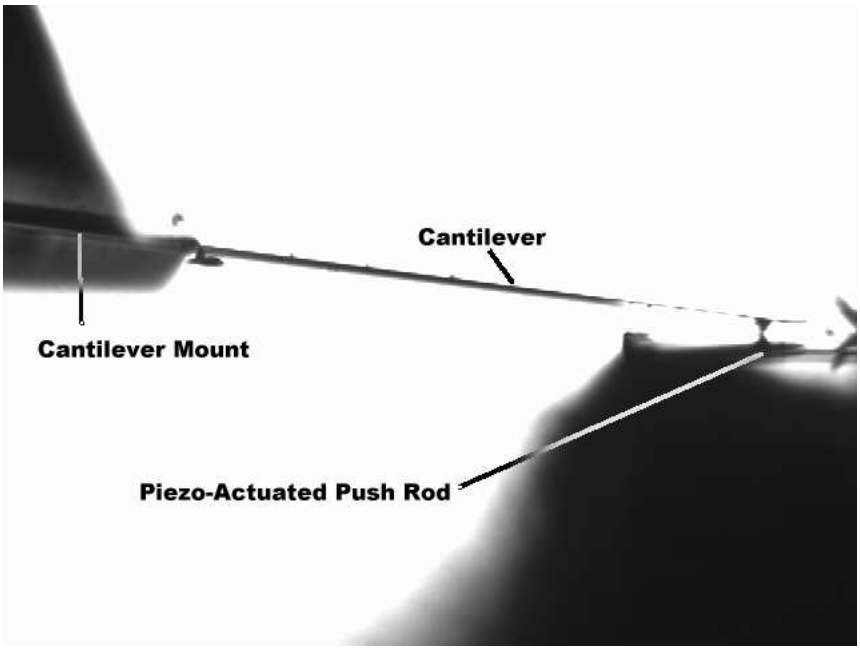


Figure 7.6: Image of micro-cantilever beam.

of the cantilever beam is shown in Figure 7.6. The beam was manufactured using standard microfabrication techniques and is of a type used in atomic force microscopes.

7.3.1 Incorporating Force Into The Template Matching Algorithm

Minimizing (2.4) will determine the rigid body motion of the object. The nonrigid motion of the template will be modelled using (7.1). For the cantilever beam, the Bernoulli-Euler law is assumed to apply [43]. The Bernoulli-Euler law can be used to simplify (7.1) and leads to the following displacement solution for the cantilever beam shown in Figure 7.7(a):

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{Fr_1^2}{6EI}(3L - r_1) \end{bmatrix} \tag{7.10}$$

where F is the force being applied to the cantilever, L is the length of the cantilever, E is the modulus of elasticity of the cantilever, and I is the moment of inertia of the cantilever's cross section.

Next, it is necessary to apply this displacement field to the cantilever template. It can be seen from (7.10) that the x_1 component of the template points will remain undeformed. It should also be noted that if a template point's x_1 component is less than zero, then the

point will not be transformed. If a template point's x_1 component is between zero and L, then the point will be translated in the x_2 direction by u_2 in (7.10). For points with an x_1 component greater than L, they will be translated in the x_2 direction, but the equation will be different from (7.10) because the radius of curvature of the cantilever becomes infinite for $r_1 > L$. Once all of the template points are translated appropriately, then the affine transform (2.3) can be applied to the new template points to add the rigid body motion. The equations used to obtained the transformed template pixels are

$$r' = A(r) \quad \text{for} \quad (r_1 < 0) \tag{7.11}$$

$$r' = A\left(\begin{bmatrix} r_1 \\ r_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{Fr_1^2}{6EI}(3L - r_1) \end{bmatrix}\right) \quad \text{for} \quad (0 \leq r_1 < L) \tag{7.12}$$

$$r' = A\left(\begin{bmatrix} r_1 \\ r_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{FL^2}{6EI}(3r_1 - L) \end{bmatrix}\right) \quad \text{for} \quad (L \leq r_1) \tag{7.13}$$

where $A(r)$ is an affine transform defined in (2.3).

Using the template transformations (7.11) through (7.13), the template matching error function (2.7) becomes a function of one additional variable, the force F applied to the template. This is shown by the following error function:

$$E(\theta, X, F) = \sum_{i=1}^{N} \|r^{i'} - w^i\|^2 \tag{7.14}$$

When this error function is minimized the force F that is being applied to the cantilever is obtained. Figure 7.7(b) shows the undeformed cantilever template that was obtained from a Canny edge image of the cantilever. Figure 7.7(c) shows a deformed template along with the undeformed template. The left half of Figure 7.8 shows an image of a deflected cantilever beam and the right half of this figure shows a cantilever template matched to the image.

7.3.2 Experimental Setup

A diagram of the experimental setup used to measure the force applied to the cantilever beam is shown in Figure 7.9. The cantilever beam is a 450 μm long AFM probe tip with a spring constant of approximately 0.1 N/m. A known displacement is applied to

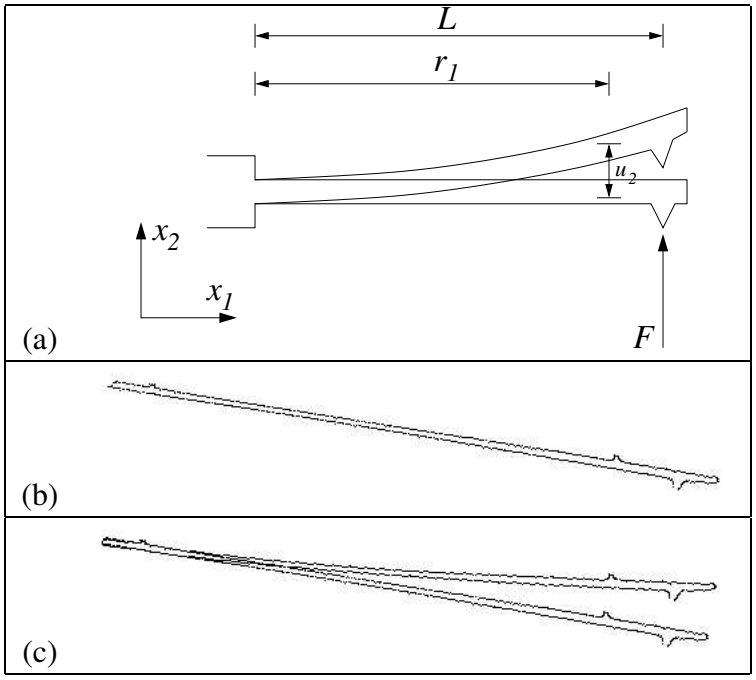


Figure 7.7: Deformable cantilever template.

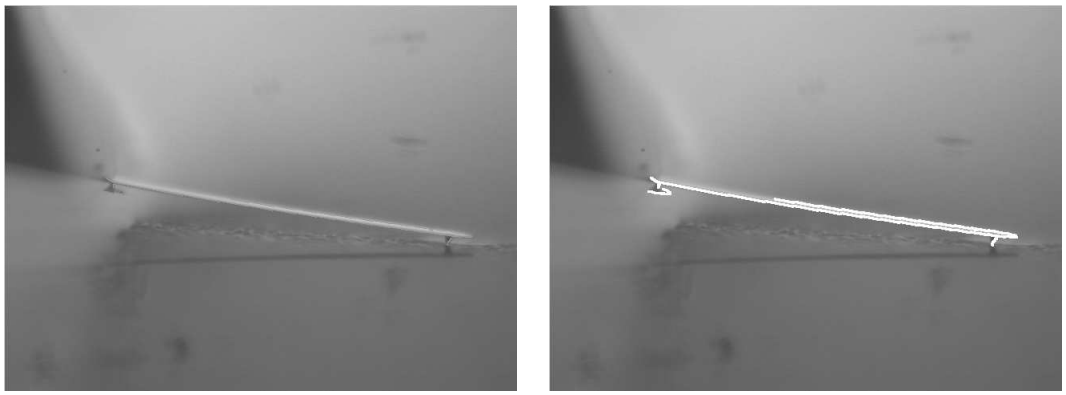


Figure 7.8: Deflected cantilever is on the left and the template matched to the deflected cantilever is on right.

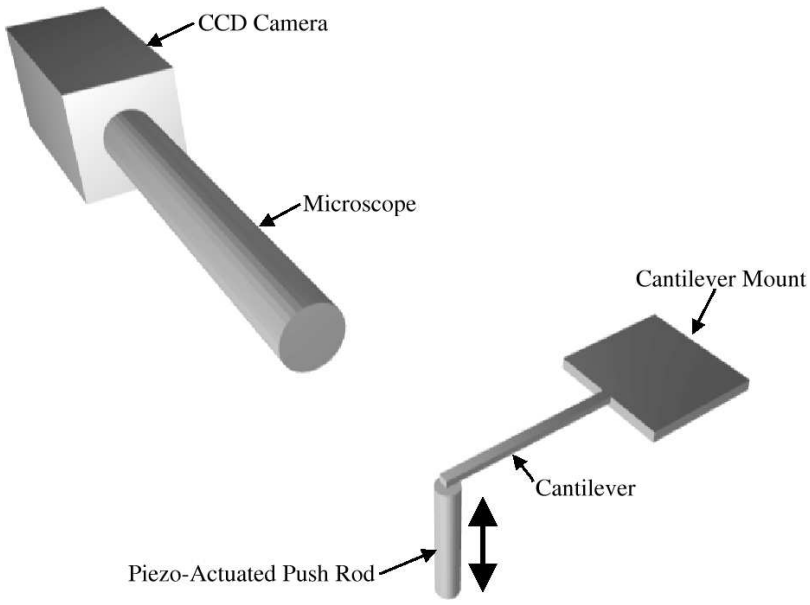


Figure 7.9: Vision-based force measurement experimental setup.

the cantilever beam using a 3 DOF piezo-actuated nanopositioner from Queensgate with sub-nanometer positioning resolution. A microscope mounted with a CCD camera is used to obtain an image of the cantilever. A video capture card digitizes the CCD image. An image captured from the CCD camera is shown in Figure 7.6.

7.3.3 Cantilever Results

To evaluate the performance of the VBFM algorithm, a known displacement was applied to the cantilever beam using a Queensgate nanopositioner. These displacement inputs were used to calibrate the force measurement system. Figure 7.10 shows a force versus applied displacement plot. The 1 σ confidence intervals are shown on the plot. The maximum 1 σ confidence interval for the force measurements is +/- 2.85 nN. The system was tested with both a 10x and a 20x objective lens. Table 7.1 summarizes the performance of the force sensor with both the 10x objective lens and the 20x objective lens.

Table 7.1: Summary of results for visual force sensor applied to a cantilever beam.

Magnification	NA	Rayleigh Limit (μm)	Pixel Size (μm)	Deflection Error (nm)	Force Error (nN)
10x	.28	1.000	1.050	1σ 88.7	8.87
				2σ 180.5	18.05
				3σ 276.5	27.65
20x	.42	.700	.530	1σ 28.5	2.85
				2σ 58.0	5.80
				3σ 88.7	8.87

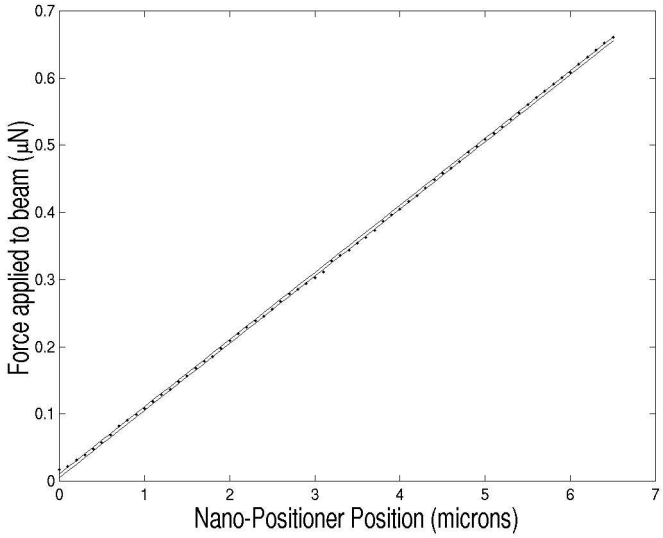


Figure 7.10: Calibration plot for cantilever force sensor with 20x objective lens.

7.4 Vision-Based Force Measurement Applied to a Micro-Gripper

The methods presented in this chapter can be applied to more complex objects. In this section, vision-based force measurement will be applied to a micro-gripper. The top of Figure 7.11 shows a micro-gripper developed by Yang et al. which is used for a specific microassembly task. The gripper is cut by micro-wire electro-discharge machining (EDM) from spring steel 254 μm thick. An object is grasped by sliding a tube over the gripper to force the jaws of the gripper together. When this gripper is used for an assembly task it is important to measure the gripping force that the gripper applies to the object being held in order to ensure a stable grasp is achieved and to avoid damage to the object being grasped.

