Using Active Deformable Models in Robotic Visual Servoing

Michael J. Sullivan
sullivan@cs.umn.edu

Nikolaos P. Papanikolopoulos
npapas@cs.umn.edu

Rahul Singh
singh@cs.umn.edu

Ioannis Pavlidis
pavlidis@cs.umn.edu

Artificial Intelligence, Robotics, and Vision Laboratory
Department of Computer Science
University of Minnesota
4-192 EE/CS Building
200 Union Street SE
Minneapolis, MN 55455

September 9, 1997

Submitted to the book
"Event-Based Sensing, Planning and Control
of a Robotic System: An Integrated Approach"

Abstract

The potential of visual servoing systems using eye-in-hand cameras has been demonstrated by many research efforts. However, previous efforts have generally used one of only two approaches to the extraction of a error signal from the visual input: blob analysis or pixel-level feature tracking. In this chapter, we describe a third approach which combines some of the advantages of both previous methods. We use deformable active models to track image contours related to the object of interest. These contour models provide global information about the position of the object. In addition, by combining such models with *a priori* knowledge of the object shape, this approach may be extended to provide the orientation of the object in three-dimensions.

We present a model-based approach for visual tracking and eye-in-hand robotic visual-servoing. Our approach uses active deformable models to track a rigid or a semi-rigid object in the manipulator's workspace. These deformable models (also known as "snakes") approximate the contour of the object boundary, defined by a set of control points. During tracking, the control points are updated at frame rates by minimizing an energy function involving the relative position of model points, image data, and the characteristics of figure pixels. When visual servoing is combined with the use of active deformable models, movement of the manipulator can compensate for translations and deformations of the object's image. To verify the potential of our approach, we run several experiments and present here our findings.

1 Introduction

Robotic systems which operate in uncalibrated and/or uncontrolled environments must be able to react flexibly to changes in their environment. In the simplest case, such changes may be the result of the movement of a single object in the manipulator's workspace. Previous work [1, 10, 13] has demonstrated that such changes can be handled effectively by incorporating information from eye-in-hand visual sensors into the manipulator's feedback loop. However, these systems sometimes have difficulty tracking targets which are semi-rigid or partially occluded. We propose to overcome these difficulties by incorporating active deformable models (commonly referred to as "snakes") into the visual system. Active deformable models attempt to conform to contours in the image as defined by the intensity gradient which correspond to the boundaries of the object being tracked. As the object contours translate or deform, the parameters of the active deformable model are adjusted. Simultaneously, the parameters of the active deformable model can be used as inputs to the manipulator controller.

The system presented in this chapter combines recent work from two different streams of research in the computer vision and robotics communities to improve the performance of eye-in-hand manipulators tracking moving objects. We have incorporated active deformable models into the established visual-servoing paradigm. By combining the two methods, we strengthen both. The use of visual servoing to produce compensating movements of the end-effector reduces the amount of deformation required from active deformable models to cope with object movements. At the same time, active deformable models allow manipulators controlled by visual servoing to deal with semi-rigid objects and motions (such as rotations) which challenge feature-based approaches to tracking (see Chapters 2 and 3 in this section of the book).

Recent work in visual servoing has demonstrated the benefit of "closing the control loop" of a robotic manipulator guided by an "eye-in-hand" visual sensor. Generally, systems designed using this technique seek to hold an aspect of the visual input invariant through appropriate movements of the manipulator on which the camera is mounted. For example, Papanikolopoulos *et al.* [20] identify one or more features in the image and seek to maintain the features' locations in the image plane by producing compensating translations and rotations of the "end-effector." Visual servoing obviates the need to maintain a detailed, metric workspace model. Rather than constructing

a model of effector and target positions and computing a trajectory which matches that of the target, the system reacts directly to information provided by the sensor during the last control iteration.

In this chapter, we use active deformable models to provide the control signal to a visual servoing system. The organization of the chapter is as follows: First, Section 2 highlights the importance of the problem. Then, Section 3 describes the issues and motivations for using this approach. Section 4 presents some previous work conducted in the area. Section 5 discusses the approach we propose. In this section, we also present an algorithm for the automatic selection of control points. In Section 6, we describe the hardware used to implement our experimental system. In Section 7, we present experimental results and in Section 8 these results are discussed. Finally, in Section 10, we conclude and highlight the contributions made by this work.

2 Importance of the Visual Servoing Problem

Vision-based control and active vision can have a significant impact on space applications, intelligent highways, manufacturing, and nuclear waste clean-up efforts. Vision-based control can enhance the performance of industrial robots in assembly lines, aid in better alignment of an object with the camera in automatic inspection systems, improve the automatic assembly of electronic devices (surface mount technology), assist in the realization of vehicle following (platooning), make possible autonomous satellite docking and recovery, and improve the efficiency of outdoor navigation techniques.

One area where robotic devices enhanced with sensing capabilities can have a significant impact is the area of nuclear waste clean-up. In particular, autonomous or semi-autonomous robotic devices can participate in the inspection of waste storage tanks, detect and remove buried waste, automate the handling and analysis of contaminants, and help in decontamination and decommissioning operations. Moreover, the manipulator is a useful tool since it intervenes between the hazardous environment and the human operator. Since the human operator does not have any direct view of the environment where the task takes place, sensing devices must be used in order to provide some information about the status of the robotic task and the environment. Thus, in order to improve the efficiency of robotic devices in hazardous sites, it is important to augment them with sensing devices. Among the sensing devices, visual sensors play a critical role. The primary advantage of vision

sensors is their ability to provide information on relatively large regions of the workspace. This same ability, however, presents problems that must be overcome. Noise, time-consuming image processing, and large amounts of visual information make their use challenging. Therefore, the modeling and design of robotic devices that include visual sensing has become a difficult and challenging task. These difficulties increase if we include many different sensing modules in the same feedback loop.

It is important to mention that currently no framework covers all the issues that are introduced by integrating the vision sensor or any sensor, in the feedback loop of a robotic device. We think that there is a significant waste due to the fact that there is a trend to build systems that only address the use of specific sensing modules in the feedback loop. Small changes in the hardware or the software of a specific sensing module require significant redesign of the whole system, thereby increasing the cost and the development time. We firmly believe that the described system addresses some of these sensor-based control issues and provides a unified way of looking at problems of this type.

