Int J Comput Vis (2008) 78: 29–45
DOI 10.1007/s11263-007-0076-6

A Deformable Object Tracking Algorithm Based on the Boundary Element Method that is Robust to Occlusions and Spurious Edges

Michael A. Greminger · Bradley J. Nelson

Received: 29 January 2006 / Accepted: 17 July 2007 / Published online: 9 August 2007

Abstract The manipulation of deformable objects is an important problem in robotics and arises in many applications including biomanipulation, microassembly, and robotic surgery. For some applications, the robotic manipulator itself may be deformable. Vision-based deformable object tracking can provide feedback for these applications. Computer vision is a logical sensing choice for tracking deformable objects because the large amount of data that is collected by a vision system allows many points within the deformable object to be tracked simultaneously. This article introduces a template based deformable object tracking algorithm, based on the boundary element method, that is able to track a wide range of deformable objects. The robustness of this algorithm to occlusions and to spurious edges in the source image is also demonstrated. A robust error measure is used to handle the problem of occlusion and an improved edge detector based on the Canny edge operator is used to suppress spurious edges. This article concludes by quantifying the performance increase provided by the robust error measure and the robust edge detector. The performance of the algorithm is also demonstrated through the tracking of a sequence of cardiac MRI images.

Keywords Deformable object tracking · Boundary element method · Robust tracking · Edge detection · Robust statistics · Artificial neural networks

M.A. Greminger (✉)
Department of Mechanical Engineering, University of Minnesota, Minneapolis, MN 55455, USA
e-mail: grem@me.umn.edu

B.J. Nelson
Institute of Robotics and Intelligent Systems, Swiss Federal Institute of Technology (ETH), 8092 Zurich, Switzerland

1 Introduction

Deformable object tracking has many important applications. Application areas include medical imaging (Metaxas 1997; McInerney and Terzopoulos 1993), robotic manipulation of deformable objects (Nakamura et al. 2001; Sun and Nelson 2002), and vision-based force measurement (Greminger and Nelson 2004; Wang et al. 2001). The use of elastic models to track these deformable objects 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 article builds upon the methods in this second class.

1.1 Active Contour Model Based Methods

Kass et al. (1988) 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. There are many methods that build upon the active contour model framework. Yuille et al. (1992) 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.

1.2 Model-Based Methods

Metaxas (1997) introduced the use of 3D meshes with physics-based elastic properties to track both rigid and non-rigid objects. The primary application of this 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. Tsap et al. (1998) proposed a method to use nonlinear finite element modeling (FEM) to track non-rigid objects in order to detect the differences in elasticity between normal and abnormal skin. There has also been work in using FEM based deformable object tracking algorithms to measure forces from images. Wang et al. (2001) used Finite Element Modeling (FEM) techniques to derive the forces that were applied to deformable microparts. Their method is limited by the need to track each FEM mesh point in the image. The algorithm presented here builds upon these existing model-based deformable object tracking algorithms.

1.3 Deformable Object Tracking Using the Boundary Element Method

As mentioned above, existing model based tracking algorithms have used the finite element method (FEM) to model material deformations. The boundary element method (BEM), like FEM, is a method to model an elastic 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 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 flip normals if they experience a large deformation.

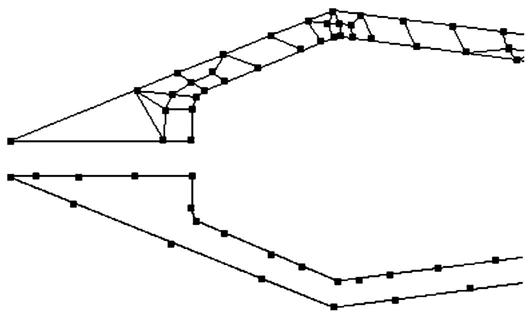
Fig. 1 Comparison between a 2D boundary element mesh (*bottom*) and a finite element mesh (*top*)

Enhancements are also presented that make the deformable tracking algorithm robust to occlusions in the image and to spurious edges in the image. A robust error measure, which replaces the least squares error measure, is used to handle the occlusion problem. To handle the spurious edge problem, a modified edge detection algorithm is presented that is able to suppress spurious edges. The modified edge detection algorithm is based off of the Canny edge detection algorithm and uses an artificial neural network to classify each edge as either a desired edge or a spurious edge.

This article is organized as follows: An overview of the deformable object tracking algorithm used is presented in Sect. 2. Section 3 presents the boundary element method and the method used to numerically solve the boundary element problem. The actual deformable template matching algorithm that makes use of the solution to the boundary element problem is presented in Sect. 4. Section 5 presents tracking results for the deformable object tracking algorithm. Modifications to make the tracking algorithm robust to occlusions and spurious edges are presented in Sect. 6. These modifications include the use of a robust error measure and the use of a new edge detection algorithm which suppresses spurious edges. Quantitative tracking results are presented in Sect. 7. The results of tracking a series of Cardiac MRI images are also presented in this section. Finally, a summary and concluding comments are presented in Sect. 8.

2 Overview of the Deformable Object Tracking Algorithm

A template based deformable object tracking algorithm is used in this article. The image is first converted to a binary edge image using the Canny edge operator (Canny 1986), and then the error is measured between the image and the template using a least squares error measure. The template has both rigid body and deformable degrees of freedom. The deformable degrees of freedom are provided by the boundary element method. The error between the template and the image is minimized using a gradient based numerical minimization routine. This section describes the error measure used, the template degrees of freedom, and the minimization of the error function.

2.1 The Least Squares 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 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 $\mathbf{x}$.

$$\mathbf{r}'_i = T(\mathbf{r}_i, \mathbf{x}). \tag{1}$$

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

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

where $\mathbf{r}'_i$ is the position vector of the ith edge pixel of the template transformed by (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 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

The degrees of freedom of the template are defined next. The degrees of freedom consist of a rigid body motion contribution and a deformable body contribution. The rigid body component is an affine transformation and the deformable body component is based on elasticity theory.

2.2.1 *The Rigid Body Motion Degrees of Freedom*

For the rigid body case the template affine transform is defined as

$$\begin{aligned}\mathbf{r}' &= T(\mathbf{r}, \theta, S, \mathbf{X}) = A(r) \\ &= \mathbf{X} + \begin{bmatrix} S\cos\theta & -S\sin\theta \\ S\sin\theta & S\cos\theta \end{bmatrix}\mathbf{r},\end{aligned} \tag{3}$$

where θ is the angle of rotation, S is the scale factor, and $\mathbf{X}$ is the translation vector. Figure 3 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

$$E(\theta, S, \mathbf{X}) = \sum_{i=1}^{M} \|\mathbf{r}'_i - \mathbf{w}_i\|^2. \tag{4}$$

2.2.2 *The Deformation Degrees of Freedom*

Since the goal of the method is to track 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 deformable object tracking algorithms can be used. The deformable object tracking algorithm presented in this article uses the boundary element method (BEM) to model deformations.