7.4.1 Modeling of the Micro-Gripper

Each jaw of the gripper can be modeled as a cantilever beam. The bottom of Figure 7.11 shows the deflection model parameters, where D is the displacement of the jaw due to the clamping tube making contact with the gripper, F is the force applied to the object being held and u is the displacement of the jaw at a position x along its length. D is a function of the clamping tube position, L_2, and can be written as:

$$D = L_2 \sin\beta - r\cos\beta \tag{7.15}$$

where r is the inner radius of the clamping tube and β is defined in Figure 7.11. Because the tube forces the gripper jaw to deflect there is a reaction force R applied to the tube by the jaw. In order to solve for u it is first necessary to solve for this reaction force. The displacement D applied to the gripper acts as a constraint applied to the cantilever beam. Given this constraint, the reaction force R can be solved for by using the beam equation with the constraint that the displacement must be of magnitude D at the location where the tube comes into contact with the gripper jaw. Using this approach, R can be written as

$$R = \frac{3}{2}\frac{Fcos(\beta)^2 L_1}{L_2} - \frac{1}{2}Fcos(\beta) + 3\frac{EIcos(\beta)^3 sin(\beta)}{L_2^2} - 3\frac{EIcos(\beta)^4 r}{L_2^3} \tag{7.16}$$

Using the above equation for R, is straightforward to calculate the displacement field, u, of the jaw using the Bernoulli-Euler law. The equations for u are shown below. The form of

u depends on the value of x relative to the position of the reaction force R and the gripping force F.

$$u_1 = \frac{1}{6}\frac{Fcos\beta x^2(3L_1 - x)}{EI}d - \frac{1}{6}\left[\frac{1}{2L_2^3}\left(Fcos(\beta)^4 L_1^3\left(2 - 3\frac{L_1 - \frac{L_2}{cos\beta}}{L_1} + \frac{\left(L_1 - \frac{L_2}{cos\beta}\right)^3}{L_1^3}\right)\right) + 3\frac{EIcos(\beta)^3(L_2 sin\beta - rcos\beta)}{L_3^3}\right] x^2\frac{\left(3\frac{L_2}{cos\beta} - x\right)}{EI} \tag{7.17}$$

$$u_2 = \frac{1}{6}\frac{Fcos\beta x^2(3L_1 - x)}{EI} - \frac{1}{6}\left[\frac{1}{2L_2^3}\left(Fcos(\beta)^4 L_1^3\left(2 - 3\frac{L_1 - \frac{L_2}{cos\beta}}{L_1} + \frac{\left(L_1 - \frac{L_2}{cos\beta}\right)^3}{L_1^3}\right)\right) + 3\frac{EIcos(\beta)^3(L_2 sin\beta - rcos\beta)}{L_2^3}\right] L_2^2\frac{\left(3x - \frac{L_2}{cos\beta}\right)}{cos(\beta)^2 EI} \tag{7.18}$$

$$u_3 = \frac{1}{6}\frac{Fcos\beta L_1^2(3x - L_1)}{EI} - \frac{1}{6}\left[\frac{1}{2L_3^3}\left(Fcos(\beta)^4 L_1^3\left(2 - 3\frac{L_1 - \frac{L_2}{cos\beta}}{L_1} + \frac{\left(L_1 - \frac{L_2}{cos\beta}\right)^3}{L_1^3}\right)\right) + 3\frac{EIcos(\beta)^3(L_2 sin\beta - rcos\beta)}{L_2^3}\right] L_3^2\frac{\left(3x - \frac{L_2}{cos\beta}\right)}{cos(\beta)^2 EI} \tag{7.19}$$

where u_1 is valid for $0 < x \leq L_2/(cos\beta)$, u_2 is valid for $L_2/(cos\beta) < x \leq L_1$, and u_3 is valid for $L_1 < x$. The same set of equations can be obtained for the bottom half of the microgripper modulo sign. Since the clamping force F and the position of the tube L_2 are the degrees of freedom of the above equations, the vision algorithm must solve for both L_2 and F. Therefore (7.14) becomes:

$$E(\theta, X, F, L_2) = \sum_{i=1}^{N} \|r^{i'} - w^i\|^2 \tag{7.20}$$

When this error function is minimized the force applied to the gripper F and the position of the clamping tube L_2 are found.

7.4.2 Micro-Gripper Results

The tweezer template was first tested on images created by ANSYS where the gripping force was known. Once such image is shown in Figure 7.11(b). The tweezer was modeled in ANSYS as a two dimensional object in plane stress with constant thickness. The clamping tube was modeled by applying a displacement boundary condition to the tweezer jaw of a magnitude given by (7.15). Instead of applying a force to the tweezer jaw where the object was being held, a part thickness was assumed that provided another displacement boundary condition at the tip of the jaw. When this finite element model is solved the reaction forces are given as a result. The reaction force where the object is being held is then taken as the gripping force. This process was repeated for a series of increasing clamping tube positions L_2 with the same part thickness (much like the situation where a part is being gripped by gradually sliding the clamping tube in order to increase the gripping force). When this series of images created by ANSYS is used as input to the vision algorithm, the vision algorithm should return the same gripping force that was calculated by the finite element model. Figure 7.12 shows the gripping force versus clamping tube position from both the ANSYS model and VBFM algorithm applied to the images generated by the ANSYS model.

Equations (7.17) - (7.19) can also be verified using the ANSYS model by solving for the clamping force F as a function of the part thickness and the clamping tube position (specifying a part thickness is in effect specifying the value of u at the point $x = L_1$, once these values are substituted into (7.17) - (7.19), F can easily be solved for in terms of the remaining variables). Figure 7.12 also shows this force value predicted by equations (7.17) - (7.19). It can be seen from the plot that ANSYS, the beam equations, and the vision algorithm (which uses the beam equations for its deflection model) correspond closely.

Next, the vision algorithm was tested on the actual gripper. A piezoresistive force sensor manufactured by SensorOne (model AE801) was used to validate the output of the vision-based force sensing algorithm. The top of Figure 7.13 shows the experimental setup. One jaw of the gripper applied force to the piezoresistive force sensor and the other applied force to a rigid screw. The 1 σ precision of the vision algorithm applied to the micro-gripper was found to be +/- 3.1 mN. The calibration error of the vision algorithm can be seen in the bottom of Figure 7.13, where the output of the vision algorithm, along with the output of the piezoresistive force sensor, is shown versus clamping tube position. The error in this figure is due to errors in the material properties that were assumed for the gripper template. This problem is not particular to vision-based force measurement. With piezoresistive force

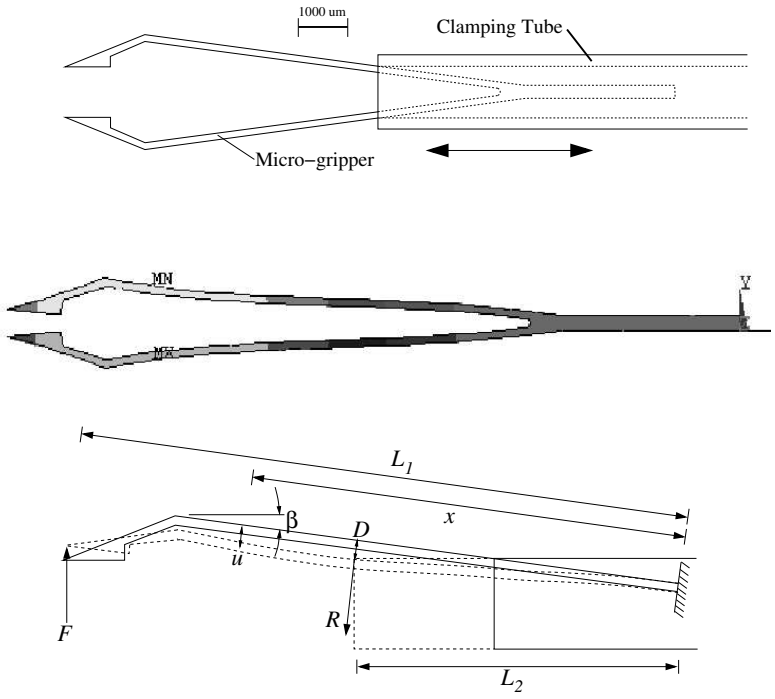


Figure 7.11: Micro-gripper manipulator on top, gripper ANSYS model center, and gripper jaw deflection model on bottom.

sensing and laser-based optical force sensing, material properties are also assumed for the object being deformed.

7.5 Conclusions

A method has been presented that reliably measures the force applied to an elastic object through the use of computer vision. It was shown that, through the application of the Dirichlet to Neumann map, the vision-based force measurement problem can be reduced to that of measuring the displacement field of the contour of a deformed object. This observation is important because it shows that boundary data is sufficient to completely recover the force applied to an linearly elastic object independent of object geometry. For nonlinear problems, the Dirichlet to Neumann map can often be modelled numerically and the force measurement problem can be solved using the approach presented in this chapter.

Microassembly and biomanipulation require force sensing for success. VBFM provides

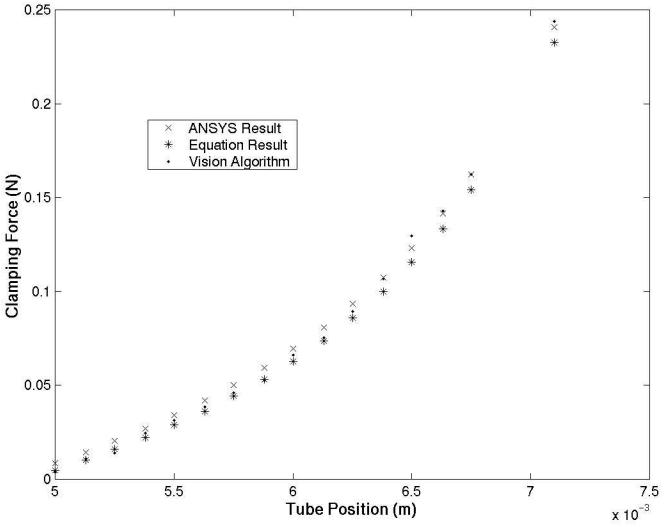


Figure 7.12: Forces obtained from the ANSYS simulation, beam equation and vision algorithm.

a means to measure forces by visually observing an elastic object. In the two examples given here, the cantilever and the micro-gripper, the objects were not initially designed to be used with VBFM, however, they were successfully used as force sensors with no alterations. For the cantilever, a sensor resolution of +/- 2.8 nN was achieved, while for the micro-gripper a resolution of +/- 3.1 mN was achieved. These specifications approach resolutions that are achievable with piezoresistive transducers.

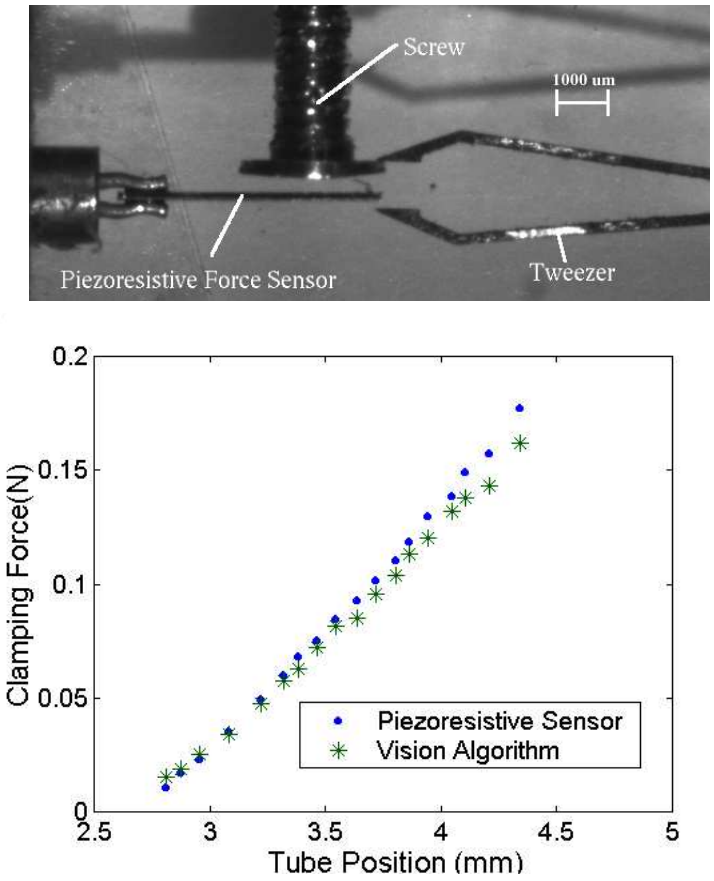


Figure 7.13: Microgripper experimental setup on top and micro-gripper results on bottom.

Chapter 8

Modeling Elastic Objects with Neural Networks for Vision-Based Force Measurement[1]

It was shown in the previous chapter that if an accurate model is available defining the elastic behavior of an object, then the force applied to that object can be measured using computer vision. Three cases exist where this explicit model based approach is not appropriate. The first case is for materials that exhibit nonlinear elastic properties where it is computationally prohibitive to evaluate the model in real-time. This situation occurs when there are large deflections or when the stress strain relationship for the material is nonlinear. The second case occurs when an accurate material model is unavailable for an object. Biological structures such as cells or organs are an example of a class of structures without accurate material models. The third case is when a model is available for the object, but the parameters that define the model, such as material properties or geometry, are not known or are known to a low certainty. In this case, the difficulty with the model based approach is in calibration.

For all three of these cases a neural network model approach to VBFM is preferable. Figure 8.1 illustrates the neural network approach to VBFM. For the neural network approach, a sequence of images is taken under various loading conditions. This sequence of images is then passed to the neural network training algorithm. Once the neural network

[1] ©2003 IEEE. Reprinted, with permission, from "Modeling elastic objects with neural networks for vision-based force measurement," Greminger, M.A., Nelson, B.J., IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003.