3 Issues

Many visual servoing systems use feature-based approaches or image attributes derived from image features such as optical flow. These systems typically find correspondences between features present in two images acquired at different instances in time from a camera mounted on the end-effector. Such systems may have difficultly handling cases in which object features become occluded or object motion or deformation alters the feature beyond recognition. For example, systems which define a feature as a template of pixels can fail when a feature rotates relative to the template used to match it.

To overcome these difficulties, the system proposed in this chapter incorporates contour tracking techniques. When a contour corresponding to the object boundary can be extracted from the image, it provides information about the object location in the environment. If prior information about the set of objects which may appear in the environment is available to the system, the contour might be used to recognize the object or to determine its distance from the camera. In other words, if a contour can be extracted from the image, and this contour corresponds to an object boundary, the contour

provides information useful to a visual servoing system. If additional, prior information about object shape and size can be combined with the contour information, the system could be extended to respond to object rotations and changes in depth.

For contour extraction, we have adopted the active deformable model methodology. Active deformable model techniques attempt to identify image contours by minimizing a contour energy function which includes terms representing regularization constraints such as contour smoothness and continuity as well as terms dependent on image attributes such as local contrast. A number of approaches have been proposed for formulating and finding a minimum for these functions, since the introduction of active deformable models to computer vision by Kass *et al.* [16]. The algorithm presented by William and Shah [29] and elaborated by Yoshimi and Allen [28] is particularly well-suited for our application as it is both iterative and greedy. Because it is iterative, partial solutions are available during the minimization process; because it is greedy, the quality of these partial solutions tends to increase. We take advantage of these properties by running the control system and the contour extraction algorithm simultaneously. The controller issues commands to the arm based on the most recent partial solution. If the image were static, the system would tend to a steady-state in which the object is centered in the image and the minimizing algorithm reaches a stable solution. However, since the object we are interested in is moving, the movements of the arm tend to change the image in ways that increase the energy of the current model configuration (since areas of high contrast in the image have moved), which forces the minimization algorithm to find a new configuration of minimum energy.

On the other hand, as long as object translations and deformations between frames are reasonably small, a minimum configuration of the active deformable model in one frame will be close to a minimum configuration for the model in the subsequent frame. The algorithm may find the minimum in a few iterations. In the best case, the minimization algorithm will be fast enough and the inter-frame displacements small enough that a minimum configuration can be found for each frame. This theoretical ideal may be impossible to achieve in practice—it would require a perfect model, signal processing system and control law, but it illustrates the advantage of combining models and servoing. In short, tracking with active deformable models makes servoing possible and, in turn, the act of servoing simplifies the task of deforming the model to fit the contours of the image in the plane.

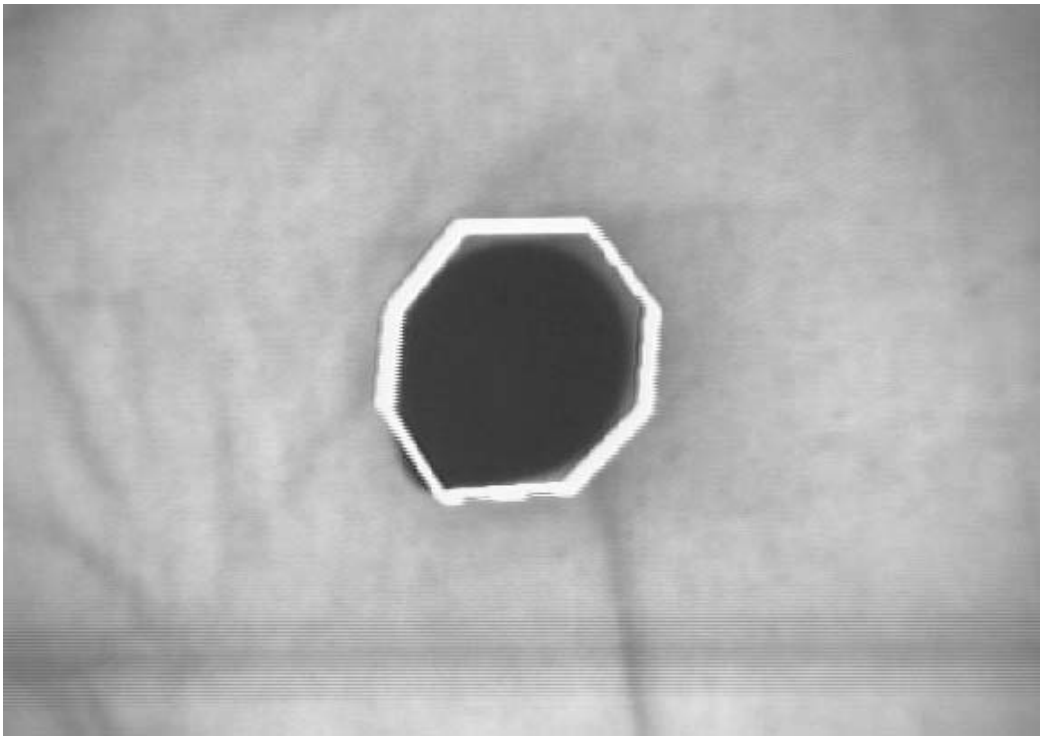

Figure 1: An active deformable contour tracking a balloon.

The center of the model—defined as the average location of the control points or as the two-dimensional center of mass—is used as the input to the manipulator's controller. Optimal estimation and control techniques (an LQG regulator) are used to deal with noise in this signal. We have conducted experiments which indicate the feasibility of this approach in dynamic, but controlled environments, such as an automated factory floor.

When the speed of the minimization algorithm relative to the speed of the contour displacements and deformations in the image plane is sufficient the system presented in this chapter tracks reliably. We have begun experimental work (see Section 7) in an effort to define and quantify the relevant factors, (*e.g.* image displacement, image deformation, speed of the minimization algorithm, gains for energy function, number of control points, *etc.*). When the key factors have been identified, future work will focus on improving the critical elements of the system.

4 Previous Work

This work draws on two streams of research in the computer vision and robotics communities. We have combined techniques developed by the visual servoing community with contour extraction techniques developed in the graphics and computer vision communities.