Deformations are expressed in general by the following equation

$$\mathbf{u} = D(\mathbf{r}, \{t\}), \tag{5}$$

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

$$\begin{aligned}\mathbf{r}' &= T(\mathbf{r}, \theta, S, \mathbf{X}, \{t\}) \\ &= \mathbf{X} + \begin{bmatrix} S\cos\theta & -S\sin\theta \\ S\sin\theta & S\cos\theta \end{bmatrix}(\mathbf{r} + \mathbf{u}),\end{aligned} \tag{6}$$

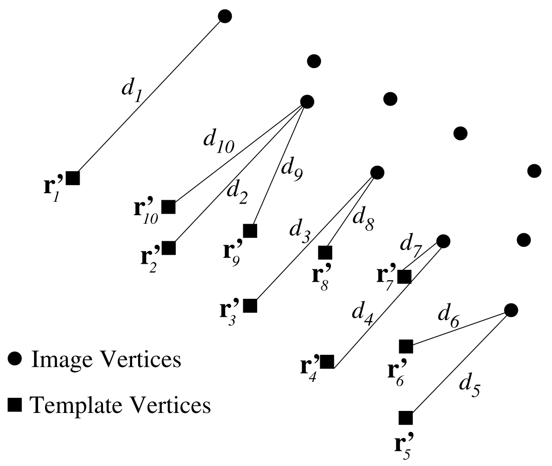

Fig. 2 Example of measuring error between a template and an image where d_i is the distance from the template vertex $\mathbf{r}'_i$ to the nearest image vertex. The error in this example is $E = \sum_{i=1}^{10} d_i^2$

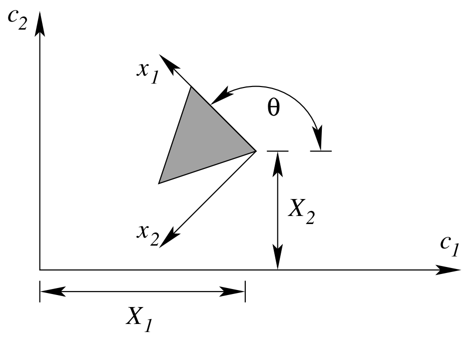

Fig. 3 Rigid body template degrees of freedom

where each template vertex $\mathbf{r}$ is translated by the vector $\mathbf{u}$ obtained using (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, S, and $\mathbf{X}$. The resulting error function is

$$E(\theta, S, \mathbf{X}, \{t\}) = \sum_{i=1}^{M} \|\mathbf{r}'_i - \mathbf{w}_i\|^2. \tag{7}$$

2.3 Minimizing the Error Function

The error function (7) is minimized by the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method, a gradient-based multi-variable minimization technique (Fletcher 1987). The BFGS method differs from the steepest decent method in that it 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.

3 The Boundary Element Method

As was mentioned in the previous section, the boundary element method is used to model the deformation of the template. 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 must be partitioned.

The linearly elastic 2D plane stress model is 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 (Sokolnikoff 1983)

$$\mu\nabla^2 u_\alpha + \mu\left(\frac{1+\nu}{1-\nu}\right)\frac{\partial}{\partial x_\alpha}\left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right) + F_\alpha = 0, \tag{8}$$

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 (8) 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 (8) into a boundary integral equation, it is first necessary to obtain the fundamental solutions for (8). The fundamental solutions can then be used to construct the boundary integral equation.

3.1 Fundamental Solutions

The fundamental solutions for the plane stress elasticity problem are the solutions to (8) 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 (Beer 2001)

$$U_{lk}(p,q) = C_1\left[C_2 \ln\frac{1}{r}\delta_{lk} + \frac{(p_l - q_l)(p_k - q_k)}{r^2}\right], \tag{9}$$

where

$$r = [(p_1 - q_1)^2 + (p_2 - q_2)^2]^{\frac{1}{2}},$$
$$C_1 = \frac{1}{8\pi\mu(1-\nu)},$$
$$C_2 = 3 - 4\nu.$$

δ_{lk} is the Kronecker delta, 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 Fig. 4 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

$$T_{lk}(p,q) = \frac{C_3}{r}\left[\frac{\partial r}{\partial n}\left(C_4\delta_{lk} + 2\frac{(p_l - q_l)(p_k - q_k)}{r^2}\right) + C_4\left(\frac{n_l(p_k - q_k)}{r} - \frac{n_k(p_l - q_l)}{r}\right)\right], \tag{10}$$

where

$$\frac{\partial r}{\partial n} = \frac{n_1(p_1 - q_1)}{r} - \frac{n_2(p_2 - q_2)}{r},$$

$$C_3 = \frac{1}{4\pi(1-\nu)},$$
$$C_4 = 1 - 2\nu$$

and n_l is the outward normal vector to the line l at the point q. The traction fundamental solution is shown graphically in Fig. 4 for the unit load $F = (1, 0)$. Note that the fundamental solutions are singular when $p = q$.

3.2 Boundary Integral Equations

The fundamental solutions (9) and (10) are used to construct the boundary integral equations for the elasticity problem (8). The integral equation that relates interior displacements to boundary displacements and boundary tractions is known as Somigliana's identity and is as follows (Rizzo 1967)

$$u_i(p) = \int_{\partial R} [U_{ij}(p,q)t_j(q) - T_{ij}(p,q)u_j(q)]\,\mathrm{d}\partial R(q) \quad (11)$$

for $\forall p \in R \backslash \partial R$, $i, j = 1, 2$, 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 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 (Rizzo 1967)

$$c_{ij}u_j(p) = \mathrm{PV}\int_{\partial R} [U_{ij}(p,q)t_j(q) - T_{ij}(p,q)u_j(q)]\,\mathrm{d}\partial R(q) \quad (12)$$

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 Sect. 3.3.

3.3 Partitioning of the Boundary Integral Equations

In order to numerically solve the boundary integral equation (12) it is necessary to partition the boundary of the object. The partitioned boundary is shown in Fig. 5. 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 (Beer 2001)

$$\phi_1(s) = \frac{1}{2}s(s-1),$$
$$\phi_2(s) = (1 - s^2), \quad (13)$$
$$\phi_3(s) = \frac{1}{2}s(s+1),$$

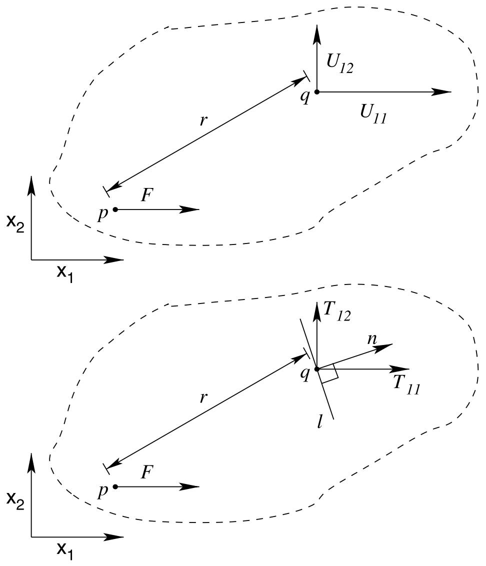

Fig. 4 Displacement fundamental solution on top and traction fundamental solution on bottom

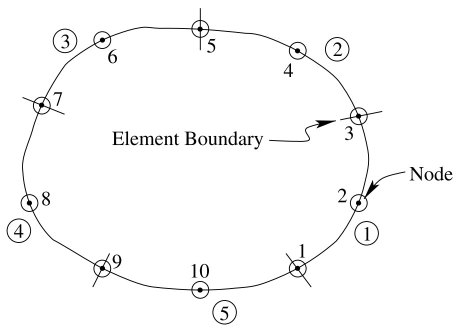

Fig. 5 Boundary mesh

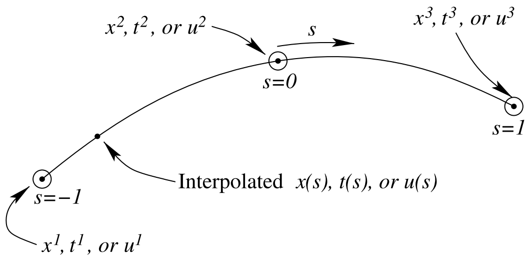

Fig. 6 Elemental interpolation of geometry x, displacements u, and tractions t

where s is a parameter that varies from -1 to 1 along the length of an element as shown in Fig. 6. The value of u at any point along an element is given by

$$u_i(s) = \sum_{j=1}^{3} \phi_j(s) u_i^j, \tag{14}$$

where the superscript j indicates the node number. An interpolation equation analogous to (14) 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 7 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, (12) is expressed in a form that can be numerically computed. The first step is to break the integral in (12) into N integrals, one for each element. Integrating over each element the following equation is obtained