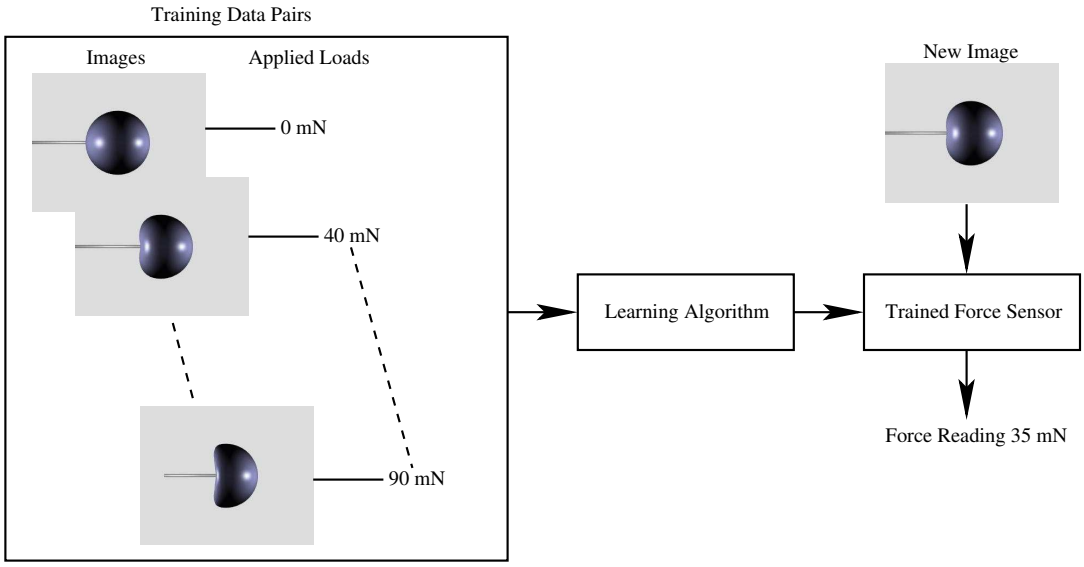


Figure 8.1: Diagram illustrating the learning approach to vision-based force measurement.

model is trained, it can then be used to measure forces from new images.

8.1 The Neural Network Elastic Material Model

A feed-forward two-layer neural network with the layout shown in Figure 8.2 is used. Each hidden node has a logistic sigmoid activation function of the form

$$g(a) = \frac{1}{1 + \exp(-a)} \tag{8.1}$$

where a is the sum of all of the weighted inputs to the node. The output nodes have a linear activation function that returns the sum of all of the node's weighted inputs. The neural network is represented in equation form by

$$y_i = w_{iM+1} + \sum_{j=1}^{P} \left[g\left(v_{jD+1} + \sum_{k=1}^{D} \left[v_{jk} x_k \right] \right) w_{ij} \right] \tag{8.2}$$

where the neural network has D inputs, P hidden nodes, and N outputs. The hidden layer weights are stored in the v matrix and the output layer weights are stored in the w matrix. Neural networks of this type have the property that they are universal approximators meaning that the neural network can approximate a general nonlinear function to arbitrary

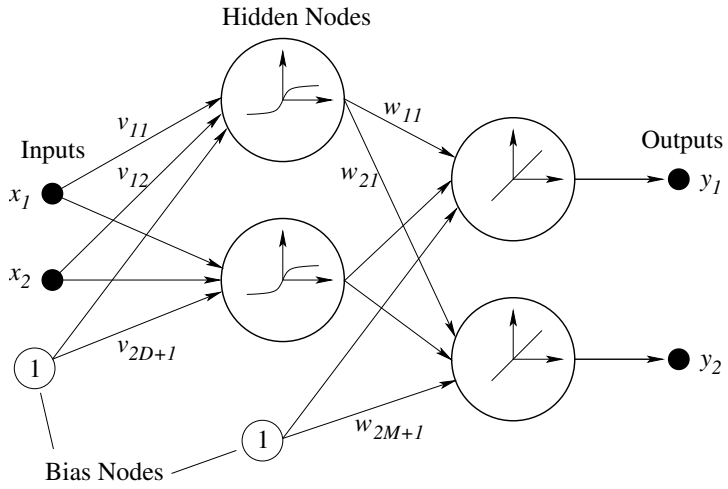


Figure 8.2: Neural network diagram.

accuracy provided that there are a sufficient number of hidden nodes [26].

The neural network elastic model presented in this chapter is structured so that given a point (x, y) in the object and the load F applied to the object, the neural network returns the deformed location (x', y') of the point. This process is shown Figure 8.3 for the two dimensional case. The neural network has three inputs and two outputs. If more than one load is being applied to the object, additional inputs can be added to the neural network. A neural network model constructed in this manner completely defines the deformation of an elastic object subject to an applied load F.

8.2 Neural Network Based Deformable Template Matching Algorithm

The deformable template will be deformed according to the neural network elastic model. The deformation model (2.5) becomes

$$u = D\left(r, \{t\}\right) = NeuralNetwork(r, F) \tag{8.3}$$

where $NeuralNetwork(r, F)$ represents the neural network material model. This model returns the displaced location of the template edge pixel r due to the applied force F. The error function (2.4) becomes

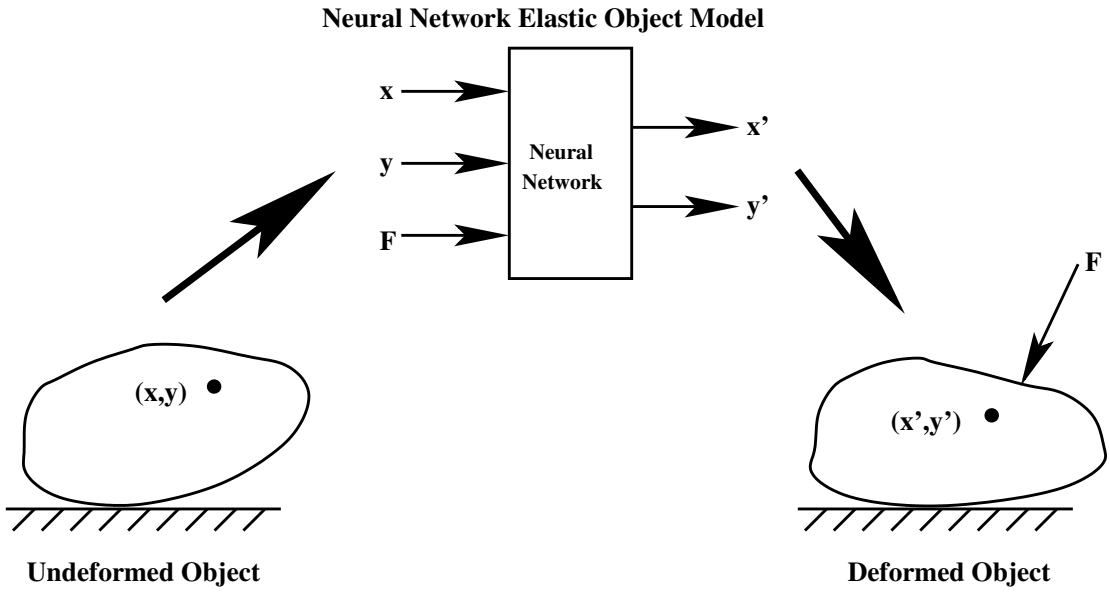


Figure 8.3: Process of using a neural network elastic model to apply a load F to an object.

$$E(\theta, X, F) = \sum_{i=1}^{M} \|r_i' - w_i\|^2 \tag{8.4}$$

Minimizing the above error function, in addition to giving the position and orientation of the object, gives the force F applied to the object. This is done by finding the applied force F that, when applied to the template, causes the template to match the image.

8.3 Acquisition of Training Data and Network Training

As previously shown, the neural network elastic model has three inputs and two outputs. The inputs are the x-y coordinates of a point in the undeformed object and the load applied to the object. The outputs are the x-y coordinates of the same point in the deformed object. Therefore, in order train the neural network it is necessary to obtain training pairs that consist of the x-y coordinate of a point in the undeformed object, the force applied to the object, and the x-y coordinate of the same point in the deformed object. Many such training pairs are needed to adequately train the neural network. Training pairs are obtained for the object subject to many loads representing the range of expected loads.

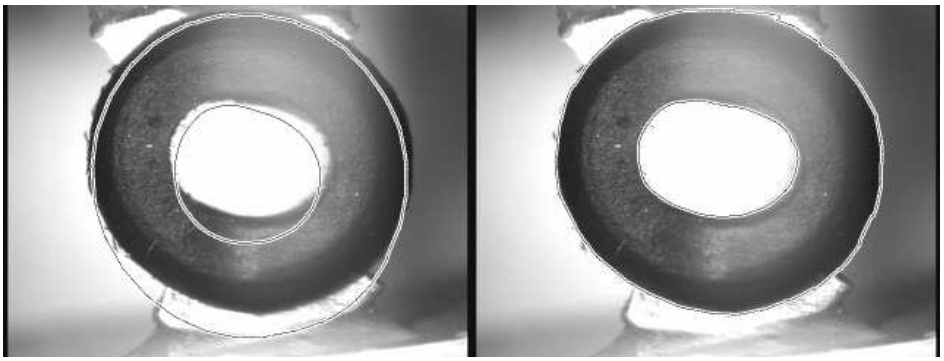


Figure 8.4: BEM deformable tracking algorithm tracking the rubber torus object. Undeformed template on left and template fitted to deformed torus on right.

8.3.1 Obtaining Training Data

The training data pairs are obtained directly from images of the object under known loads. By obtaining training data from images, a neural network model of the object is created without using a material model for the object.

The BEM deformable body tracking algorithm described in Chapter 5 is used to obtain the training data directly from images. The BEM tracking algorithm's ability to track general 2D deformations makes it well suited to this task. Images of the object under various loads are used for training.

The left half of Figure 8.4 shows a deformed rubber torus under a known load along with an undeformed BEM deformable template. The right half of this figure shows the BEM template matched to the deformed rubber torus. This process is repeated for all of the images used to train the neural network elastic model. The training images represent the range of loads that are likely to be encountered by the object. Figure 8.5 shows the training data obtained using the BEM tracking algorithm for the rubber torus object. In the figure, the same edge points are shown for the undeformed torus, the torus under a 0.613 N load and the torus under a 1.222 N load. When these point deformations are obtained for a sufficient number of different loads, they are used to train the neural network elastic model.

8.3.2 Training the Neural Network

The neural network is trained using the error back-propagation method [26]. The success of the training process depends on the number of training pairs that are used and the number

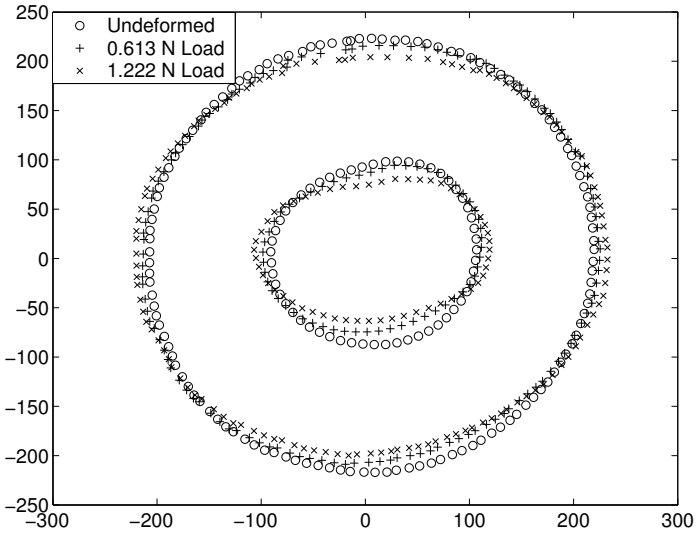


Figure 8.5: Training data obtained for the rubber torus object under two different applied loads.

of hidden nodes in the neural network model. If there are not a sufficient number of training data pairs, the network may model the training data well, but it will not be able to perform well for new loads that are not part of the training data. If there are not a sufficient number of hidden nodes, the neural network will not be able to model the elastic object to a high degree of accuracy. On the other hand, if there are too many hidden nodes, the neural network will be able to model the training pairs well but it may be very inaccurate for loads that are not part of the training data. This phenomenon is known as over-fitting [26]. In general, the more training pairs that can be used the more accurate the model will be. The number of hidden nodes used should be the minimum number that still achieves adequate training. The number of hidden nodes is increased until adequate traing is achieved.

8.4 Experimental Results

8.5 Rubber Torus Application

The neural network tracking algorithm was first applied to the rubber torus object shown in Figure 8.4 using the setup shown in Figure 8.6. The torus, because it is constructed of rubber, has nonlinear elastic behavior. Therefore, it is not computationally feasible to

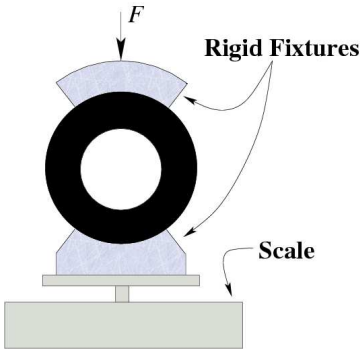


Figure 8.6: Rubber torus loading condition.