Several research efforts have focused on using vision information in the dynamic feedback loop [7, 10, 13, 14, 19, 24, 27]. Weiss *et al.* [24] have proposed a model reference adaptive control scheme for robotic visual servoing. In this work, servoing is performed with the goal of reducing the error between the desired image attributes (center-of-mass, first or second moment of the image) and the current image attributes. The verification of the proposed algorithms has been limited to simulation. Allen [1] has proposed an approach that uses image-differencing techniques in order to track and grab a moving object. Dickmanns [11, 12] has presented methods (Kalman filters) for the integration of vision information in the feedback loop of various mechanical systems such as satellites and automobiles. Koivo and Houshangi [17] have proposed an adaptive scheme for visually servoing a manipulator based on the information obtained by a static sensor. Feddema and Lee [13] have proposed a MIMO adaptive controller for hand-eye visual tracking. Their work has been used as the basis for our approach. Several other researchers [3, 18] have proposed strategies for vision-based exploration. Finally, Ghosh [14] has addressed several vision-based robotic issues with the aid of a new "Realization Theory" for perspective systems.

The concept of active deformable models, also called "snakes," was first introduced to the field of computer vision by Kass *et al.* [16]. Snakes have been used in a number of applications including image-based tracking of rigid and non-rigid objects. Using snakes requires a minimization process of an energy function. Several techniques have been used to solve this problem including variational calculus [16], dynamic programming [2], and greedy methods using heuristics [28, 29]. The latter method has the advantage of being fast as well as numerically stable. Our method uses a greedy method similar to that used by Williams and Shah [29] and Yoshimi and Allen [28].

Other researchers have also combined elements of visual servoing and active deformable models techniques to approach different problems than the one presented in this chapter. Blake, Curwen, and Zisserman [4] have presented a different algorithm for contour estimation and used it in a system which tracks a contour in an image (it does not include a robotic component).

Yoshimi and Allen [28] have used a greedy, iterative minimization algorithm to track a robotic finger with a static camera and detect contact between the finger and a stationary object. Finally, Colombo *et al.* [9] published a description of a system which uses a spline contour model to plan and execute a movement that positions an eye-in-hand robot so as to bring a known object into a canonical orientation relative to the camera. They report initial simulation results.

5 Proposed Approach

We describe a system that tracks a moving, deformable object in the workspace of robotic arm with an eye-in-hand camera. For these experiments, we have used a figure-ground approach to object detection and identification. The figure-ground methodology allows pixels to be identified as object or background pixels, a distinction which is useful during the initial placement of the active deformable model. However, this limits the applicability of our system to environments in which the background is uniform.

Once the active deformable model has been placed, two simultaneous processes commence. One process uses an iterative, greedy algorithm to find a minimum energy configuration of the active deformable model. The second process issues control commands to the manipulator based on the current configuration of the active deformable model. Movements of the manipulator alter the position of the camera and, consequently, the image forces used by the minimization algorithm, closing the control loop.

5.1 Placing the Model

Movement in a scene can be detected by comparing two images acquired by a camera: a ground image taken before the movement occurred and the current image. This difference image is defined as (where x and y are image coordinates):

$$I_{diff}(x,y) = |I_{ground}(x,y) - I_{curr}(x,y)|. \tag{1}$$

To enhance the boundary contours of the object's image in the difference image, we increase the contrast of the difference image with a simple thresholding operation, where:

$$I'_{diff}(x,y) = \begin{cases} 0 & \text{if } I_{diff}(x,y) < T \\ 255 & \text{otherwise} \end{cases} \tag{2}$$

for a threshold T. In this work, when we use the term difference image to refer to a specific image, we mean the binary image I'_{diff} that is obtained from these operations.

When the system begins operation, a background image of the scene is captured to be used as the ground image, I_{ground}, for the calculation of image differences. The object to be tracked is introduced to the scene and the tracking system is alerted by a signal from the operator. Either manually or automatically, a bounding-box is placed around the region of differences created by the object.

After a bounding box has been selected, an initial configuration for the active deformable model is chosen by one of several algorithms. In the course of this research effort, we experimented initially with three significantly different techniques.

The initial placement algorithm simply placed control points along the bounding box by splitting each edge of the bounding box into a number of model edges. In other words, the four corners of the bounding box became control points in the initial configuration. If desired, one or more equidistant points were also chosen on each edge. When sufficiently large values were chosen for the balloon constraint (described in Section 5.3), this crude placement method was fairly successful. The model quickly collapsed upon the image contour. However, this method is ill-suited for experiments using variations of active deformable models which incorporate task-specific constraints (as opposed to the generic "snake" constraints of equidistance and equal angles). It does not provide useful default values for the constraints. Indeed the initial configuration of the active deformable model has little relation to the desired model configuration.

Current work has focussed on the use of a more specific, but still fairly generic, determination of initial model configuration. First, a blob is chosen as the primary object of interest. Then, the boundary of the connected blob is extracted by an edge-following algorithm (similar to the Boundary-Following Algorithm described in [15]). Then the boundary a predetermined number of control points are placed along the boundary. A configuration determined by this method is generally well-suited for tracking using generic constraints, and is a reasonable configuration more specific constraints—at least for one orientation of the model.

There are at least three serious disadvantages to this approach:

- The connected blob chosen for boundary following may not correspond

to the image contour that should be tracked. There may be blobs in the difference image which are unrelated to the object of interest. These blobs may be caused by noise or the presence of other objects. Alternatively, the object of interest may create more than one blob in the difference image because it does not create a uniform difference with the background. Both of these types of errors should be alleviated by the application of simple image processing techniques, such as blob size thresholding combined with morphological operators.

- Points which are equally spaced on the perimeter are not necessarily equidistant. For example, points on a serrated boundary will be much closer in image coordinates than in perimeter coordinates. An optimization algorithm could overcome this limitation in the general case, but at high computational cost. There are probably heuristic approaches which would find good configurations in the majority of cases, but we have not identified any.

- Points chosen by a perimeter walk may not be points of deformation on the object contour or points of high curvature on the contour. Ideally, the control points would be placed at points where deformation will occur, or at an object corner, where changing relative viewpoint changes the angle projected on the image plane. Since the perimeter walking scheme does not consider object characteristics or the curvature of the extracted boundary, it does not reflect these characteristics of the contour.