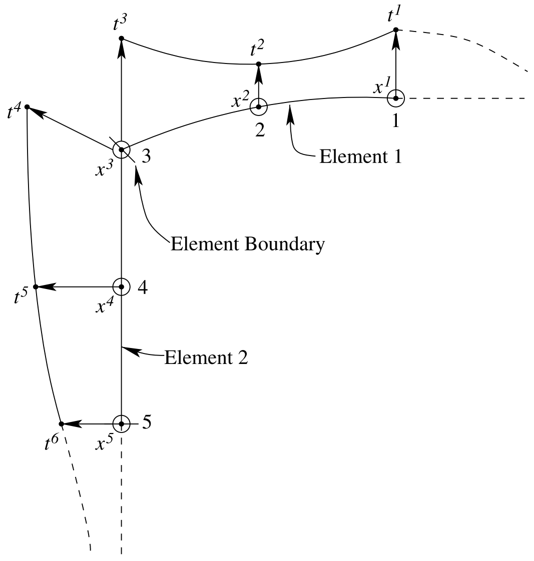

Fig. 7 Boundary tractions t can have discontinuities between elements while boundary geometry x must have $\mathcal{C}^0$ continuity between elements

$$c_{ij}u_j(p) = \sum_{k=1}^{N} \int_{-1}^{1} \left[U_{ij}(p, q^k(s)) \left(\sum_{n=1}^{3} \phi_n(s) t_j^{k,n} \right) - T_{ij}(p, q^k(s)) \left(\sum_{n=1}^{3} \phi_n(s) u_j^{k,n} \right) \right] J^k(s)\, \mathrm{d}s, \tag{15}$$

where

$$q_i^k(s) = \sum_{n=1}^{3} \phi_n(s) x_i^{k,n},$$

$$J^k(s) = \frac{\mathrm{d}\partial R}{\mathrm{d}s} = \sqrt{\left(\sum_{n=1}^{3} \frac{\partial \phi_n(s)}{\partial s} x_1^{k,n} \right)^2 + \left(\sum_{n=1}^{3} \frac{\partial \phi_n(s)}{\partial s} x_2^{k,n} \right)^2}.$$

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

$$c_{ij}u_j(p) = \sum_{k=1}^{N} \sum_{n=1}^{3} t_j^{k,n} \int_{-1}^{1} U_{ij}(p, q^k(s)) \phi_n(s) J^k(s)\, \mathrm{d}s - \sum_{k=1}^{N} \sum_{n=1}^{3} u_j^{k,n} \int_{-1}^{1} T_{ij}(p, q^k(s)) \phi_n(s) J^k(s)\, \mathrm{d}s. \tag{16}$$

Switching to matrix notation

$$\mathbf{cu}(p)+\sum_{k=1}^{N}\sum_{n=1}^{3}\Delta\mathbf{T}_n^k\mathbf{u}_n^k=\sum_{k=1}^{N}\sum_{n=1}^{3}\Delta\mathbf{U}_n^k\mathbf{t}_n^k, \tag{17}$$

where

$$\begin{aligned}\Delta\mathbf{U}_n^k&=\textstyle\int_{-1}^{1}\phi_n(s)\mathbf{U}(p,q^k(s))J^k\,\mathrm{d}s,\\ \Delta\mathbf{T}_n^k&=\textstyle\int_{-1}^{1}\phi_n(s)\mathbf{T}(p,q^k(s))J^k\,\mathrm{d}s\end{aligned} \tag{18}$$

and $\mathbf{c}$, $\Delta\mathbf{T}_n^k$, and $\Delta\mathbf{U}_n^k$, are 2×2 matrices and $\mathbf{u}_n^k$ and $\mathbf{t}_n^k$ 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, (17) is applied to all $2N$ nodes in the boundary element mesh to obtain $2N$ vector equations ($4N$ scalar equations) that can be used to solve the elasticity problem. These equations are

$$\mathbf{cu}(p_i)+\sum_{k=1}^{N}\sum_{n=1}^{3}\Delta\mathbf{T}_n^k\mathbf{u}_n^k=\sum_{k=1}^{N}\sum_{n=1}^{3}\Delta\mathbf{U}_n^k\mathbf{t}_n^k, \tag{19}$$

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

$$[I]\{cu\}+[F]\{u\}=[G]\{t\} \tag{20}$$

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 (18). 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 term $[F]\{u\}$ to obtain the following equation

$$[F']\{u\}=[G]\{t\}, \tag{21}$$

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 (9). 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 (Galassi et al. 2001).

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 (10) 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 (Beer 2001). 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

$$\begin{aligned}&[F']\{1\,0\,1\,0\cdots1\,0\}^{\mathrm{T}}=0,\\ &[F']\{0\,1\,0\,1\cdots0\,1\}^{\mathrm{T}}=0.\end{aligned} \tag{22}$$

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 (20) 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 (21) can be used to calculate the nodal tractions $\{t\}$. For the Neumann problem, the nodal tractions $\{t\}$ are known and (21) can be used to calculate the nodal displacements $\{u\}$. For the deformable template application, the Neumann problem is the problem of interest since it is necessary to apply a force distribution to the template and then calculate the resulting nodal displacements.

3.4 Solution of the Neumann Problem

The Neumann problem is defined as the case where the surface tractions $\{t\}$ are known and the surface tractions $\{u\}$ must be computed using (21). It can be shown (Jaswon and Symm 1977) that a solution to the Neumann problem exists if and only if

$$\int_{\partial R} t_i(q)\,\mathrm{d}\partial R(q)=0, \tag{23}$$

where $i=1,2$. An object that satisfies (23) is said to be in a state of static equilibrium. If (23) is satisfied then there are an infinite number of solutions $\{u\}$ to (21). 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 three possible 2D rigid body motions. Since $[F']$ is not invertible, singular value decomposition (SVD) is used to solve (21). SVD decomposes $[F']$ into the following form (Press et al. 1993)

$$[F'] = [U][W][V]^{\mathrm{T}}, \tag{24}$$

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 the following equation (Press et al. 1993)

$$\{u\} = [V][\operatorname{diag}(1/w_j)][U]^{\mathrm{T}}[G]\{t\}, \tag{25}$$

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 $[A] = [V][\operatorname{diag}(1/w_j)][U]^{\mathrm{T}}[G]$, (25) can be rewritten as

$$\{u\} = [A]\{t\}, \tag{26}$$

where $[A]$ only depends on the object's geometry and 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 (13).

4 The BEM Deformable Template Matching Algorithm

4.1 The Error Function for BEM Deformable Object Tracking

BEM is 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 (6) becomes

$$\begin{aligned}\mathbf{r}' &= T(\mathbf{r}, \theta, S, \mathbf{X}, \{t\}) \\ &= \mathbf{X} + \begin{bmatrix} S\cos\theta & -S\sin\theta \\ S\sin\theta & S\cos\theta \end{bmatrix}(\mathbf{r}+\mathbf{u}),\end{aligned} \tag{27}$$

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 (26). The error function (7) becomes

$$E(\theta, S, \mathbf{X}, \{t\}) = \sum_{i=1}^{M} \|\mathbf{r}'_i - \mathbf{w}_i\|^2. \tag{28}$$

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 (23). Therefore, the minimization of (28) is a constrained minimization problem. The next section demonstrates how to convert this constrained minimization problem into an unconstrained minimization problem of reduced dimensionality.

4.2 Constrained Minimization

The Neumann problem can only be solved if the equilibrium condition (23) is satisfied. Therefore, when (28) is minimized, $\{t\}$ must be constrained to satisfy (23). A method is presented in this section that solves the constrained minimization problem efficiently.