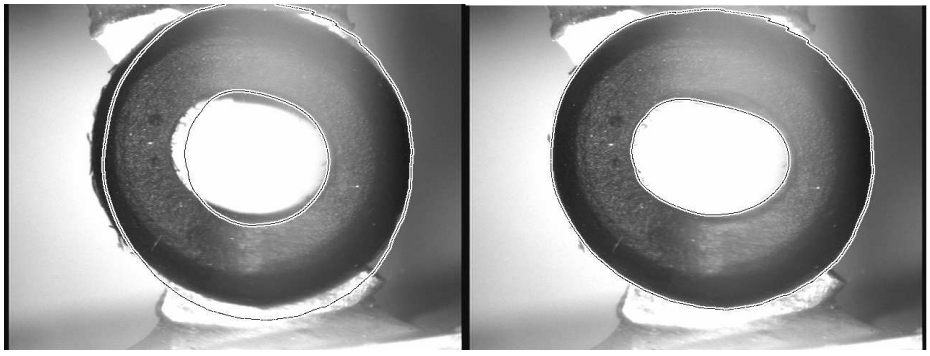


Figure 8.7: Neural network based deformable tracking algorithm applied the rubber torus object. The undeformed template is on the left and template fitted to deformed torus is on the right.

model the elastic behavior of this object accurately in real-time. A neural network was trained directly from 39 images of the torus under loads varying from 0.00 and 0.84 N. There are 70 hidden nodes in the neural network model. Once the training process was completed, the neural network elastic model was used to predict the force applied to the object visually as shown in Figure 8.7. This figure shows the torus being tracked with the deformable template based on the neural network model.

This trained model was tested on 39 images that were not used in the training process. Figure 8.8 shows a plot of vision-based force measurement versus applied load. The average error in the vision-based force measurement is 10.7 mN.

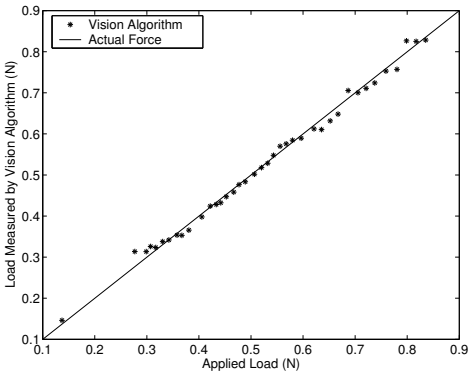


Figure 8.8: Neural network elastic model force measurement results for the rubber torus object.

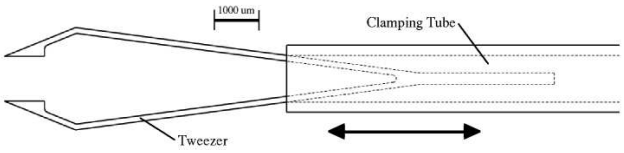


Figure 8.9: Microgripper.

8.6 Micro-gripper Application

The neural network modeling method was also applied to the microgripper shown in Figure 8.9. When this microgripper is used for a microassembly task it is important to measure the gripping force that the microgripper applies to the object being held. In Chapter 7, VBFM was applied to this gripper by modeling each jaw of the microgripper as a cantilever beam. This model based approach required precise knowledge of the material properties and geometric parameters of the microgripper. It also required a precise calibration of the camera system so that deflections in pixel space can be converted into world space deflections. The experimental setup shown in Figure 8.10 makes use of a piezoresistive force sensor to measure the clamping force for the microgripper. Figure 8.11 shows a plot of force versus clamping tube position for both the cantilever beam model based VBFM method and the piezoresistive force sensor. The average error with this model based approach to VBFM was 6.0 mN.

The neural network modeling approach can be applied to this microgripper avoiding the need for precise knowledge of the parameters that define the microgripper or precise

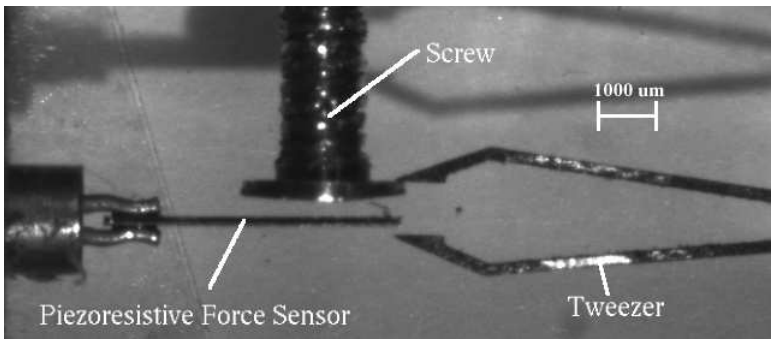


Figure 8.10: Experimental setup for measuring microgripper gripping force.

knowledge of the camera calibration parameters. Figure 8.12 shows a plot of force versus tube position for both the neural network based force measurement and the piezoresistive force sensor. The average error is 3.4 mN. Not only does the neural network modeling approach avoid having to model the microgripper, it also provides a more accurate result. The neural network approach is more accurate in this case because it is not necessary to assume values for the material properties and geometry parameters of the microgripper.

8.7 Conclusions

A method to model the deformation of elastic objects through the use of artificial neural networks has been presented. The neural network model can be incorporated into a deformable template matching algorithm to perform vision-based force measurement. This technique is useful for objects that have complex material models or objects that can not be accurately modeled with existing modeling techniques. It was also shown that this method can be useful even when there is an available model for the object because the neural network based method does not require knowledge of material properties or geometry. A precisely calibrated camera system is also not needed with the neural network approach.

This learning by seeing method is particularly useful for the manipulation of biological tissues or cells because it is difficult to accurately model such objects. These objects also tend to be easily damaged if excessive load is applied, making force feedback essential.

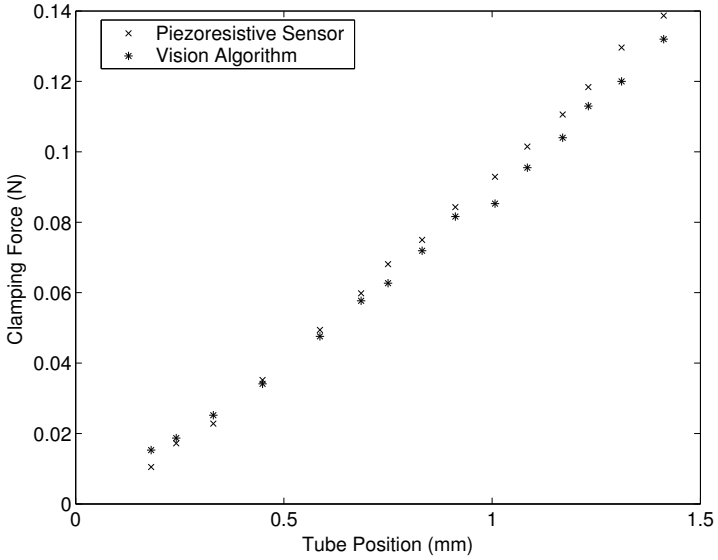


Figure 8.11: Cantilever beam model based VBFM results for the microgripper. The average error is 6.0 mN.

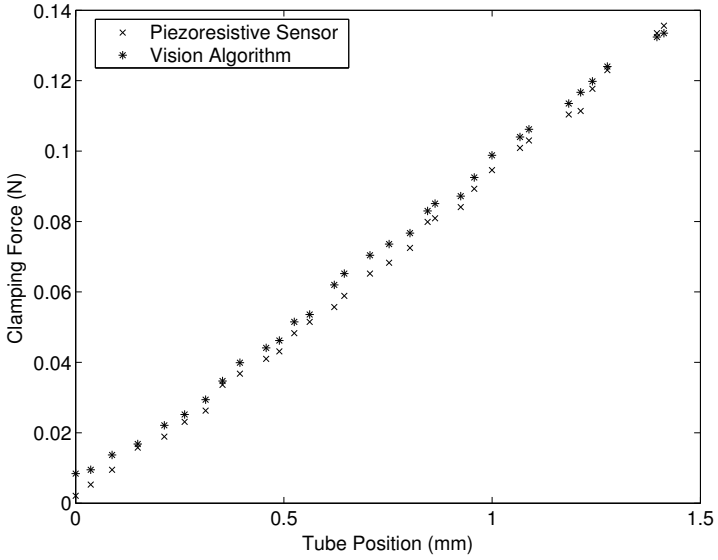


Figure 8.12: Neural network model based VBFM results for the microgripper. The average error is 3.4 mN.

Chapter 9

Deformable Object Tracking for Microrobotics: A MEMS 4 DOF Thermally Actuated Microgripper[1]

9.1 Introduction

Microrobotic systems are currently used for many applications in microassembly [9][53] and in biomanipulation [7][46]. Microrobotic systems hold much promise. However, the capabilities of these systems are limited when compared to their macroscale counterparts. For example, microgrippers typically are limited to a single degree of freedom for both actuation and feedback. Greater end effector dexterity and sensory feedback is required to produce more advanced microrobotic systems. This chapter presents technology that will help to increase capabilities of microrobotic systems by presenting a microgripper that both provides increased dexterity and the potential for increased sensory feedback when compared to previous designs.

In this chapter, a four degree of freedom bulk micromachined MEMS microgripper is presented. Each jaw of the microgripper is a compliant mechanism with both an x and a y degree of freedom. A bi-directional bending thermal actuator is used to actuate each degree of freedom. The thermal actuator is able to bend in both directions because the electric circuit is completed through the kinematic chain of the compliant mechanism. Therefore,

[1] ©2005 IEEE. Reprinted, with permission, from "A Four Degree of Freedom MEMS Microgripper with Novel Bi-Directional Thermal Actuators," Greminger, M.A., Sezen, A.S., Nelson, B.J., IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.

the location of heating within the actuator can be controlled remotely by applying electrical potential at the appropriate contacts.

Force feedback can also be provided for the microgripper design presented in this chapter. Since compliant mechanisms are used to provide the motion of the gripper jaws, the deformation of these compliant mechanisms can be tracked visually to provide force feedback through the use of vision-based force measurement. Force feedback can be provided for each degree of freedom of the gripper. It was shown in Chapter 7 that vision-based force measurement provides a robust and non-contact approach for providing force feedback in microsystems.

This chapter is organized as follows. First the design of the microgripper is discussed with the design of both the compliant mechanism and the design of the thermal actuators are addressed. Next, the fabrication and instrumentation of the microgripper is presented. The characteristics and performance of the devices and their actuators are presented next. Finally, a deformable tracking algorithm based on a frame FEM model is presented.

9.2 Design

The design goal for this microgripper is for each gripper jaw to have both x and y degrees of freedom. Typically, MEMS microgrippers only possess a single degree of freedom which opens and closes the gripper. The addition of these three extra degrees of freedom gives the gripper added dexterity.

A compliant mechanism is used for each jaw of the microgripper to give it the necessary degrees of freedom. The compliant mechanism design is based off of a five-bar rigid link mechanism. The design of the thermal actuators used to actuate the device is also discussed in this section.

9.2.1 Mechanism Design

The design of the compliant mechanism for each jaw of the gripper is based off of a five-bar mechanism design. A five-bar mechanism has two degrees of freedom. The input to the mechanism is the rotation angle of each of the links attached to ground as shown in Figure 9.1. The position of the entire mechanism is completely determined by the position of these two input links.

The design objectives for the compliant mechanism are to decouple each of the degrees

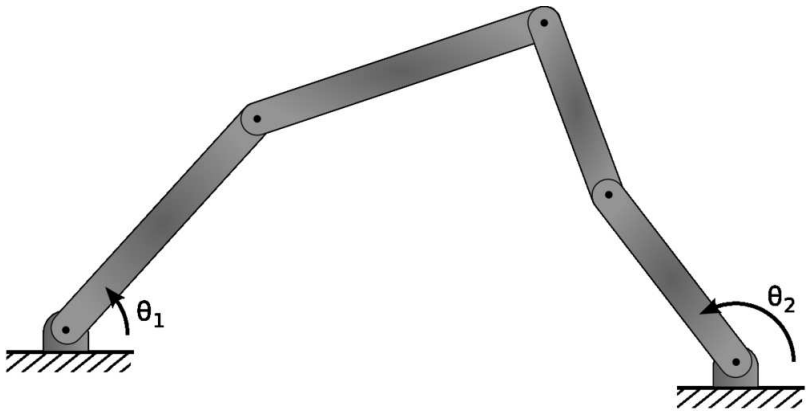


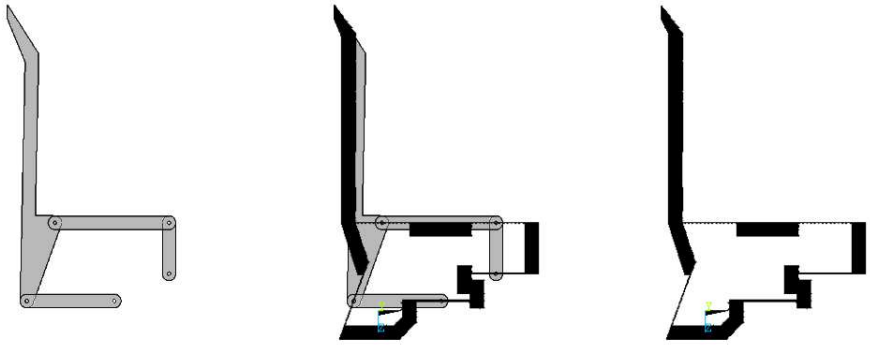
Figure 9.1: Five-bar rigid link mechanism.

of freedom and to amplify the motion of the actuator inputs. The mechanism design that is used for each jaw of the gripper is shown in Figure 9.2(a). The paths followed by the a gripper jaw for each degree of freedom are shown in Figures 9.3(a) and (b). It can be seen that the degrees of freedom are nearly completely decoupled.