In order to address the last two of these weaknesses, we have undertaken preliminary investigation of an automatic placement technique discussed in the next section.

All three of the methods considered are only suited for use in conditions when the contour model must be determined from a single example provided at run-time. For situations in which the contour model is known *a priori* it would be straightforward to apply a Generalized Hough Transform to edge-detected image to determine the position, orientation and any uncontrolled parameters of the model.

Once the active deformable model has been placed by any of the methods described here, its movements are controlled by the minimization of an energy function as described in Section 5.3.

5.2 Automatic Selection of Control Points Using the P & P Algorithm

We have developed an algorithm (P & P algorithm in [22]) which locates points of high curvature (corners) using a method similar to that in [6]. It also locates key in-between low curvature points (key flat points) by employing a procedure conjugate to that for locating corners. We have tried the particular algorithm for the selection of control points.

5.2.1 Selection of Corner Points

The determination of corners is done in a way very similar to the method followed in Brault's algorithm [6]. The notable difference is that there is no need for parameter tuning. The basic mechanism is the same with that of Brault's algorithm [6]. Each point c of the curve is seen as a potential corner. The neighboring points from either side of point c contribute to the cornerness of c in a degree determined by certain conditions.

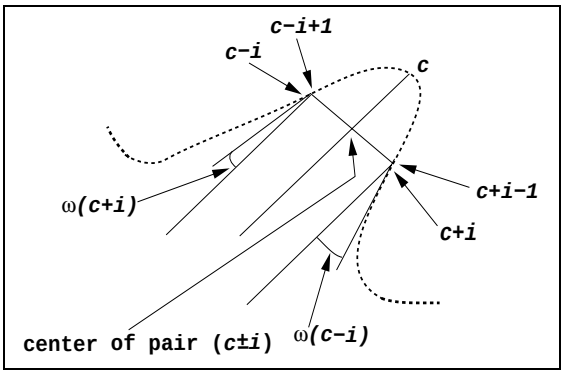

Figure 2: Geometric model for corner determination as proposed by Brault [6].

In more detail, the angles $\omega(c+i)$ and $\omega(c-i)$ (see Figure 2) are computed for each pair of neighbors $c \pm i$ $(i = 1, 2...)$. In order for a pair $c \pm i$ to belong to the corner domain of point c the following inequalities must be satisfied:

$$\omega(c+i) < \frac{\pi}{2} \quad and \quad \omega(c-i) < \frac{\pi}{2}. \tag{3}$$

The contribution CF (Cornerness Factor) of each pair $c \pm i$ to the making

of the candidate corner c is computed by the formula

$$CF(c,i) = \cos(\omega(c+i)) * \cos(\omega(c-i)). \tag{4}$$

Using $\frac{\pi}{2}$ as a fixed upper limit in the inequalities, Formula (3) is a departure from the method followed in Brault [6] and is what renders the corner determination parameterless. The first $M(c)$ points that satisfy the inequalities (3) constitute the corner domain of point c and their total contribution to the cornerness of point c is computed by

$$TCF(c) = \sum_{i=1}^{M(c)} CF(c,i). \tag{5}$$

The corner segmentation points are identified by searching the values of the function $TCF(c)$. The TCF values of the curve points present a very consistent pattern: strings of nonzero values spaced by strings of zero values. Each of the nonzero strings corresponds to a high curvature segment, and the maximum value contained in each such string corresponds to a corner segmentation point.

5.2.2 Selection of Key Low Curvature Points

While corners are the perceptually most important parts in a curve, corners alone provide insufficient data for an accurate reconstruction of a curve. The situation improves substantially if we provide some key points with rather low curvature surroundings, that lie between corners, as extra segmentation points. The way we find these key low curvature points is conjugate to the way we find the corner points.

More precisely, a separate processing step is taking place for the location of the *key low curvature points*. The geometric parameters shown in Figure 3 are the same with these in Figure 2 and are computed for each pair of neighbors $f \pm i$ $(i = 1, 2...)$ of every point f of the curve. This time, however, the larger the angles $\omega(f+i)$, and $\omega(f-i)$ are than $\frac{\pi}{2}$, the more the corresponding pair of neighboring points contributes to the *low curvatureness* of point f. As a result, by a suitable analysis of the angles $\omega(f+i)$ and $\omega(f-i)$, one can determine whether or not the pair of points $f \pm i$ is a part of the low curvature domain of f and, in addition, can estimate the importance of the contribution of these points to the low curvatureness of point f.

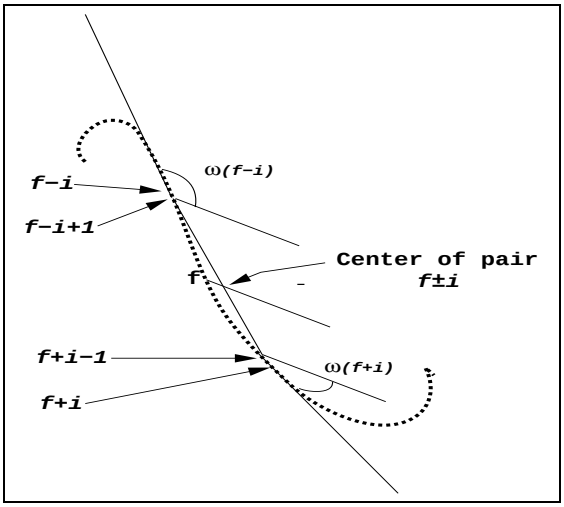

Figure 3: Geometric model for key low curvature point determination.

The angles $\omega(f+i)$ and $\omega(f-i)$ must satisfy the following inequalities:

$$\omega(f+i) > \frac{\pi}{2} \quad or \quad \omega(f-i) > \frac{\pi}{2}. \tag{6}$$

The contribution FF (Low Curvatureness Factor) of each pair $f \pm i$ to the making of the candidate key low curvature point i is computed by the formula

$$FF(f,i) = \mid \cos(\omega(f,i)) \mid * \mid \cos(\omega(f,i)) \mid . \tag{7}$$

In contrast to Eq. (5), Eq. (7) uses the absolute value of the trigonometric function *cos* since the range of the angles $\omega(f+i)$ and/or $\omega(f-i)$ features now $\frac{\pi}{2}$ as a lower and not as an upper limit. The total contribution of the first $M(f)$ points belonging to the low curvature domain of f (the ones that satisfy the inequalities (6)) is computed by

$$TFF(f) = \sum_{i=1}^{M(f)} FF(f,i). \tag{8}$$

The identification of the key low curvature segmentation points from the function $TFF(f)$ is done in a way analogous to the determination of corner points from the function $TCF(c)$.