The first step is to express (23) in discrete form. The equilibrium condition can be partitioned into sums of integrals over each of the N elements as follows:

$$\sum_{k=1}^{N}\sum_{n=1}^{3} \Delta\mathbf{P}_n \mathbf{t}_n = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{29}$$

where

$$\Delta\mathbf{P}_n = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \int_{-1}^{1} \phi_n(s) J(s)\, \mathrm{d}s. \tag{30}$$

This discrete form can now be written as a matrix equation

$$[P]^{\mathrm{T}}\{t\} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{31}$$

where $[P]$ is a $6N \times 2$ matrix. The above constraint equation is a linear equality constraint. The Generalized Elimination Method as described by Fletcher (1987) is used to eliminate two variables from the optimization problem by applying a linear transformation. Applying this transformation, the minimization problem (28) is recast in the form:

$$E(\theta, S, \mathbf{X}, [Z]\{y\}) = \sum_{i=1}^{M} \|\mathbf{r}'_i - \mathbf{w}_i\|^2, \tag{32}$$

where $\{Z\}$ is a $6N \times (6N-2)$ matrix. Minimizing (32) gives values for θ, S, $\mathbf{X}$ and $\{y\}$ that best match the image in a least squares fashion. The traction distribution is 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 $[P]$ is written as

$$[P] = [Q]\begin{bmatrix} R \\ 0 \end{bmatrix} = [Q_1 Q_2]\begin{bmatrix} R \\ 0 \end{bmatrix} = [Q_1][R], \tag{33}$$

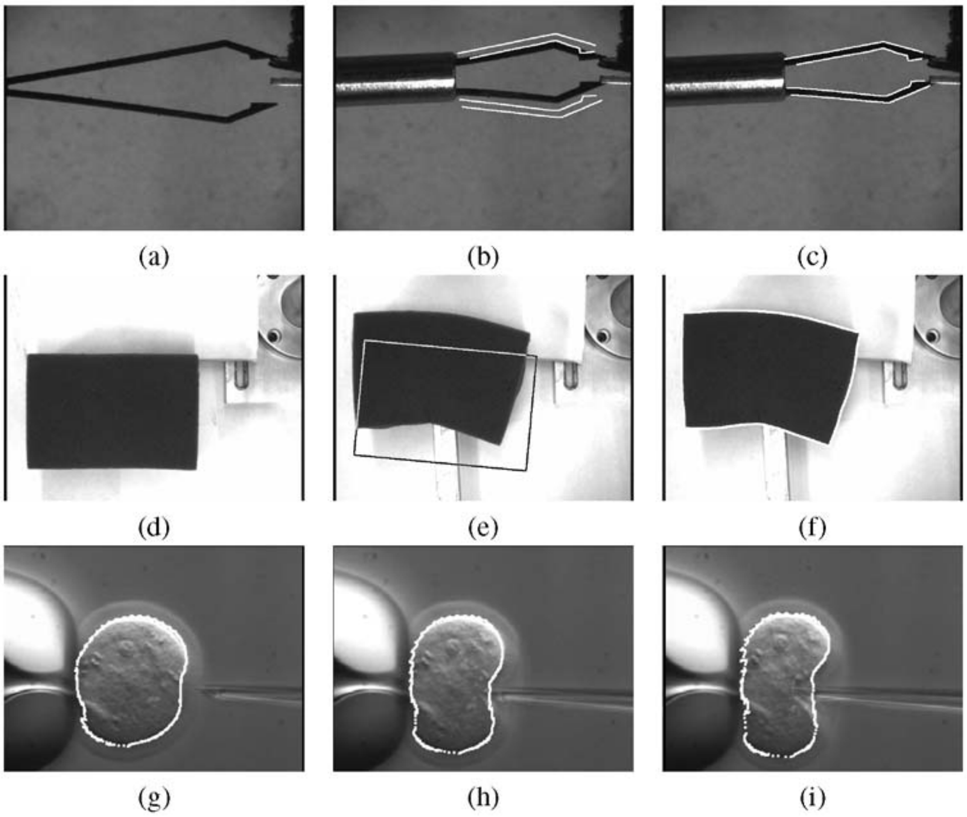

Fig. 8 (**a**–**c**) BEM deformable template results when applied to a microgripper, (**d**–**f**) to a foam block, and (**g**–**i**) to a mouse oocyte (approximately 50 µm in diameter in the undeformed state). Images (**b**) and (**e**) show the initial position and shape of the template used to obtain the tracking solutions shown in images (**c**) and (**f**) respectively

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 is now in an unconstrained form of reduced dimensionality, (32), that can be minimized using standard gradient based minimization techniques.

5 Initial BEM Template Matching Results

The results of the BEM template matching algorithm applied to a microgripper, a foam block, and a mouse embryo cell are shown in Fig. 8. In each of the tracking examples, the initial traction distribution is taken to be zero. The microgripper in Figs. 8(a–c) is used for robotic micro-assembly tasks. A 90 node BEM mesh is used to model the gripper and is shown in Fig. 9. Figure 8(a) shows the undeformed gripper, Fig. 8(b) shows the template's initial location and shape, and Fig. 8(c) shows the tracking solution. 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 (Sect. 6 describes a method to handle this occlusion in a robust way without altering 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.

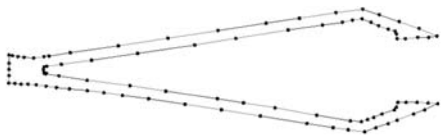

Fig. 9 BEM mesh used to track microgripper

Figures 8(d–e) shows the template registered to a foam block. The foam block registration example demonstrates the ability of the BEM mesh to accommodate large deformations. The modulus of elasticity used by the BEM model was set to a relatively low value for this example, and the cell example that follows, to allow for the large deformations.

Figures 8(g–i) show three frames from the tracking of a mouse oocyte. The cell is being robotically injected (Sun and Nelson 2002). 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.

Figure 10 shows an example of the BEM deformable object tracking algorithm applied to computer generated image. Figure 10(a) shows the tracking solution. The undeformed boundary element mesh is shown in Fig. 10(b) and the traction distribution that was obtained by minimizing (32) is shown in Fig. 10(c).

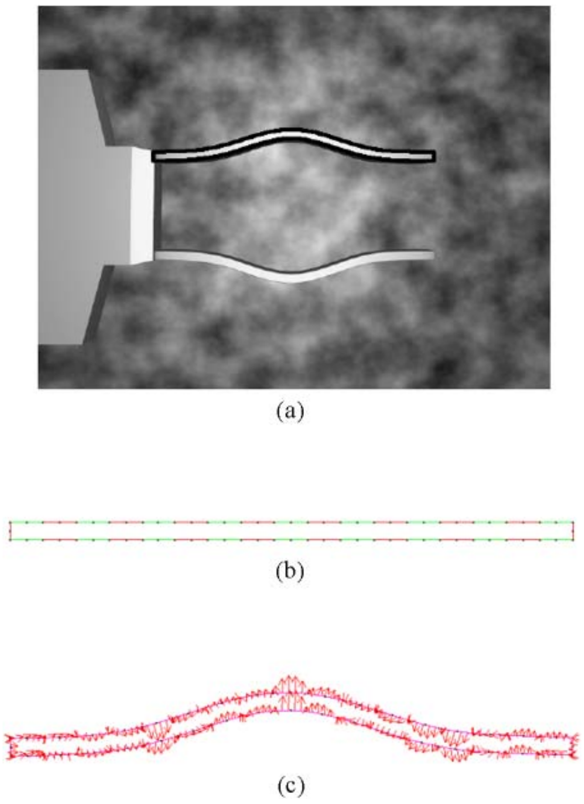

Fig. 10 (**a**) BEM tracking solution, (**b**) undeformed mesh, and (**c**) the traction distribution $\{t\}$ obtained by minimizing the error function

6 Deformable Object Tracking Robust to Occlusions and Spurious Edges

In order for the deformable object tracking algorithm to be a useful feedback mechanism 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 section presents modifications to the BEM deformable object tracking algorithm that make it robust to spurious edges and occlusions. 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 classifier. The performance of each of these improvements is demonstrated for the problem of tracking a gripper in a highly cluttered environment in Sect. 7. Figure 11 shows the BEM tracking algorithm applied to this microgripper.