A compliant mechanism is created from the rigid mechanism by using a compliant member to approximate each pivot joint in the rigid link design. Each compliant member is a flexible beam with the center point of the beam lying at the location of the pivot it replaces. The length of each of the compliant members determines how closely the motion of the compliant mechanism approximates that of the rigid link mechanism. The shorter the compliant members, the closer the motion of the compliant mechanism matches that of the rigid link mechanism. The drawback of short compliant links is that they experience greater internal stresses for the same level of gripper motion when compared to longer compliant members. Finite element simulations of the compliant mechanism were used to evaluate the kinematic performance of the compliant mechanism as well as to insure that the compliant members will not fail under normal operating conditions. Simulation results for each degree of freedom are shown in Figures 9.3(c) and (d).

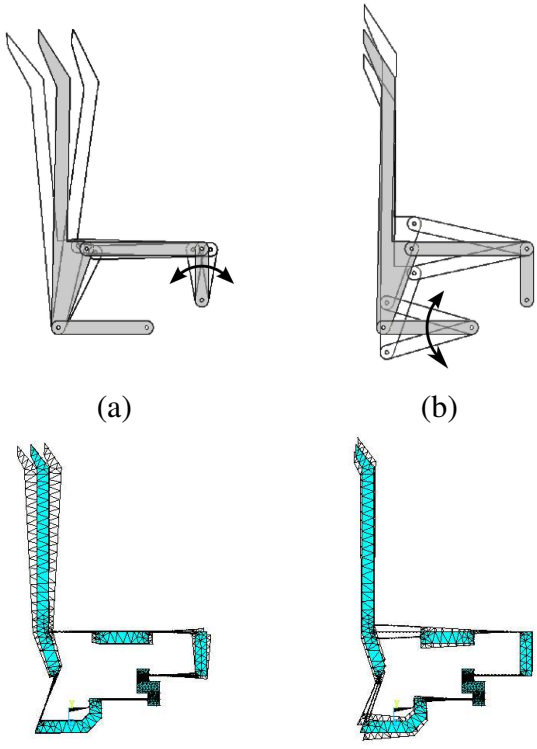
9.2.2 Actuator Design

Both electrostatic actuators and thermal actuators were considered as the source of actuation for each degree of freedom of the microgripper. Thermal actuation was chosen for two reasons. The first reason is that electrostatic actuators can not provide the forces necessary



(a) (b) (c)

Figure 9.2: A compliant mechanism (c) based on a rigid link mechanism (a).



(a) (b)

(c) (d)

Figure 9.3: Degrees of freedom for the rigid link mechanism (a)(b) and the associated compliant mechanism (c)(d).

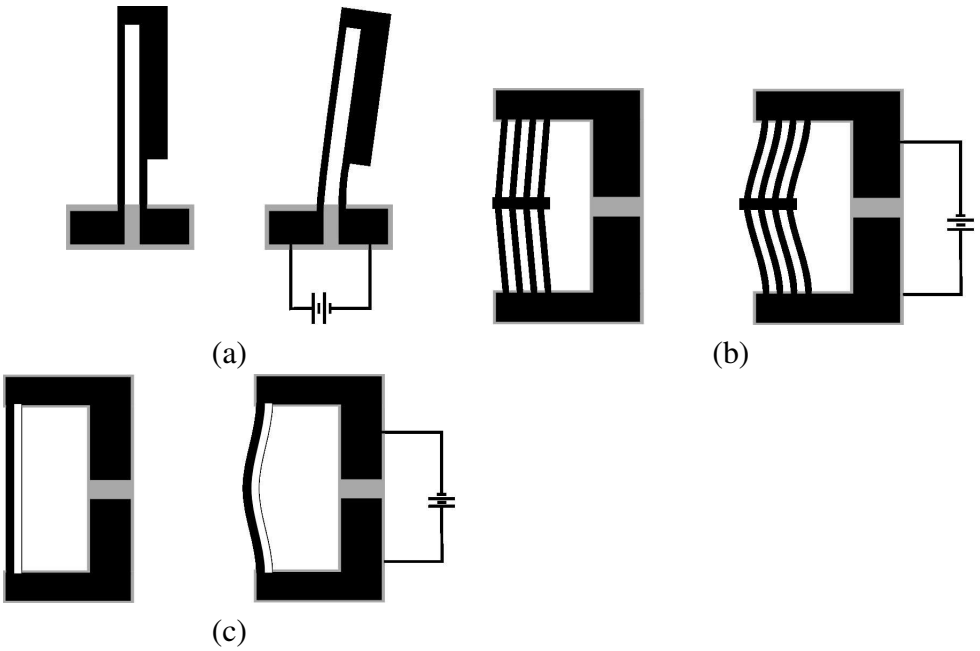


Figure 9.4: Common thermal actuator designs: (a) bending thermal actuator, (b) buckling beam thermal actuator, and (c) bimorph thermal actuator.

to actuate the compliant mechanism without requiring excessive device area or extremely high actuation voltages. Second, electrostatic actuators are most effective at providing linear actuation whereas the gripper's compliant mechanism requires a rotational input. Bending thermal actuators naturally provide a rotational motion.

All thermal actuators deflect as the result of an applied thermal stress where heating is generally provided by Joule heating. Thermal stresses arise in three situations: when there is a nonhomogeneous temperature distribution in a structure; when there is a nonhomogeneous coefficient of thermal expansion and the temperature changes; and when a structure is overly constrained and the temperature is changed thus causing buckling. Thermal actuators using each of these mechanisms are currently implemented in MEMS devices. The commonly used MEMS thermal actuators are shown in Figure 9.4. Figure 9.4(a) shows a thermal actuator that relies on nonuniform temperature distribution [17][19], Figure 9.4(b) shows a thermal actuator based on the buckling principle [38][42], and Figure 9.4(c) shows a thermal actuator that relies on a nonhomogeneous coefficient of thermal expansion [3][39]. The limitation of each of these designs is that they can only be actuated in a single direction.

The thermal actuator design that is proposed here is shown in Figure 9.5. As can be

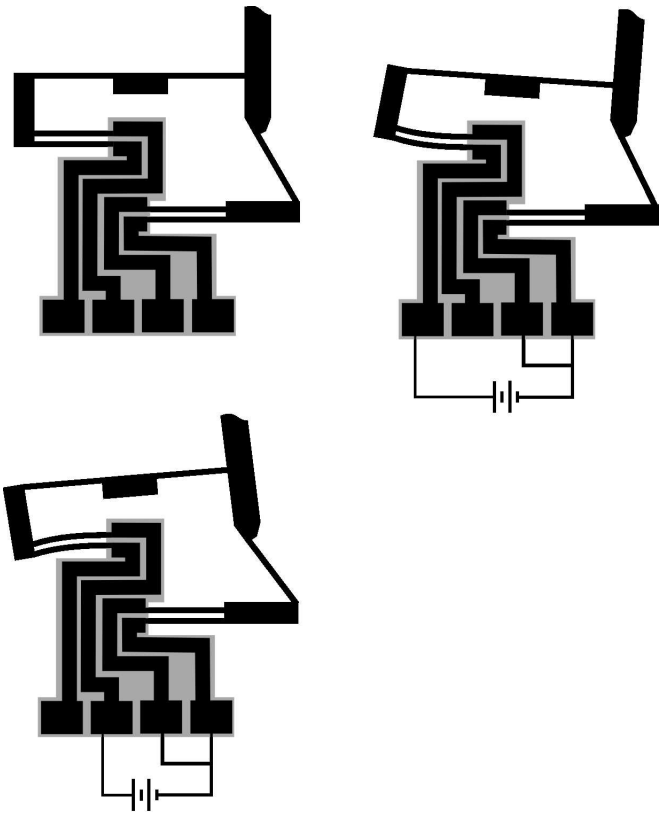


Figure 9.5: The thermal actuator design introduced in this chapter.

seen from the figure, the actuator can be actuated in either direction depending on which contacts electrical potential is applied. This is possible because the compliant mechanism itself is used to close the electrical path, thus providing a return path for the current used to heat the hot arm of the actuator. In order to control this type of an actuator with a computer, an interface which converts the signal level voltages from the computer to currents has been designed. This circuit is discussed in Section 9.3.2. The limitation of this type of actuator is that it can only be used in a situation where the actuator is connected to a closed kinematic chain. However, when a closed kinematic chain is being used, this design is superior to the previous actuator technologies because it can be actuated in two directions.

The displacement of this actuator design can be estimated by modeling the actuator as two parallel beams separated by an air gap. The beams are clamped on both ends as shown in Figure 9.6(a). The properties of each of the beams are identical with the only difference being the temperature. This temperature difference induces a stress which results in a deflection of the beam. The actuator is in a state of pure bending when the beams are at

different temperatures. The effective bending moment associated with this state of pure bending is [36]:

$$M = \frac{eEA\alpha}{2}(T_2 - T_1) \tag{9.1}$$

where A is the cross sectional area of each beam, T_1 and T_2 are the temperatures of beams 1 and 2 respectively, e is the distance between the centers of the beams, and α is the coefficient of thermal expansion for both of the beams. The moment M is the moment that would have to be applied to the end of the undeflected actuator to cause it to deflect the same magnitude as when the temperatures of beam 1 and 2 are T_1 and T_2 respectively. The radius of curvature ρ of the actuator is related to the bending moment by the following:

$$\frac{1}{\rho} = \frac{M}{EI} = \frac{eA\alpha}{2I}(T_2 - T_1) \tag{9.2}$$

where I is the moment of inertia of both beams together and is defined as:

$$I = 2\left(\frac{bh^3}{12} + \frac{Ae^2}{4}\right) \tag{9.3}$$

where h is the height of each beam and b is the depth of each beam. From the curvature of the beam, the total deflection of the actuator can be calculated using the following equation [55]:

$$\delta = \frac{l^2}{2\rho} = \frac{eA\alpha l^2}{4I}(T_2 - T_1) \tag{9.4}$$

where l is the length of the actuator. It can be seen from the above equation that the deflection of the actuator is proportional to the temperature difference between the two beams. The maximum actuation force can also be calculated in terms of the temperature difference between the beams. The maximum force that the actuator can supply is equivalent to the force F required to keep the end of the actuator from moving when a moment M of magnitude (9.1) is applied to the end of the beam (see Figure 9.6(c)). The equation for F is [55]:

$$F = \frac{3EI}{2l\rho} = \frac{3eEA\alpha}{4l}(T_2 - T_1) \tag{9.5}$$

This is the force that the actuator can generate at zero deflection which is referred to as the blocking force. The force that the actuator can generate decreases as the actuator travels through its range of motion.

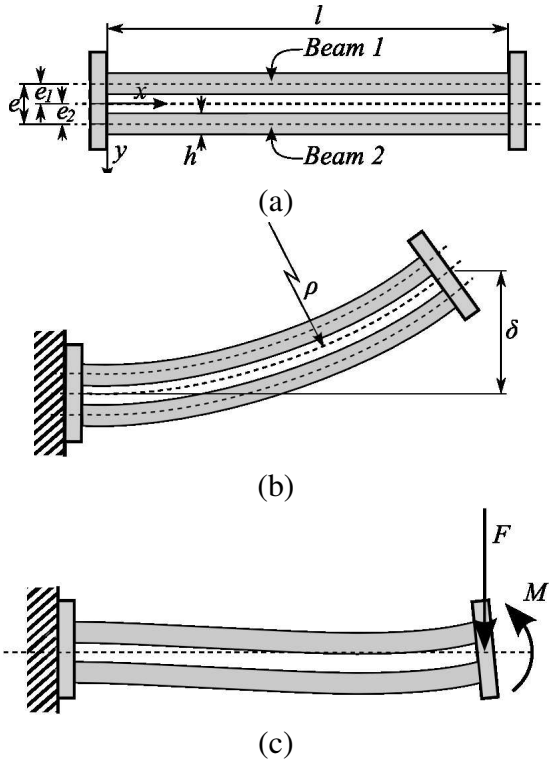


Figure 9.6: Model used to calculate the displacement of the actuator and the maximum force of the actuator.

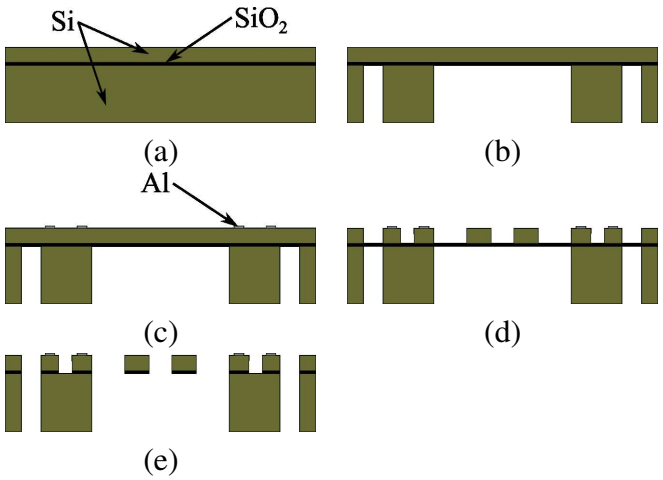


Figure 9.7: Fabrication sequence used to fabricate the microgripper.