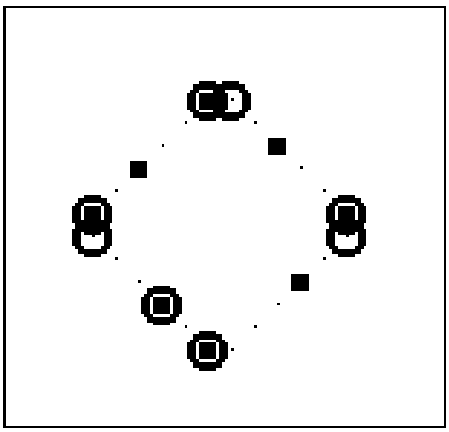

Figure 4: A square contour.

5.2.3 Evaluation of the Proposed Algorithm

In order to get an indication of the goodness of the algorithmic selection of control points in terms of the accuracy of shape description, the following experiment was devised. Let a contour $\mathcal{C}$ of an arbitrary shape consist of N points ($\mathcal{C} = (\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_N)$). Let the P & P algorithm select for the contour $\mathcal{C}$ a set $\mathcal{S}$ of m control points ($\mathcal{S} = (\mathbf{P}_{s1}, \mathbf{P}_{s2}, \ldots, \mathbf{P}_{sm})$). Let also a set $\mathcal{T}$ of m control points ($\mathcal{T} = (\mathbf{P}_{t1}, \mathbf{P}_{t2}, \ldots, \mathbf{P}_{tm})$) to be chosen in a way so that an error norm is driven to minimum (optimal polygonal fit). The norm chosen for the purposes of the particular experiment was the Euclidean distance error of the polygonal fit represented by the point set. The set $\mathcal{T}$ was determined after an exhaustive search of all the $\binom{N}{m}$ combinations for the contour $\mathcal{C}$. It is interesting to compare the set of control points given by the P & P algorithm with the optimal polygonal fit point set for a variety of shapes (see Figures 4 through 7).

The small circles in the above figures represent the points of the optimal polygonal fit set while the points given by the P & P algorithm are represented by small squares. In all the shapes, the prominent corners are included in both the optimal polygonal fit set and the set of the P & P algorithm. Discrepancies arise only for the key flat points of the algorithm. The equivalent points of the optimal polygonal fit are mostly clustered in noisy areas of the shape. In contrast, the key flat points of the algorithm are uniformly distributed between the prominent corner points. This behavior is highly desirable, since the algorithm has not been designed specifically for a polyg-

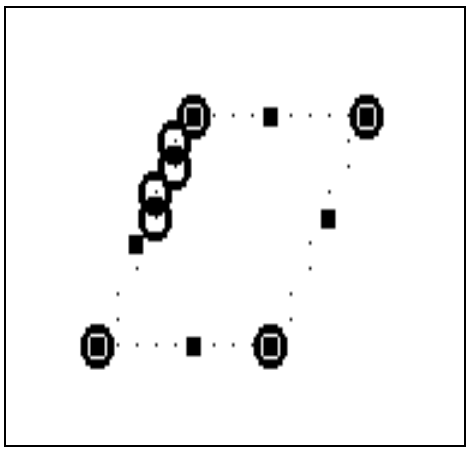

Figure 5: A parallelogram contour.

onal fit but for a more generic fit that may be even a spline fit. In fact, some model-based techniques use the control points for polygonal fits [26, 25, 28] and some others for spline fits [4]. The algorithm loses very little in terms of polygonal fit accuracy by placing the key flat points in a distributed instead of a clustered manner. For example, in the irregular contour case of Figure 7, the error of the optimal fit is 0.8189 pixels while the error of the P & P fit is 2.1701 pixels. The error of an arbitrary polygonal fit for this shape could run as high as 42.8378 pixels. The small compromise the algorithm concedes in the polygonal fit case pays off in the spline fit case where a clustered distribution like the one favored by the optimal polygonal fit would give very poor results.

5.3 The Active Deformable Model

The formulation of active deformable models used in this work to approximate the object boundary draws on the work done in recent years by the computer vision community on active deformable models of contours, often referred to as "snakes." Given a continuous contour, described as a vector:

$$\mathbf{v}(s) = (x(s), y(s)) \tag{9}$$

where s is the arc length, Kass *et. al.* [16] related the task of finding a contour in an image to the minimization of an energy function (adopting the

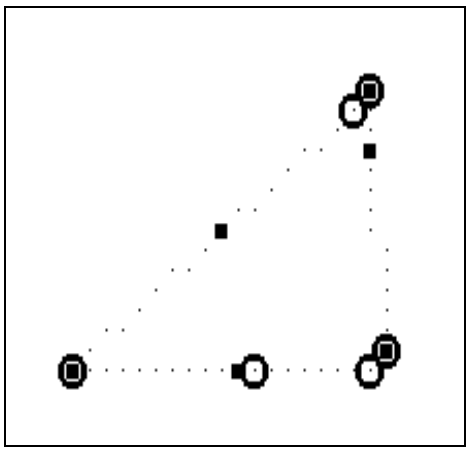

Figure 6: A triangular contour.

notation used in [29]):

$$\begin{aligned} E^*_{snake} &= \int_0^1 E_{snake}(\mathbf{v}(s))ds \\ &= \int_0^1 [E_{int}(\mathbf{v}(s)) + \\ &\quad E_{image}(\mathbf{v}(s)) + \\ &\quad E_{con}(\mathbf{v}(s))]ds. \end{aligned} \tag{10}$$