6.1 Robustness to Occlusion

The least squares error measure used to obtain (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 (Draper 1998). 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 Fig. 12(b).

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

$$\rho(r) = \log\left(1 + \frac{1}{2}r^2\right) \tag{34}$$

where r is the error, has robust properties particularly relevant to this problem (Stewart 1999). 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 (28) becomes

$$E(\theta, S, \mathbf{X}, \{t\}) = \sum_{i=1}^{M} \log\left(1 + \frac{1}{2}\|\mathbf{r}_i' - \mathbf{w}_i\|^2\right). \tag{35}$$

This error measure was chosen both for its robust properties and because it has a defined derivative for all values of $\|\mathbf{r}_i' - \mathbf{w}_i\|^2$. This is a required property of the error function since a gradient based minimization algorithm is used to minimize the error function. Using the Cauchy 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 Fig. 12(c).

The Cauchy error measure has better performance in this situation because the Cauchy 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 Fig. 13 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.

6.2 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.

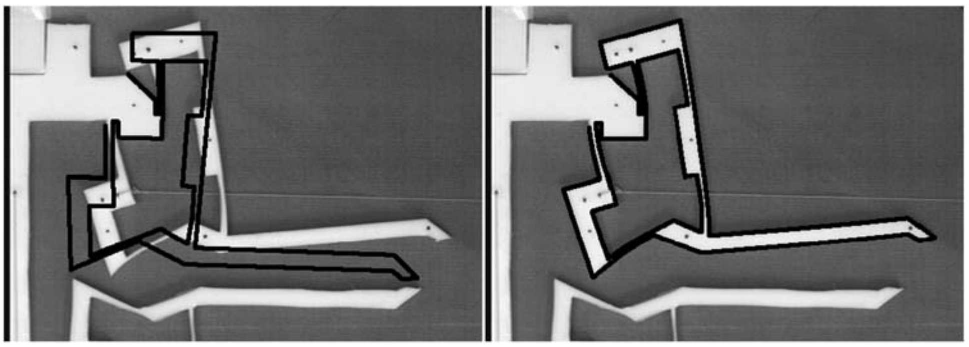
Fig. 11 Deformable object tracking example using BEM to model deformations. The template's initial condition is shown on the *left* and the tracking solution is shown on the *right*

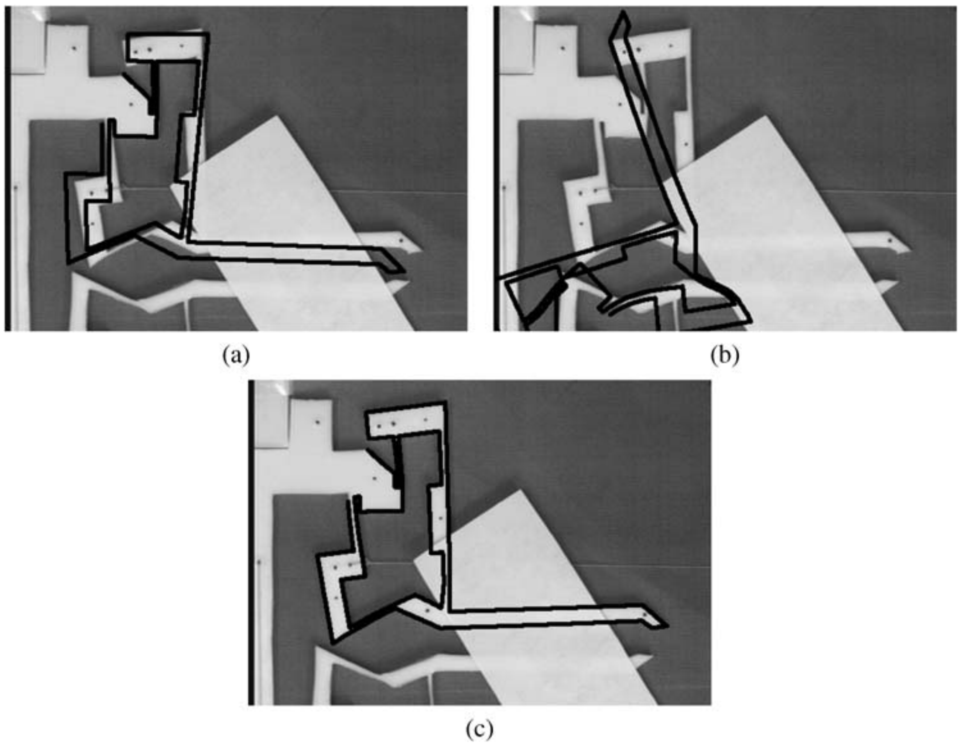

Fig. 12 Effect of error measure on tracking results when tracking a partially occluded object. (**a**) The initial template location and shape for both algorithms, (**b**) the tracking solution when using the least squares error measure, and (**c**) the tracking solution when using the Cauchy error measure

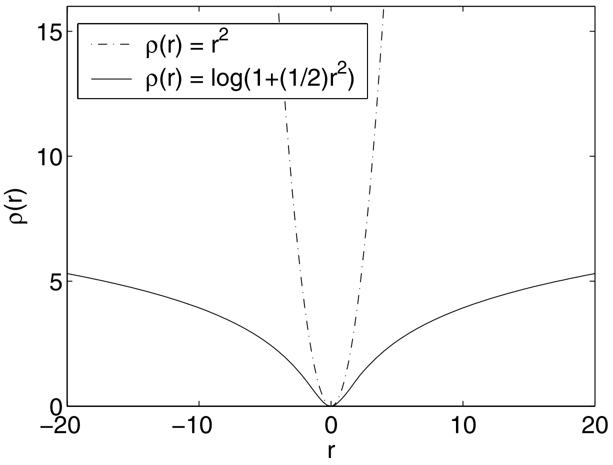

Fig. 13 Comparison between least squares and Cauchy error measures

Figure 14 illustrates the problem. Figure 14(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 Fig. 14(b) and Fig. 14(c) shows only the edges from the object that is to be tracked. In order to obtain the edge image shown in Fig. 14(c) it is necessary to probe the pixel values on either side of the edges shown in Fig. 14(b) to determine if the edge in question bounds the object being tracked. For example, in Fig. 14, 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

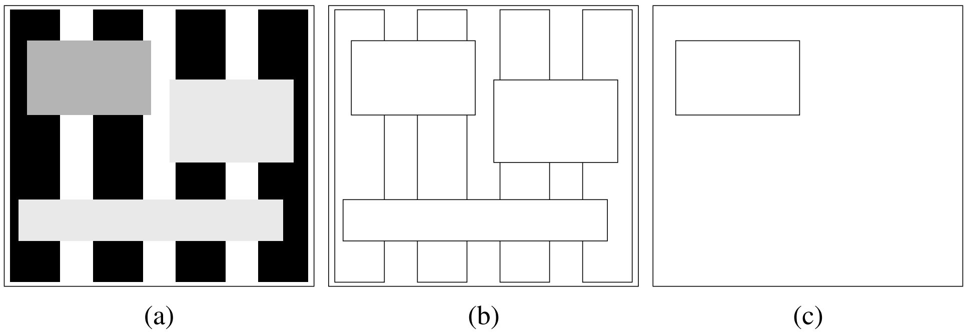

Fig. 14 Edge detection in a cluttered scene: (**a**) original scene, (**b**) edges in the scene, and (**c**) edges from the object of interest

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:

$$\mathbf{n}_i = \frac{\nabla \mathbf{I}_i}{\|\nabla \mathbf{I}_i\|} \tag{36}$$

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 15 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. Anywhere from five to twenty edge samples were used for the results presented in this article.