9.3 Fabrication and Instrumentation

The microgripper is bulk micromachined from 100 μm thick <100> silicon with less than 0.01 Ohm-cm resistivity. This 100 μm silicon layer makes up the device layer of a silicon-on-insulator (SOI) wafer which has a 2 μm silicon-dioxide box layer and a 250 μm thick silicon handle layer. The oxide layer serves both as an etch stop layer during fabrication and as an electrical insulator for the final device. Deep reactive ion etching (DRIE) is used to create the microgripper's features.

After the microgripper is fabricated it needs to be interfaced with a circuit to provide current to actuate the thermal actuators. An interface circuit was designed so that voltages from a digital to analog converter can be used to control the displacement of the actuators.

9.3.1 Fabrication Process

The fabrication sequence is outlined in Figure 9.7. The SOI wafer that forms the starting point is shown in Figure 9.7(a). The first step is to etch the handle layer of the wafer using DRIE (see Figure 9.7(b)). This step forms the frame and the support structure for the final device. Figure 9.7(c) shows the next step which is to pattern the aluminum contacts that are used to electrically connect the device to its control circuit. The next step is to etch the device layer (see Figure 9.7(d)). The partions of the device layer that remains after this etching step forms the geometry of the gripper jaws, the compliant mechanisms, and the thermal actuators. Finally, the devices are released from the wafer by etching away the

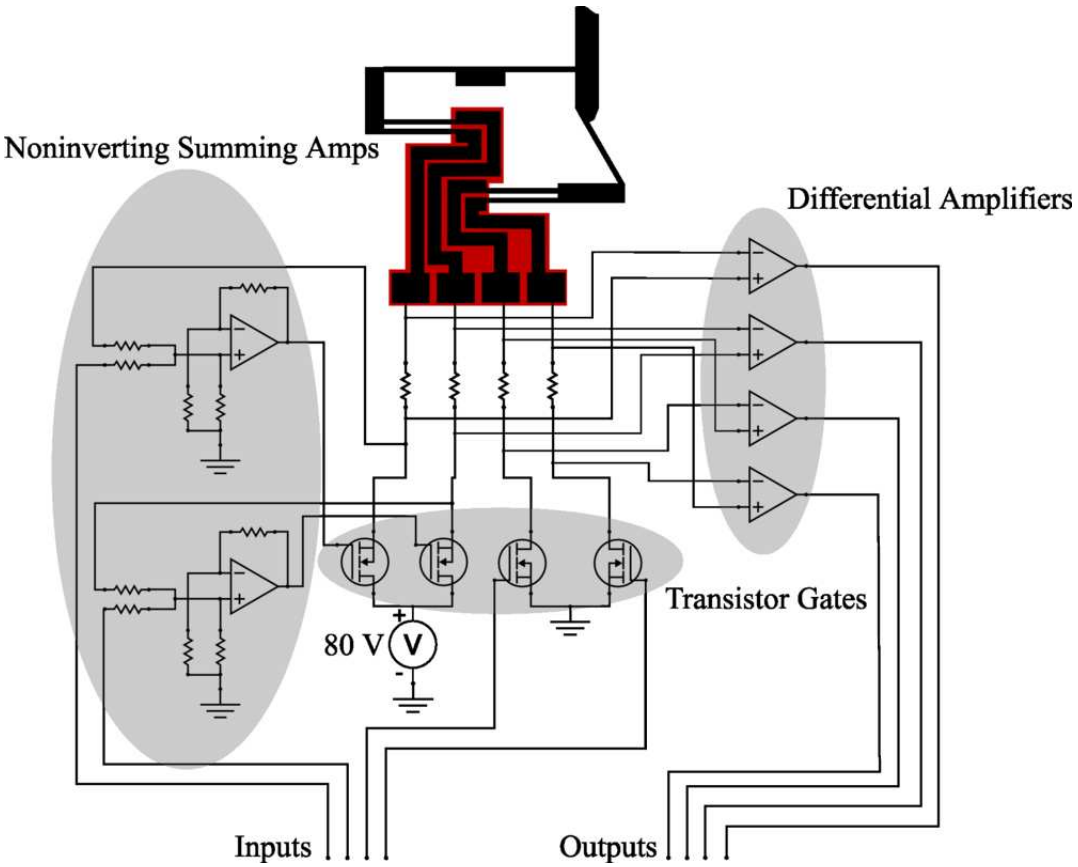


Figure 9.8: Interface circuit used to control the gripper with signal level voltage inputs.

exposed oxide (see Figure 9.7(e)). The oxide is etched using a plasma etcher.

9.3.2 Instrumentation

An interface circuit was implemented so that voltages from a digital to analog converter can be used to control the motion of the gripper. The direction of actuation, and which actuator is actuated depends on the path the electrical current takes through the device. Transistors are used as gates to control this path. The state of the transistors determines which of the two actuators is activated and the direction of actuation. A simplified version of the interface circuit is is shown in Figure 9.8 for one jaw of the microgripper. The input to the circuit is the gate voltage applied to each of the transistors. The input voltage is boosted using noninverting summing amplifiers for the transistors on the high voltage side of the circuit because the gate voltage required for these transistors is above signal levels.

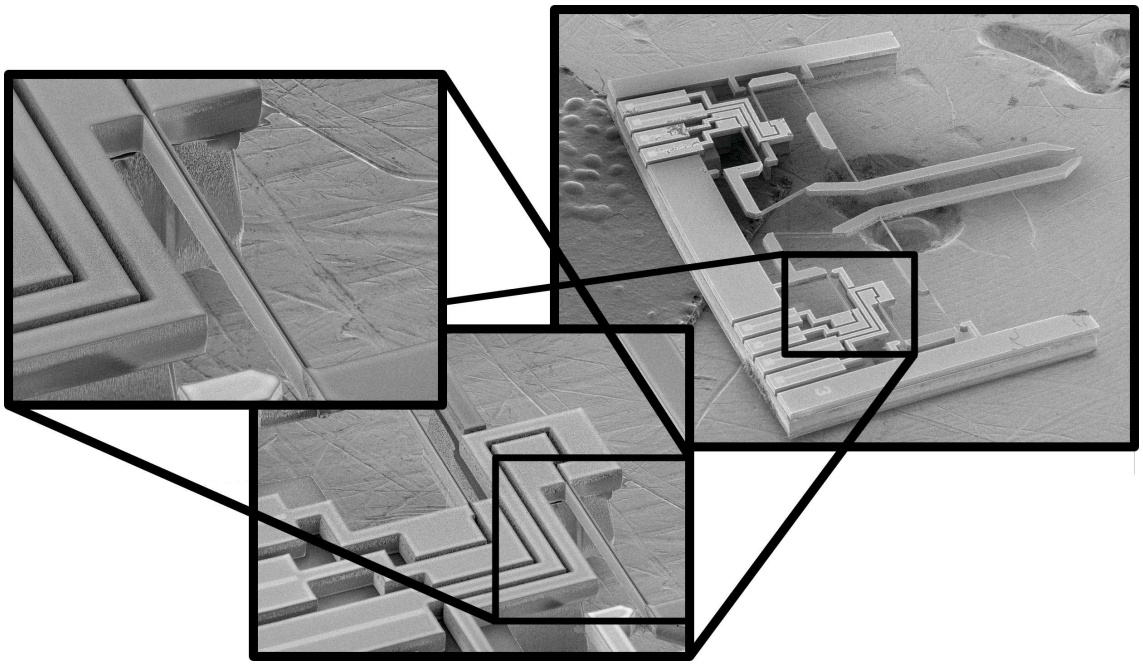


Figure 9.9: SEM image of the fabricated microgripper. This device is 6mm by 6mm in size. The insets show a closeup of one of the thermal actuators.

The supply voltage to the circuit is 80 volts. The voltage required to actuate the device to full scale deflection is 40 volts. The circuit runs at above 40 volts to account for voltage drops at the gate transistors.

Shunt resisters are placed in series with each electrical contact of the device. The voltage drop across each of these resisters is measured by a differential amplifier to provide a voltage proportional to the current entering the device. These current measurements are the output of the interface circuit and are used for two purposes. The first is to provide feedback for the user of the device. The second is to provide feedback for a current limiting circuit that is not shown. The current limiting circuit limits the maximum amount of current that can be run through the device. The current limiting circuit protects the device from being damaged by overheating. This limiting is important because in order to achieve the most performance out of the thermal actuators it is desirable to operate them as hot as possible without overheating them. Any control input can be sent to the device and the current limiting circuit insures that the device will not be overheated.

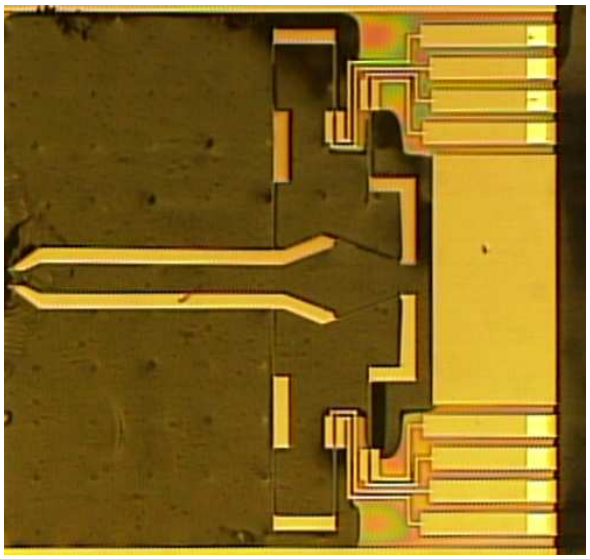


Figure 9.10: Image of fabricated microgripper. This device is 3 mm by 3 mm.

9.4 Actuation Results

Figure 9.9 shows an SEM image of a fabricated device. A microscope image of another device is shown in Figure 9.10. This device has an overall size of 3 mm by 3 mm and is used for the results presented in this chapter. Figure 9.11 shows the range of motion of the microgripper jaws for each of their degrees of freedom. The opening and closing range of motion for each gripper jaw is 38.4 μm, and the orthogonal range of motion for each gripper jaw is 11.6 μm.

The motion of a thermal actuator is shown in Figure 9.12. The total range of motion for this actuator is 12.7 μm. The actuator shown is 400 μm long, 100 μm deep, has beams that are 4 μm thick, and has a 10 μm gap between the beams. Using (9.4), the amplitude of deflection shown corresponds to a temperature difference of 578 °C between the hot and cold beams. The maximum force that this actuator can produce is 1.9 mN, which is calculated using (9.5).

For the results given here, the achievable actuator deflection was limited because the compliant members of the compliant mechanism overheated before the actuator could reach its maximum temperature. This is due to increased Joule heating of the compliant members as compared to the thermal actuator beams. This occurs even though the actuator beams and the compliant members were designed to have the same thickness. The compliant

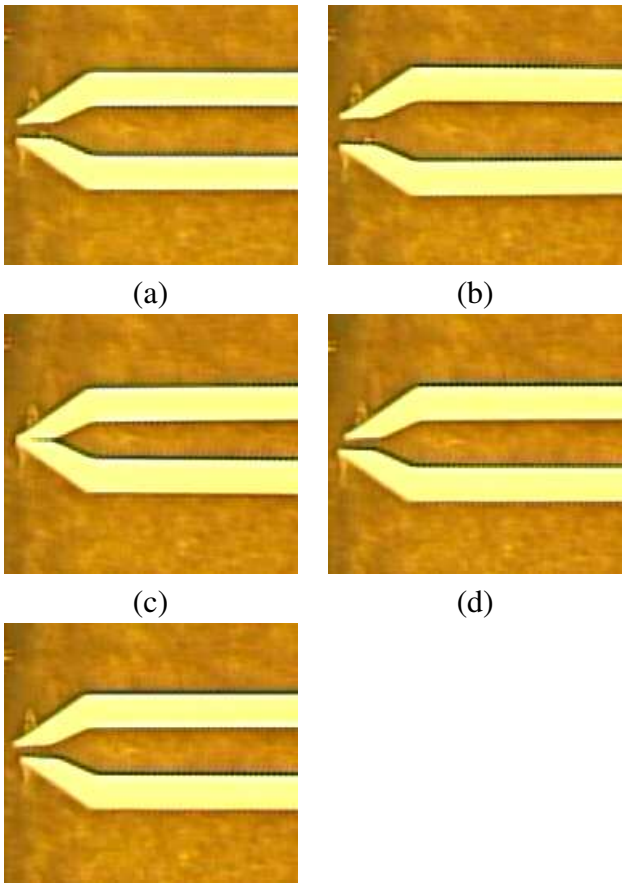
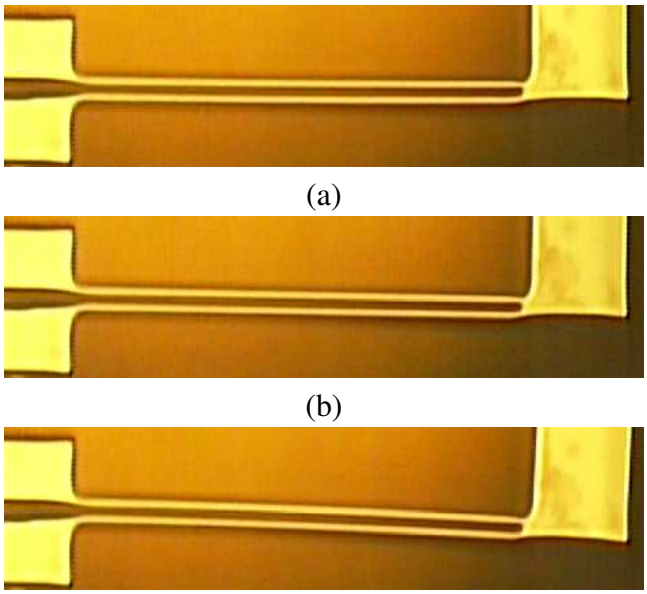


Figure 9.11: Range of motion for microgripper.

members are thinner in the final device because of the ununiform etching rate of DRIE. The problem of compliant member heating can be solved in future designs by patterning a metal conductor onto the compliant members. This fix would allow the actuators to reach their maximum temperature without the concern that the compliant members will overheat.