In this function, E^*_{snake} is the total energy of the active deformable model, E_{int} is a measure of internal energy, such as that caused by curvature, and E_{image} is a function of image characteristics. The term E_{con} is derived from external constraints. When this continuous model is approximated in a discrete domain (*e.g.*, a digital image) the equation becomes:

$$\begin{aligned} E^*_{snake} = \sum_{j=1}^{n} & [\alpha E_{cont}(v_j) + \\ & \beta E_{curv}(v_j) + \\ & \gamma E_{image}(v_j)] \end{aligned} \tag{11}$$

in which E_{cont} is derived from the distance between v_j and its neighbors, $v_{(j-1) \bmod n}$ and $v_{(j+1) \bmod n}$. E_{curv} is a function of the angle at point v_j. Again, E_{image} represents the image forces acting on the active deformable model. The terms α, β, and γ are weighting parameters which control the proportion

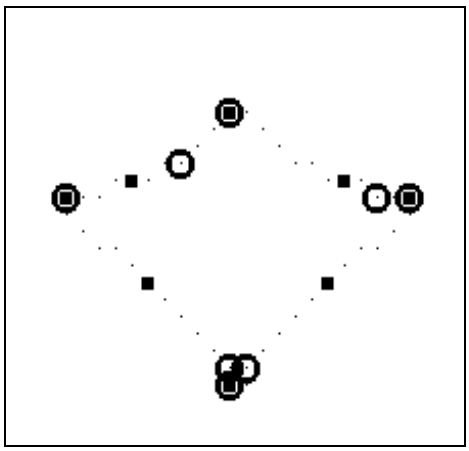

Figure 7: An irregular contour.

of the active deformable model's energy derived from each of the three terms, which are assumed to be normalized.

Kass *et. al.* [16] proposed that a minimum be found for this energy function with a variational calculus approach. Amini *et. al.* [2] have proposed a method based on dynamic programming. We have chosen to adopt the greedy method developed by Williams and Shah [29]. In the greedy method, each point on the contour is considered in turn. An energy score is calculated for locations near the current location of the control point and the control point is moved to the location which results in the lowest energy.

The E_{curv}, E_{image}, and E_{cont} terms are usually sufficient to define an active deformable model approximation of an image contour when all terms vary significantly across the neighborhood of possible control point locations. However, using our current techniques, when the active deformable model is placed, it may have several control points which are far enough from the target's image that the image gradient is unvaryingly zero throughout the neighborhood of candidate locations. For these points, the term E_{image} plays no role at all and they only respond to the internal energy and external constraints, rather than to a combination of image energy and constraints.

To facilitate the initial placement of the active deformable model, we have augmented the energy equation with an E_{model} term inspired by the "balloon factor" used by Yoshimi and Allen [28] to overcome a tendency toward implosion in their active deformable models.

The E_{model} term is calculated as follows. First, a neighborhood of the

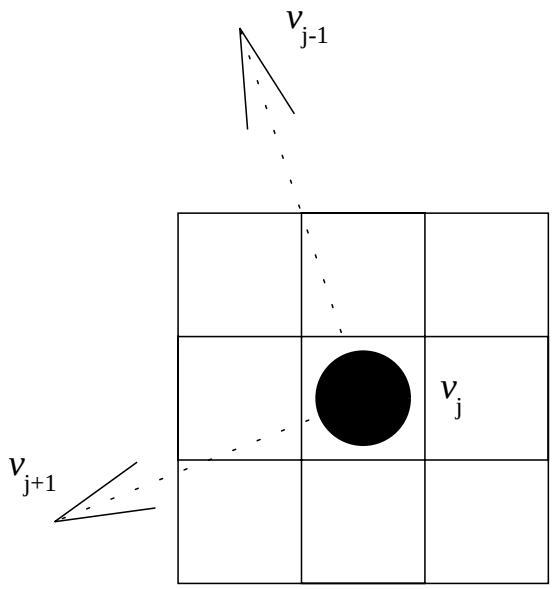

Figure 8: A single snake point in its window.

control point in the difference image is examined. If the percentage of difference pixels set within the neighborhood falls short of a predetermined level, the control point is defined as "outside" the object's image. To bias movement of the control point toward the object's image, the locations closest to the object's image are assigned the value -1 for E_{model}. Other locations are assigned the value 0. The locations closest to the object's image can be determined because the active deformable model control points are numbered counter-clockwise around the closed active deformable model. A similar energy assignment is performed for control points which are "inside" the object's image. Besides aiding initial placement of the contour, this model energy also occasionally comes into play during later tracking stages when an object moves very quickly or has been temporarily lost for some other reason (*e.g.*, occlusion).

5.4 The Control Signal Computation

Concurrently with the energy minimization process described above, a control signal is generated from the current configuration of the active deformable model by a process running on a separate processor. The purpose

of this process is to determine the necessary camera translation to recenter the contour extracted by the active deformable model in the image plane.

It is necessary to choose a definition for the location of a contour. We have considered two options: 1) the average location of the control points and 2) the centroid of the closed polygon defined by the contour. We have chosen to use the average location of the control points. This definition may be unsatisfying if the control points become bunched together on one side of the contour, but, in practice, this rarely occurs as the smoothness and continuity constraints described above penalize such configurations. Therefore, the slight improvement in these cases does not justify the additional processing time.

There is much more information in the configuration of the active deformable model than location. Future systems should be able to take use this information for 3-D tracking and to overcome partial occlusions.

6 The Minnesota Robotic Visual Tracker

The Minnesota Robotic Visual Tracker (MRVT) [5] that was used for these experiments consists of the Robot/Control Subsystem (RCS) and the Visual Processing System (VPS).

The RCS includes a PUMA 560 robotic arm, its Unimate computer/controller, and a VME-based Single Board Computer (SBC). The manipulator's trajectory is controlled by the Unimate controller as directed by path updates provided by an Ironics 68030 VME SBC running CHIMERA. A Sun SparcStation 330 hosts CHIMERA and shares its VME bus with the SBC via BIT-3 bus extenders. BIT-3 bus extenders also provide shared-memory communication between the RCS and VPS.