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 $f(\mathbf{S}_i^+, \mathbf{S}_i^-, \omega) = 1$ and is otherwise rejected. The vector ω represents the parameters that define the classifier. Two forms for f are used in this article: 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.

Both approaches discussed in this section assume that the intensity of the pixels representing the interior of the object being tracked can be distinguished from the intensity of those that represent the background of the image. The interior intensity based classifier has the additional assumption that the intensity of the object being tracked is uniform. The neural network classifier does not require a uniform interior intensity of the object being tracked since the neural network

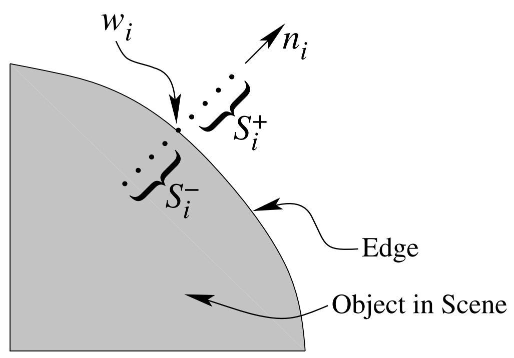

Fig. 15 Edge samples used to test whether or not to keep an edge pixel

is able to classify multiple input edge normal vector intensities as being part of the object of interest. This property of the neural network edge classifier is due to the general approximation property of neural networks.

6.2.1 *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 defined as

$$f(\mathbf{S}_i^+, \mathbf{S}_i^-, \omega) = \begin{cases} 1 & \text{if } \|\mathbf{S}_i^+ - \mathbf{C}\| < H, \\ & \text{or } \|\mathbf{S}_i^- - \mathbf{C}\| < H, \\ 0 & \text{otherwise.} \end{cases} \tag{37}$$

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 Fig. 16(c). The original Canny edges are shown in Fig. 16(b). It can be seen from Fig. 16(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. Five edge normal samples were used.

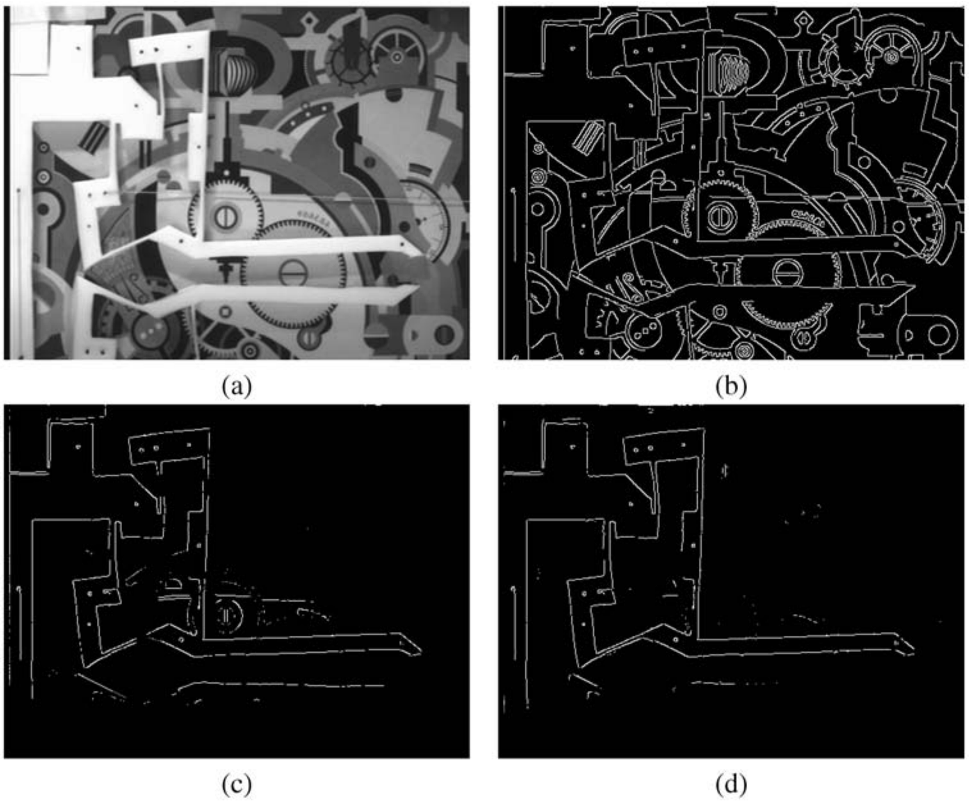

Fig. 16 Comparison of edge detection techniques discussed in the text. (**a**) The original image, (**b**) Canny edge detection algorithm, (**c**) edge detection example using only object intensity to classify edges, and (**d**) edge detection example using the neural network edge classifier

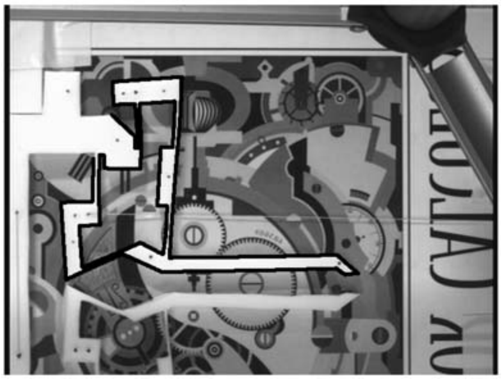

Fig. 17 One of the training images used to train the neural network edge classifier

6.2.2 *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

$$f(\mathbf{S}_i^+, \mathbf{S}_i^-, \omega) = \begin{cases} 1 & \text{if NeuralNet}(\mathbf{S}_i^+) > 0.5, \\ & \text{or NeuralNet}(\mathbf{S}_i^-) > 0.5, \\ 0 & \text{otherwise.} \end{cases} \quad (38)$$

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 BEM tracking algorithm described in this paper is used to determine which edges in the training image correspond with the object of interest. Figure 17 shows the BEM tracking solution for one of the training images. 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 neural network model. For the results presented here, the neural network model has 10 hidden nodes, 6180 training pairs were used (2060 sample vectors corre-

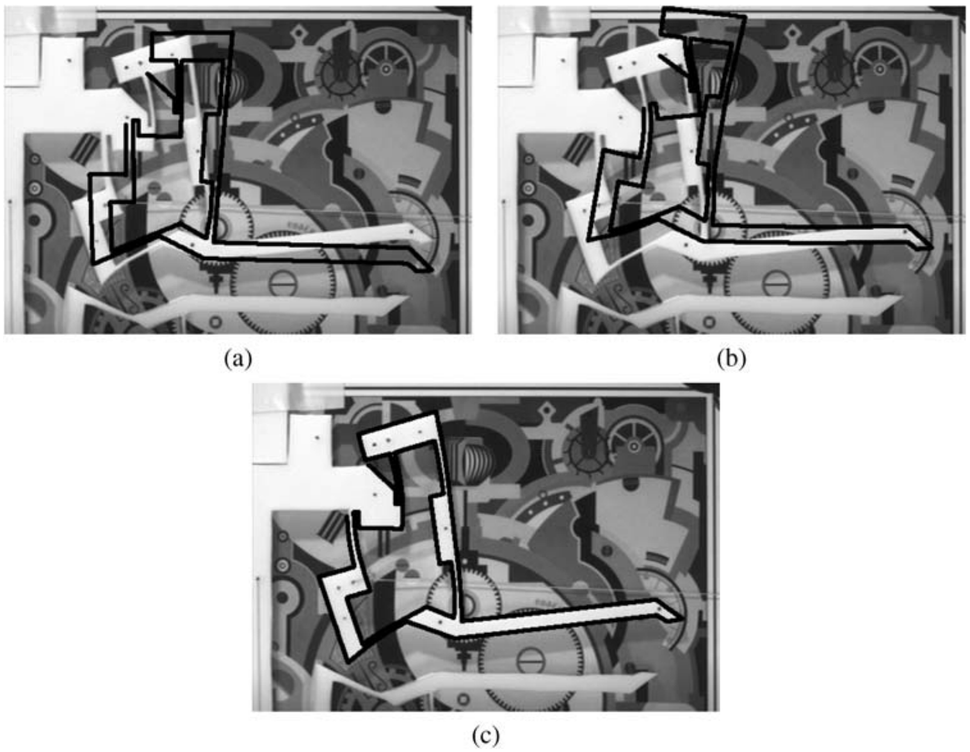

Fig. 18 Effect of edge detection algorithm on tracking results for a cluttered scene. (**a**) The initial template shape and location for both methods, (**b**) the tracking solution when using the Canny edge detector, and (**c**) the tracking result when using the neural network edge classifier

sponding to an actual edge and 4120 samples corresponding to a false edge), and five edge normal samples were used. Figure 16(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 18(c) shows the successful tracking solution for a scene with a large number of spurious edges. This method for the suppression of spurious edges does have the drawback of increased computation time over the standard Canny edge detection algorithm. For the image shown in Fig. 16(a), the Canny edge detector takes approximately 0.02 seconds to process the entire 640×480 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.