9.5 Deformable Object Tracking Applied to the Microgripper

Position and force feedback are needed for microgripper presented in this chapter to make it useful for micromanipulation applications. Since the motion of the microgripper is based on a compliant mechanism, the techniques that have been presented in this dissertation



(a)

(b)

(c)

Figure 9.12: Actuator motion.

for tracking deformable structures can be used to provide position and force feedback. A deformation model will be needed in order to perform the visual tracking necessary to provide this feedback. The boundary element method described in Chapter 5 could be used to model the microgripper but a model this general is not required for the microgripper. All of the deformation of the microgripper occurs in the springs. These springs can be very accurately modeled as beams. Therefore, a frame finite element model which models elastic objects as interconnected beam elements is a good choice for modeling the deformation of the microgripper. The use of a frame finite element model as opposed to a general boundary element model will greatly reduced the number of degrees of freedom that the model has. This decrease in the number of degrees of freedom will lead to a faster and more robust tracking algorithm.

9.5.1 The Frame Finite Element Model

The equations for general three dimensional linear elasticity are:

$$\sigma_{ij,j} + f_i = 0 \tag{9.6}$$

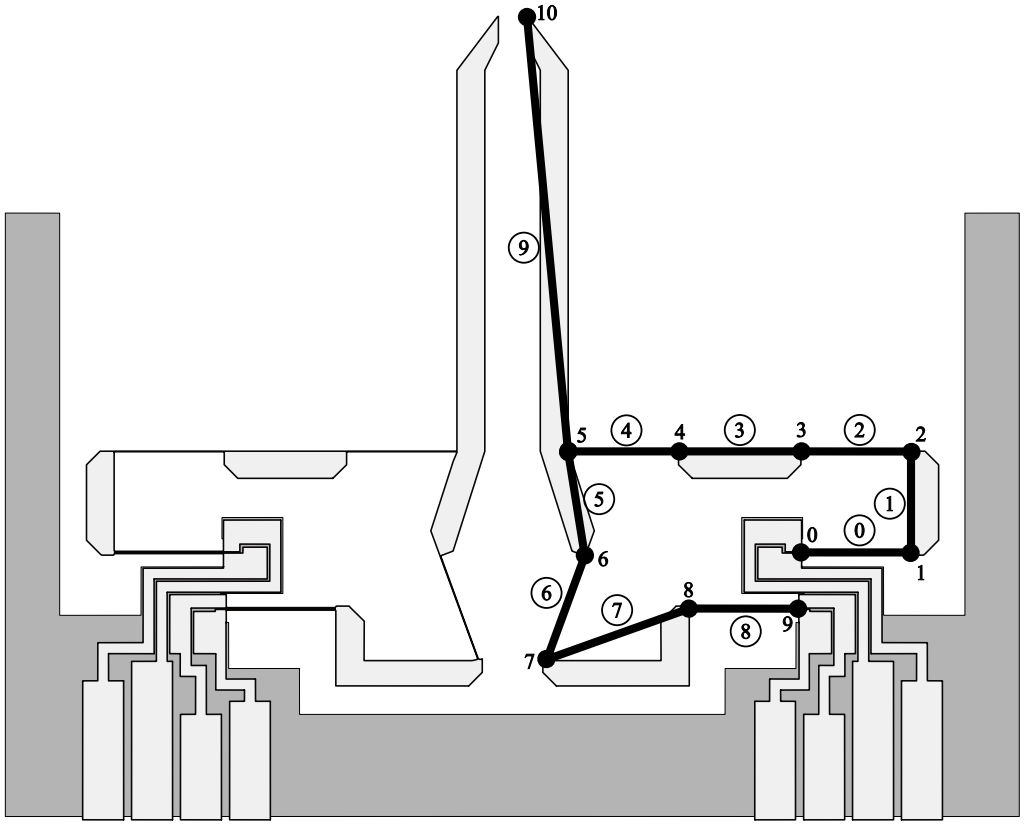


Figure 9.13: Frame mesh shown superimposed on the right jaw of the microgripper. Circled numbers label the elements and the other numbers label the nodes. Elements 1, 3, 5, 7, and 9 are assumed to be rigid.

$$\sigma_{ij} = c_{ijkl}\epsilon_{kl} \tag{9.7}$$

$$\epsilon_{ij} = u_{(i,j)} \tag{9.8}$$

where the indices take the values $1, 2, 3$. The finite element method for frames simplifies the above equations for general three dimensional linear elasticity. The domain of the frame finite element problem is composed of a finite number of beam segments which are connected at node points. Figure 9.13 shows the frame finite element mesh that is used to model each jaw of the microgripper. It is assumed that each of the beams is prismatic with a cross sectional area of A^e. A local coordinate system (x_1^e, x_2^e, x_3^e) is defined for each beam element where the x_3^e axis lies along the long axis of the beam segment (see Figure 9.14).

The stress components σ_{11}, σ_{12}, and σ_{22} in the beam segments are assumed to be zero. The displacement u_i of any point of a segment is defined by the following kinematic equa-

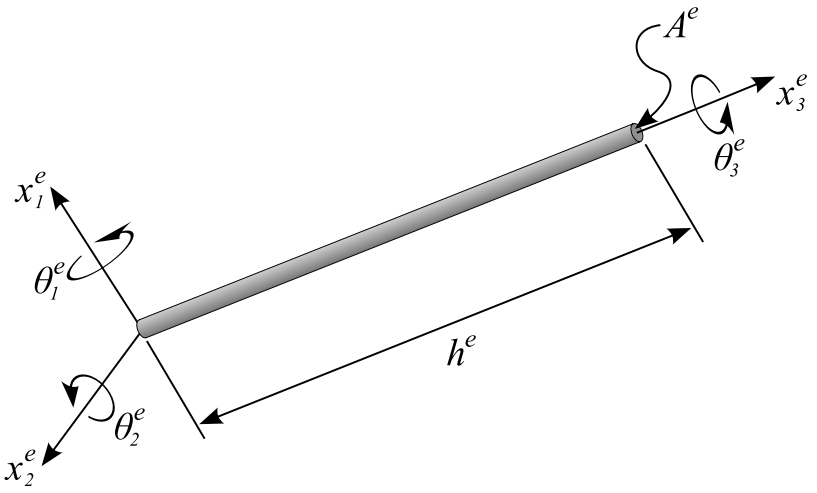


Figure 9.14: Frame beam element.

tions [21]:

$$u_1(x_1, x_2, x_3) = w_1(x_3) - x_2\theta_3(x_3) \tag{9.9}$$

$$u_2(x_1, x_2, x_3) = w_2(x_3) + x_1\theta_3(x_3) \tag{9.10}$$

$$u_3(x_1, x_2, x_3) = w_3(x_3) - x_1\theta_2(x_3) + x_2\theta_1(x_3) \tag{9.11}$$

where w_i and θ_i are the displacement and rotation degrees of freedom respectively for each frame node. These kinematic equations do not allow for warping of the plane sections of the segments.

Applying these assumptions, the constitutive equation (9.7) becomes

$$\sigma_{33} = E\epsilon_{33} \tag{9.12}$$

$$\sigma_{13} = 2\mu\epsilon_{13} \tag{9.13}$$

$$\sigma_{23} = 2\mu\epsilon_{23} \tag{9.14}$$

where E is the modulus of elasticity and μ is the shear modulus of the frame segment material. Applying the kinematic equations (9.9 - 9.11) the strain-displacement relationship (9.8) becomes:

$$\epsilon_{13} = \frac{\frac{\partial w_1}{x_3} - x_2 \frac{\partial \theta_3}{x_3} - \theta_2}{2} \tag{9.15}$$

$$\epsilon_{23} = \frac{\frac{\partial w_2}{x_3} + x_1 \frac{\partial \theta_3}{x_3} + \theta_1}{2} \tag{9.16}$$

$$\epsilon_{33} = \frac{\partial w_3}{x_3} - x_1 \frac{\partial \theta_2}{x_3} + x_2 \frac{\partial \theta_1}{x_3} \tag{9.17}$$

The finite element method is used to solve the elasticity equations for the frame FEM model (equations (9.6), (9.12) - (9.17)). The finite element method casts the equations into a variational form which can be solved by a matrix equation (see [21]). Linear shape functions are used to interpolate the displacement degrees of freedom and Gaussian integration is used to integrate each of the element equations. The matrix form of the frame finite element model is

$$[K]\{u\} = \{t\} \tag{9.18}$$

where $[K]$ is the stiffness matrix, $\{u\}$ is the nodal displacement and rotation vector, and $\{t\}$ is the nodal force and moment vector. Given a traction distribution $\{t\}$ applied to the frame finite element model, the nodal displacements can be found by solving the above matrix equation.

9.5.2 Tracking the MEMS Microgripper using the Frame FEM Deformation Model

The frame FEM deformation model described above is used to deform the template that is used to track the MEMS microgripper. The error function (2.7) becomes

$$E(\theta, X, \{t\}) = \sum_{i=1}^{M} \|r_i' - w_i\|^2 \tag{9.19}$$

where $\{t\}$ is the vector of forces and moments applied to the frame FEM mesh of the microgripper.

Figure 9.15 shows a tracking result using the frame FEM template described in this Chapter. By deforming the template with a less general model than the BEM model, the tracking is performed more quickly and more robustly than would otherwise be possible.

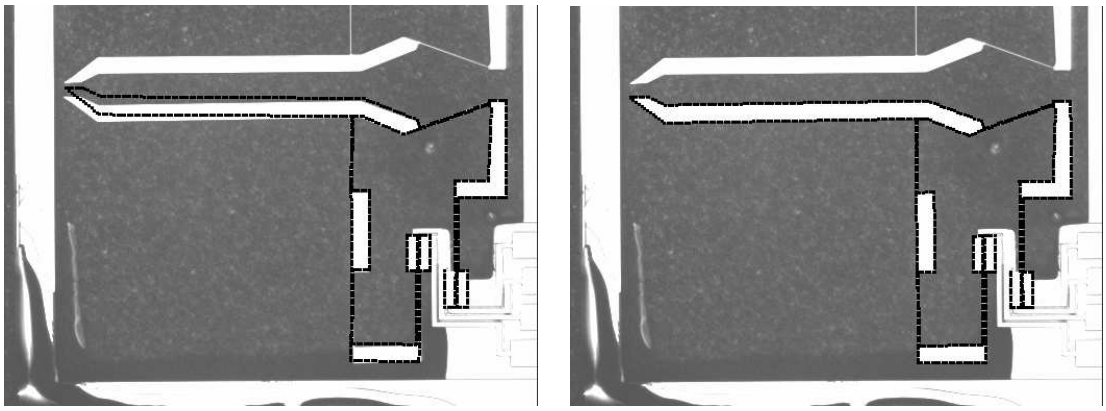


Figure 9.15: Results tracking the MEMS microgripper using the frame FEM deformation model to deform the template.

9.6 Conclusions and Discussion

A new gripper design has been presented that offers greater manipulation dexterity than is currently available. It also introduces a novel thermal actuator design that takes advantage of the closed kinematic chain design of the compliant mechanism to allow actuation in two directions. A deformable object tracking algorithm was also introduced for tracking the deformation of this microgripper in order to provide information for position and force feedback. A frame FEM model was used to deform the template since the springs of the microgripper are readily modeled using beam elements.

Improving the dexterity and the sensing capabilities of micromanipulators will greatly enhance the cababilities of micromanipulation and biomanipulation systems. These capabilities will allow the algorithms and techniques used in macromanipulation to be applied to micromanipulation.

Chapter 10

Conclusions and Discussion

10.1 Overview

This dissertation has presented a general method to track deformable objects in images. It has been shown that tracking deformable objects can be used to provide position and force feedback for microrobotic applications. These measurements can be made in situations where traditional displacement sensing techniques cannot be used. These situations include biomanipulation applications and non-invasive medical imaging applications. The use of various material models has been demontstrated including the use of an artificial neural network to learn the deformation models of objects with non-linear properties.