The VPS receives input from a video source such as a camera mounted on the end-effector of a robot arm, a static camera, or stored imagery played back through a Silicon Graphics Indigo or a video tape recorder (see Figure 9). The output of the VPS may be displayed in a readable format or can be transferred to another system component and used as an input into a control subsystem. This flexibility offers a diversity of methods by which software can be developed and tested on our system. The main component of the VPS is a Datacube MaxTower system consisting of a Motorola MVME-147 single board computer running OS-9, a Datacube MaxVideo20 video processor, and a Datacube Max860 vector processor in a portable 7-slot VME chassis.

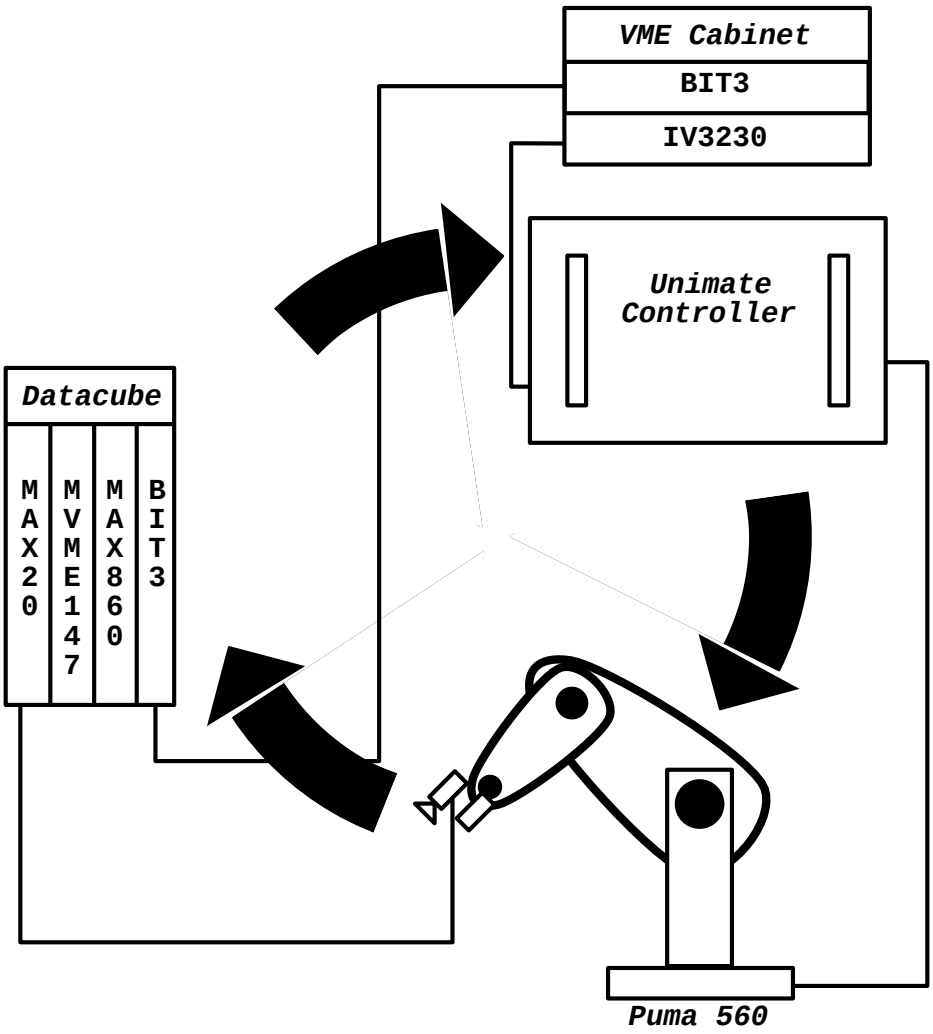

Figure 9: MRVT system architecture.

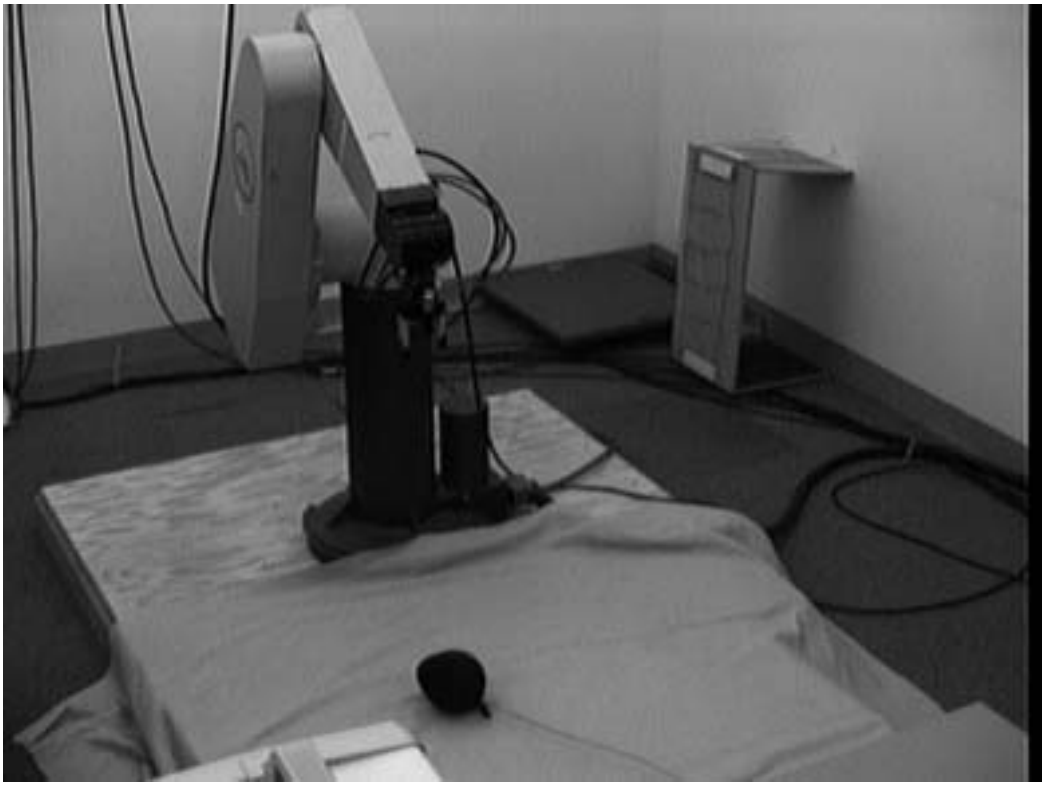

Figure 10: Experimental setup for balloon tracking.