7 Performance Analysis

7.1 Quantitative Performance Comparison of Proposed Algorithms

The performance of the tracking algorithms was evaluated quantitively in order to measure their robustness to occlusions and spurious edges. Five fiducial marks where placed on a compliant 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 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 Figs. 11, 12, and 18, respectively, for example images from each set). Each algorithm was tested on a series of eight images where the initial template location was manually set for the first tracking frame and the tracking algorithm was used to track from frame-to-frame for each subsequent frame. 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 for all eight images in the sequence. The errors are measured in pixels.

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

Table 1 Quantitive measure of the performance of the algorithms introduced in this article. Error values are measured in pixels

	Control Image Set	Occlusion Image Set	Spurious Edge Image Set
Least Squares Error Canny (LSE-C)	1.011	164.690	32.965
Cauchy Error Canny (CE-C)	1.006	1.067	2.985
Cauchy Error Canny+Intensity Classifier (CE-C+IntC)	1.033	1.158	1.088
Cauchy Error Canny+Neural Network Classifier (CE-C+NNC)	1.028	1.096	1.065

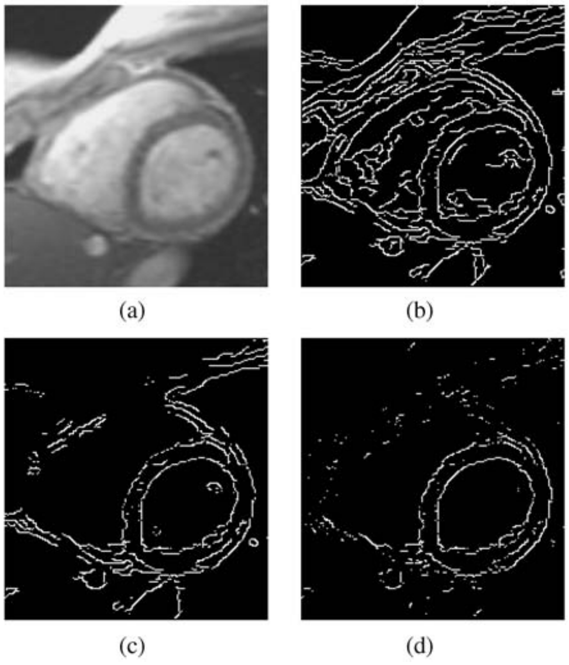

Fig. 19 Comparison of edge detection techniques applied to a cardiac MRI image. (**a**) The original image, (**b**) Canny edge detection algorithm, (**c**) edge detection using the intensity based edge classifier, and (**d**) edge detection using the neural network edge classifier

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 Cauchy error measure with either the intensity edge classifier or the neural network edge classifier (the CE-C+IntC and CE-C+NNC algorithms) have performance levels that nearly match the performance of the LSE-C algorithm applied to the control set.

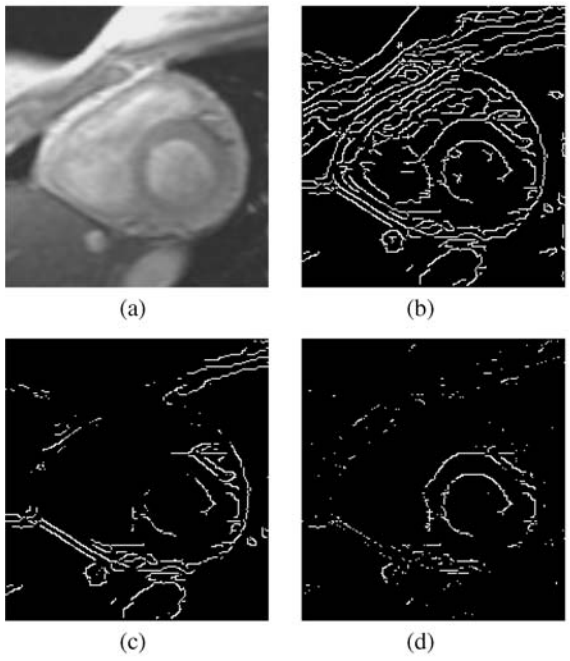

Fig. 20 Comparison of edge detection techniques applied to a cardiac MRI image with weak edges. (**a**) The original image, (**b**) Canny edge detection algorithm, (**c**) edge detection using the intensity based edge classifier, and (**d**) edge detection using the neural network edge classifier

7.2 Tracking Performance for a Sequence of Cardiac MRI Images

The methods presented in the paper were also applied to the tracking of cardiac MRI data.[1] Figure 19(a) shows an example image from the set. The image plane is oriented so that it shows the cross section of the left ventricle, which is the circular chamber on the right side of the image. In this example, the algorithm presented in the article will be used to track the muscle wall of the left ventricle throughout its stroke. By tracking the left ventricle, the volume of the left ventricle throughout its cycle can be approximated which in turn can be used to compute the ejection fraction. The ejection fraction is a measure of the level of cardiac function.

Figures 19 and 20 show the application of the edge classifier algorithms presented in this article on two different images. The edges in Fig. 19 are fairly strong and, as a result, the two edge classifier algorithms have similar performance. The edges of Fig. 20, in contrast, are relatively weak and the difference in performance between the two edge classifier algorithms becomes significant. The neural network edge classifier algorithms preserves the desired edges that

[1]The cardiac MRI images used in this paper were provided by Brian Mullan, M.D., of the Division of Diagnostic Radiology at the University of Iowa.