The robotic manipulation of deformable objects is becoming more important with the emergence of robotic surgery. Also, the limitations of robotics in certain domains including space robotics and microrobotics requires robotic manipulators that are flexible. Both of these instances require feedback of the current state of the deformable object or of the deformable manipulator. The high dimensional information that is available in images is ideal for obtaining the shape of deformable structures which inherently have many degrees of freedom. Computer vision will form a central role in the development of robots that interact with deformable objects and the in the control of robots that are themselves deformable.

10.2 Contributions to the State of the Art

Vision-Based Force Measurement: The primary goal of this work is to demonstrate the feasibility of measuring forces from images and to implement an algorithm to do this measurement accurately and robustly. Vision-based force measurement has been demonstrated for a linearly elastic cantilever beam and a one degree of freedom microgripper where forces were measured to a accuracy of +/- 2.8 nN and +/- 3.1mN respectively. Vision-based force measurement was also demostrated for an object with nonlinear material properties by using a neural network model to learn its material properties. For each of these test cases, the results were compared to a traditional sensor in order to accurately assess the performance of the vision-based approach.

Deformable Object Tracking Algorithm Robust to Occlusion and Spurious Edges: It was demonstrated that the deformable object tracking algorithm presented in this dissertation can be modified so that it is robust to two common sources of errors in image tracking: occlusions and spurious edges. It was shown that deformable objects can be tracked even in the presence of significant occlusion by using robust error measures. The problem of spurious edges was handled by introducing a new modification to the Canny edge operator that makes use of object intensity to eliminate spurious edges. These enhancements are critical for robotics applications where occlusions or spurious edges can arise without warning.

Introduction of a Modular Deformable Object Tracking Algorithm: The deformable object tracking algorithm presented in this dissertation is the first to completely separate the low-level vision routines, the deformation model, and the error minimization routine in a modular way. This modularity will allow future researchers to easily add new deformation models, make use of more robust edge detection algorithms, and make use of more efficient numerical minimization algorithms. Previous deformable object tracking algorithms have not been modular making them difficult to modify to solve new problems.

Four Degree of Freedom Microgripper: A new microgripper design was introduced which has four degrees of freedom total where typical designs have a single degree of freedom for opening and closing. These additional degrees of freedom provide added dexterity and allow for fine positioning once an object is grasped. The degrees of freedom are provided by a compliant mechanism based off of the design of a rigid link five-bar mecha-

nism. The compliant nature of this device also provides the opportunity to use deformable object tracking to provide position and force feedback.

New Thermal Actuator Design Capable of Bi-Directional Actuation: Finally, a new MEMS thermal actuator design was introduced that can be actuated in two directions. This actuator is able to actuate in two directions by taking advantage of the closed kinematic chain of the microgripper to complete the current path. Because of this characteristic, the thermal actuator design introduced has double the range of motion of previous MEMS thermal actuator designs.

Bibliography

[1] Acker, J., Henrich, D., "Manipulation of Deformable Linear Objects: From Geometric Model Towards Program Generation," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA2005),*Barcelona, Spain, 2005.

[2] Beer, G., *Programming the Boundary Element Method,* Wiley, New York, 2001.

[3] Benecke, W., Riethmuller, W., "Applications of silicon-microactuators based on bimorph structure," *Micro Electro Mechanical Systems, Proceedings, 'An Investigation of Micro Structures, Sensors, Actuators, Machines and Robots'*, pp. 116 - 120, Feb. 20-22, 1989.

[4] Bogen, D.K., Rahdert, D.A., "A strain energy approach to regularization in displacement field fits of elastically deforming bodies," *IEEE Transactions on Pattern Analysis and Machine Intelligence,*Vol. 18, No. 6, pp. 629-35, June 1996.

[5] Canny, J., "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,*Vol. PAMI-8, No. 6, November 1986.

[6] Carpenter, B., Goldstein, D., "Next Generation Solar Array Technologies for Small Satellites," *16th Annual AIAA/USU Conference on Small Satellites,* Logan, Utah, August 12-15, 2002.

[7] Chronis, N., Lee, L.P., "Polymer MEMS-based microgripper for single cell manipulation, " *17th IEEE International Conference on Micro Electro Mechanical Systems (MEMS2004),* pp:17 - 20, 2004.

[8] Danuser, G., Mazza, E., "Observing Deformations of 20 Nanometer With a Low Numerical Aperture Light Microscope," *Optical Inspection and Micromeasurements,* Vol. 2782, pp. 180-191 SPIE, 1996.

[9] Dechev, N., Cleghorn, W.L., Mills, J.K., "Microassembly of 3-D microstructures using a compliant, passive microgripper," *Journal of Microelectromechanical Systems,* vol. 13, no. 2, pp. 176 - 189, April 2004.

[10] Dennis, J.E., Schnabel, R.B., *Numerical Methods for Unconstrained Optimization of Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[11] Dong, L., Arai, F., Fukuda, T., "3D Nanorobotic Manipulations of Multi-Walled Carbon Nanotubes," *Proc. 2001 IEEE Int. Conf. of Robotics and Automation (ICRA2001)*, Seoul,Korea, May 2001.

[12] Draper, N.R., *Applied Regression Analysis,*3rd Edition, John Wiley and Sons, Inc., New York, 1998.

[13] Fletcher, R., *Practical Methods of Optimization,* John Wiley and Sons, New York, 1987.

[14] Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Booth, M., Rossi, F., *GNU Scientific Library Reference Manual,* Network Theory Limited, Bristol, UK, 2001.

[15] Gao, X.W., Davies, T.G., *Boundary Element Programming in Mechanics,* Cambridge University Press, Cambridge, UK, 2002.

[16] Greminger, M.A., Nelson, B.J., "Vision-Based Force Measurement," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 26, No. 3, pp. 290-298, March 2004.

[17] Guckel, H., Klein, J., Christenson, T., Skrobis, K., Laudon, M., Lovell, E.G., "Thermo-Magnetic Metal Flecture Actuators," *Proceedings of the 5th IEEE Solid-State Sensor and Actuator Workshop,*, Hilton Head Island, SC, USA, June 22-25, 1992.

[18] Guenther, R.B., Lee, J.W., *Partial Differential Equations of Mathematical Physics and Integral Equations,* Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[19] Huang, Q.A., Lee, N.K.S., "Analysis and design of polysilicon thermal flexure actuator," *Journal of Micromechanics and Microengineering,* vol. 9, no. 1, pp. 64-70, March 1999.

[20] Howe, R.D., Matsuoka, Y., "Robotics for Surgery," *Annual Review of Biomedical Engineering,*no. 01, 1999.

[21] Hughes, T. J.R., *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis,*Dover Publications, Mineola, New York, 2000.

[22] Jaswon, M. A., Symm, G. T., *Integral Equation Methods in Potential Theory and Elastostatics,* Academic Press, New York, 1977.

[23] Jain, R., Rangachar, K., Schunck, B.G., *Machine Vision* , McGraw-Hill, Inc., 1995.

[24] Kaneko, M., Nanayama, N., Tsuji, T., "Vision-based active sensor using a flexible beam," *IEEEASME Transactions on Mechatronics,* Vol. 6, Issue 1, pp. 7-16, March 2001.

[25] Kass, M., Witkin, A., Terzopoulos, D., "Snakes: Active Contour Models," *International Journal of Computer Vision,*pp. 321-331, 1988.

[26] Kecman, V., *Learning and Soft Computing,* MIT Press, Cambridge, Mass, 2001.

[27] Knops, R.J., Payne, L.E., *Uniqueness Theorems in Linear Elasticity,* Springer-Verlag, Berlin Heidelberg, 1971.

[28] Kypson, A.P., Randolph Chitwood, W., "Robotic Applications in Cardiac Surgery," *International Journal of Advanced Robotic Systems,* vol. 1, no. 2, 2004.

[29] Lichter, M., Dubowsky, S., "Shape, Motion, and Parameter Estimation of Large Flexible Space Structures using Range Images," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA2005),*Barcelona, Spain, 2005.

[30] Low, S.C., Phee, L., "A Review of Master-Slave Robotic Systems for Surgery," *Proceedings of teh 2004 IEEE Conference on Robotics, Automation and Mechatronics,*Sigapore, Dec. 1-3, 2004.

[31] Luo, Q., Xiao, J., "Haptic Rendering Involving an Elastic Tube for Assembly Simulations," *Proceedings of the 6th IEEE International Symposium on Assembly and Task Planning (ISATP2005),* Motreal, Canada, 2005.

[32] McInerney, T., Terzopoulos, D., "A finite element model for 3D shape reconstruction and nonrigid motion tracking," *Proceedings Fourth IEEE International Conference on Computer Vision,* pp. 518-23, April 1993.

[33] Metaxas, D., *Physics-Based Deformable Models,*Kluwer Academic Publishers, Boston, Mass, 1997.

[34] Nakamura, Y., Kishi, K., Kawakami, H., "Heartbeat synchronization for robotic cardiac surgery," *IEEE International Conference on Robotics and Automation (ICRA2001),*Vol. 2, pp. 2014 -2019, Soul, Korea, May 2001.

[35] Nelson, B., Zhou, Y., Vikramaditya, B., "Sensor-Based Microassembly of Hybrid MEMS Devices," *IEEE Control Systems,* December 1998.

[36] Noda, N., Hetnarski, R.B., Tanigawa, Y., *Thermal Stress, 2nd ed.,*Taylor and Francis, London, 2003.

[37] Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T., *Numerical Recipes in C : The Art of Scientific Computing,* Cambridge University Press, Cambridge, England, 1993.

[38] Que, L., Park, J.S., Gianchandani, Y.B., "Bent-beam electro-thermal actuators for high force applications," *Twelfth IEEE International Conference on Micro Electro Mechanical Systems (MEMS99),*, pp. 31 - 36, Jan. 17-21, 1999.

[39] Riethmüller, W., Benecke, W., "Thermally Excited Silicon Microactuators," *IEEE Transactions on Electron Devices,*vol. 35, no. 6, pp. 758 - 763, June 1988.

[40] Rizzo, F. J., "An integral equation approach to boundary value problems of classical elastostatics," *Quarterly Applied Mathematics,* 25:83-95, 1967.

[41] Samet, H., *The Design and Analysis of Spatial Data Structures,* Addison-Wesley, Reading, Mass., 1990.

[42] Sinclair, M.J., "A high force low area mems thermal actuator," *The Seventh Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems,*vol. 1 ,pp. 127-132, May 23-26, 2000.

[43] Sokolnikoff, I., *Mathematical Theory of Elasticity,* Krieger Publishing Company, Malabar, Florida, 1983.

[44] Stewart, C.V., "Robust Parameter Estimation in Computer Vision," *SIAM Review,*Vol. 41, No. 3, pp. 513-537, 1999.

[45] Sun, Y., Nelson, B.J., "Biological cell injection using an autonomous microrobotic system," *The International Journal of Robotics Research (IJRR),* Vol. 21, No. 10-11, pp. 861-868, Oct.-Nov., 2002.

[46] Sun, Y., Greminger, M.A., Nelson, B.J., "Investigating Protein Structure Change in the Zona Pellucida with a Microrobotic System," *International Journal of Robotics Research,* 2004.

[47] Sylvester, J., Uhlmann, G., "The Dirichlet to Neumann map and applications," *Inverse problems in partial differential equations,* editors: Colton, D., Ewing, R., Rundell, W., SIAM, 1990, pp. 101-139.

[48] Taylor, R., H.A. Paul, B. Mittelstadt, and e. al., "Robotic Hip Replacement Surgery in Dogs," *Eleventh Annual Conference on Engineering in Medicine and Biology, IEEE,*Seattle, 1989.

[49] Tortonese, M., Yamada, H., Barrett, R C., Quate, C F., "Atomic force microscopy using a piezoresistive cantilever," *Transducers 91 International Conference on Solid State Sensors and Actuators,* pp. 448-451, 1991.

[50] Tsap, L., Goldgof, D., Sarkar, S., "Efficient Nonlinear Finite Element Modeling of Nonrigid Objects via Optimization of Mesh Models," *Computer Vision and Image Understanding,* Vol. 69, pp. 330-350, March 1998.

[51] Wang, X., Ananthasuresh, G. K., Ostrowski, J. P., "Vision-based sensing of forces in elastic objects," *Sensors and Actuators A-Physical,* Vol. 94, No. 3, pp. 142-156, November 2001.

[52] Yang, G., Gaines, J. A., Nelson, B. J., "A Flexible Experimental Workcell for Efficient and Reliable Wafer-Level 3D Microassembly," *Proc. 2001 IEEE Int. Conf. of Robotics and Automation (ICRA2001),* Soul, Korea, May 2001.

[53] Yesin, K.B., Nelson, B.J., "A CAD-Model Based Tracking System for Visually Guided Microassembly," *Robotica,* 2004.

[54] Yeung, F., Levinson, S.F., Fu, D., Parker, K.J., "Feature-adaptive motion tracking of ultrasound image sequences using a deformable mesh," *IEEE Transactions on Medical Imaging,*Vol. 17, No. 6, pp. 945-956, Dec. 1998.

[55] Young, W.C., Budynas, R., *Roark's Formulas for Stress and Strain, 7th ed.,*McGraw-Hill, New York, 2001.

[56] Yuille, A., Cohen, D., Hallinan, W., "Feature extraction from faces using deformable templates," *International Journal of Computer Vision,* Vol. 8, No. 2, pp. 99-111, 1992.