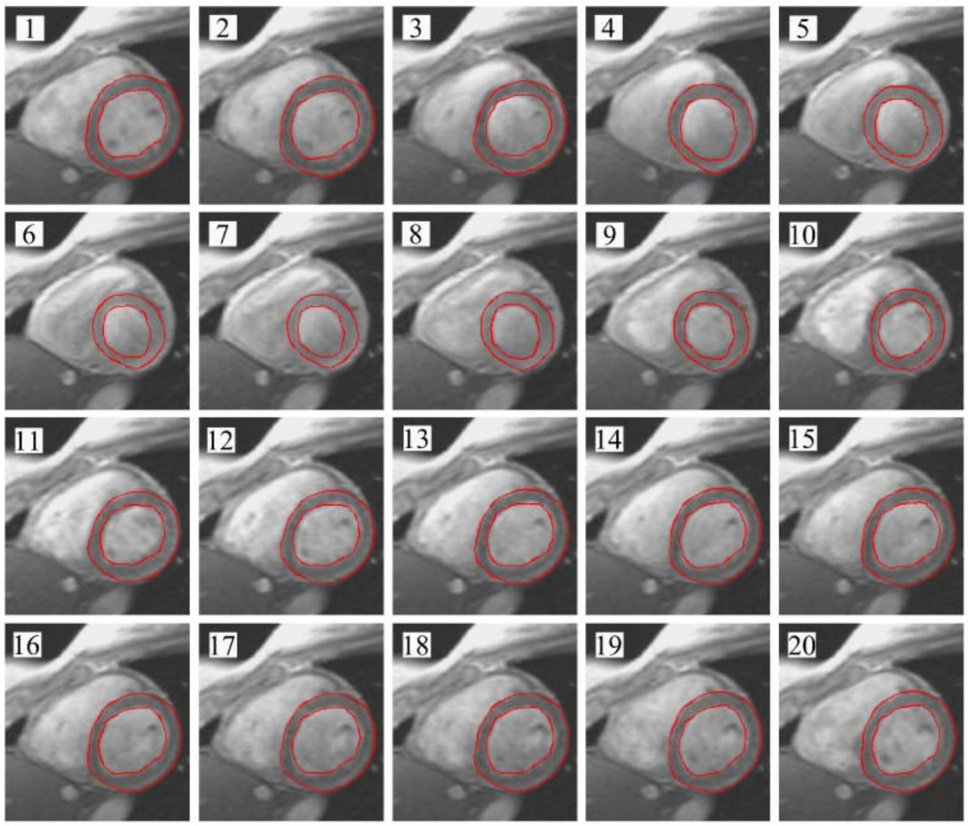

Fig. 21 Results of applying the CE-C+NNC tracking algorithm to a sequence of cardiac MRI images

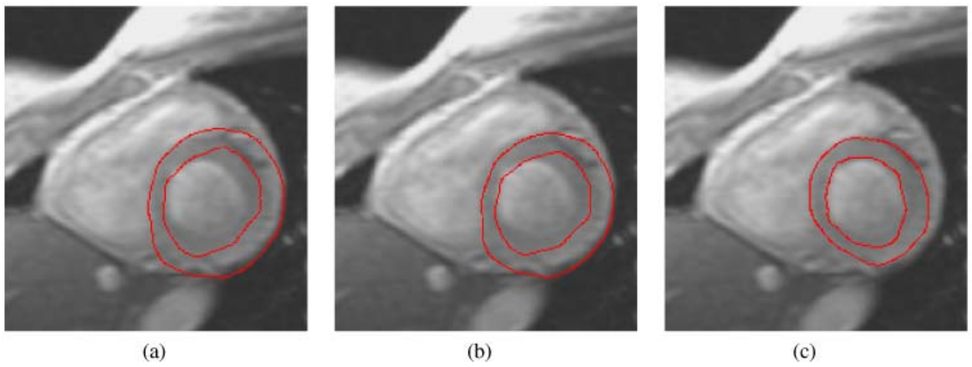

Fig. 22 Comparison of tracking results for the eighth frame of the tracking sequence shown in Fig. 21. (**a**) Tracking result using Canny edge detection algorithm (CE-C), (**b**) tracking result using the intensity based edge classifier (CE-C+IntC), and (**c**) tracking result using the neural network edge classifier (CE-C+NNC)

are present while most of the undesired edges are rejected. Even though some of the edges are too weak to be detected by the Canny edge operator, the use of the Cauchy error measure allows the tracking to be performed successfully as will be seen in Fig. 21 (Figs. 19 and 20 are frames 13 and 8 respectively from the tracking sequence in Fig. 21).

Figure 21 shows the boundary element deformable object tracking algorithm applied to a sequence of cardiac MRI images. The Cauchy error measure is used in conjection with the neural network edge classifier (the CE-C+NNC algorithm). The deformable template was manually registered to the first frame and the tracking algorithm was used to advance the tracking solution for all subsequent frames. The neural network edge classifier has 20 hidden nodes and uses 20 edge normal samples to classify edge pixels.

The same sequence of MRI images was also tracked using just the Canny edge operator (the CE-C algorithm) and the intensity based edge classifier (the CE-C+IntC algorithm). Figure 22 compares the tracking solution of the three methods for the eighth frame in the tracking sequence. For each algorithm, the same initial template placement was used for the first frame and the respective tracking algorithm was used to advance the tracking solution for each subsequent frame. For the Canny edge operator and the intensity based edge classifier, the template was pulled off of the true tracking solution by spurious edges. The neural network edge classifier algorithm provided the best tracking solution because it was able to suppress most of the spurious edges (see Fig. 20 for a comparison of this frame's edge images as provided by each edge detection algorithm).

8 Summary and Conclusions

A deformable object tracking algorithm based on the boundary element method and robust to occlusion and to spurious edges was presented. Since the algorithm uses a physics based deformation model it is able to more robustly track elastic objects when compared to a general deformation model. In addition, 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 compliant gripper and by tracking a sequence of cardiac MRI images.

Robust deformable tracking is essential for the successful manipulation of deformable objects as is required in the field of medical robotics or in the field of microrobotics. Robust deformable object tracking can also be used to provide force feedback when the material properties of the object being manipulated are known. Deformable object tracking provides a means to obtain rich feedback information when the use of other sensing technologies is limited or impossible.

Acknowledgements I would like to thank Dr. Brian Mullan of the Division of Diagnostic Radiology at the University of Iowa for providing the Cardiac MRI images. This research was supported in part by the US National Science Foundation through grant numbers IIS-9996061 and IIS-0208564. Michael Greminger was supported by the Computational Science Graduate Fellowship (CSGF) from the US Department of Energy.

References

Beer, G. (2001). *Programming the boundary element method*. New York: Wiley.

Canny, J. (1986). A Computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*(6).

Draper, N. R. (1998). *Applied regression analysis* (3rd ed.). New York: Wiley.

Fletcher, R. (1987). *Practical methods of optimization*. New York: Wiley.

Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Booth, M., & Rossi, F. (2001). *GNU scientific library reference manual*. Bristol: Network Theory Limited.

Greminger, M. A., & Nelson, B. J. (2004). Vision-based force measurement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(3), 290–298.

Jaswon, M. A., & Symm, G. T. (1977). *Integral equation methods in potential theory and elastostatics*. New York: Academic.

Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: active contour models. *International Journal of Computer Vision*, 321–331.

McInerney, T., & Terzopoulos, D. (1993). A finite element model for 3D shape reconstruction and nonrigid motion tracking. In *Proceedings fourth IEEE international conference on computer vision* (pp. 518–523), April 1993.

Metaxas, D. (1997). *Physics-based deformable models*. Boston: Kluwer Academic.

Nakamura, Y., Kishi, K., & Kawakami, H. (2001). Heartbeat synchronization for robotic cardiac surgery. In *IEEE international conference on robotics and automation (ICRA2001)* (Vol. 2, pp. 2014–2019). Seoul, Korea, May 2001.

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1993). *Numerical recipes in C: the art of scientific computing*. Cambridge: Cambridge University Press.

Rizzo, F. J. (1967). An integral equation approach to boundary value problems of classical elastostatics. *Quarterly Applied Mathematics*, *25*, 83–95.

Sokolnikoff, I. (1983). *Mathematical theory of elasticity*. Malabar: Krieger.

Stewart, C. V. (1999). Robust parameter estimation in computer vision. *SIAM Review*, *41*(3), 513–537.

Sun, Y., & Nelson, B. J. (2002). Biological cell injection using an autonomous microrobotic system. *The International Journal of Robotics Research (IJRR)*, *21*(10–11), 861–868.

Tsap, L., Goldgof, D., & Sarkar, S. (1998). Efficient nonlinear finite element modeling of nonrigid objects via optimization of mesh models. *Computer Vision and Image Understanding*, *69*, 330–350.

Wang, X., Ananthasuresh, G. K., & Ostrowski, J. P. (2001). Vision-based sensing of forces in elastic objects. *Sensors and Actuators A-Physical*, *94*(3), 142–156.

Yuille, A., Cohen, D., & Hallinan, W. (1992). Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, *8*(2), 99–111.