The VPS performs the calculation of the difference image and the active deformable model energy minimization and calculates any desired control input. It can supply the data or the input via shared memory to an off-board processor via a Bit-3 bus extender for use as input to the RCS. The video processing and calculations required to produce the desired control input are performed under a pipeline programming model using Datacube's Imageflow libraries.

7 Experiments

Initially, two types of experiments were run. In the first, a partially-inflated balloon was moved by hand in the robot's workspace. These runs were analyzed for timing information as well as qualitative information about system performance. Quantitative measures of tracking quality are not available

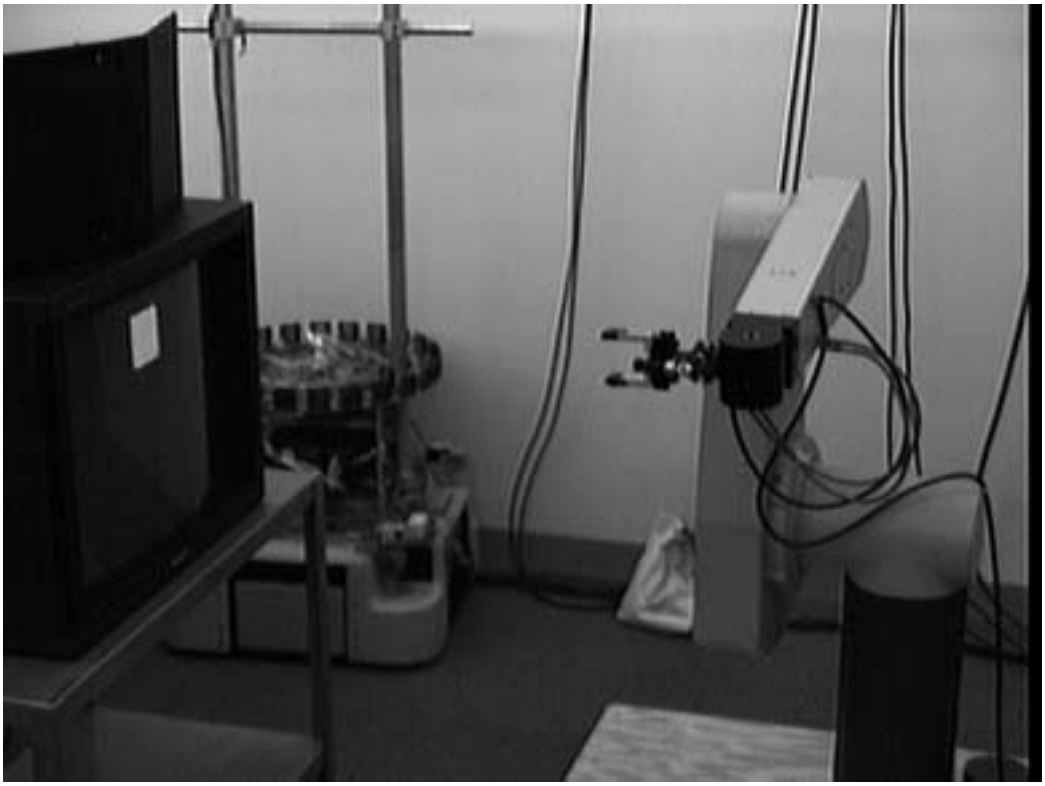

Figure 11: Experimental setup for quantitative trials.

from these runs, as the nature of the experiments denies access to "ground truth." To obtain quantitative data about system performance, a second set of experiments were conducted. In these trials, a SGI Indigo workstation was used to create a display of an object in motion along a circular path. While the MRVT tracked the object on the display, the control commands issued to the controller were collected. By comparing the control commands to the actual path of the object, tracking performance can be quantified.

In the balloon-tracking experiments, a black balloon attached to a stick was maneuvered in the manipulator's workspace by an operator. The workspace background was grey and fairly uniform, creating few distracting difference pixels (*i.e.*, non-object pixels which appear in the difference image). Empirically discovering gains which overcame this noise and resulted in good tracking performance was not difficult. The minimization algorithm performed approximately 2000 point updates per second (*e.g.*, over eight trials,

totaling 13 minutes and 9 seconds, eight-point snakes performed 251 updates per second). This update rate was seemed adequate for snakes with as many as 16 control points.

Informal testing did reveal one difficulty with the current implementation. The E_{model} term, which aids initial placement, interferes with tracking when the active deformable model is not a simple polygon. Various techniques to guarantee simplicity have been implemented and tested, but none has been effective without unacceptable performance penalties.

In the second set of experiments, a target was generated on an SGI Indigo and presented on a 27 inch monitor just outside the robot's workspace (see Figure 11). This target, a 7.3 cm square, repeatedly traveled in a circular path with a diameter of 25.7 cm or along a square path with sides of 27 cm. While traveling at about 8 cm/sec, deformation was introduced by rotating the square 360 degrees on its z-axis during each circuit. The position commands sent to the Unimate controller were collected. The first 1200 points from two sample runs are plotted in Figure 12. The left-hand and right-hand plots contain data from a four-point and eight-point model, respectively.

These plots demonstrate the trade-off between additional control points and system performance. In the four-point trials, the minimization algorithm performed 505 updates per second and the control loop sent 212 path instructions per second to the arm. In the eight-point trials, the minimization algorithm performed half as many updates per second (250) and only 146 control instructions were sent per second. Apparently, two iterations are not enough for the minimization algorithm to converge. Although the eight-point snake was able to track the target, the plots reveal many more oscillations in the path and a lack of consistency. One goal of future work should be to improve the performance of the minimization algorithm, so that better tracking can be obtained with more complex models.

For comparison, Figure 14 plots the path of a manipulator following the same target along a square path at similar speed, demonstrating how the controller handles discontinuities in the target path, and acceleration and deceleration of the manipulator.

We also tried the P & P algorithm for the automatic selection of control points. Preliminary results of experiments incorporating the P & P algorithm for automatic control point selection in a model-based tracking scheme [26] suggest that this approach holds great promise. The P & P algorithm extends the previous version of our system in two important ways. It automates the selection of both the number and location of control points. its operator.

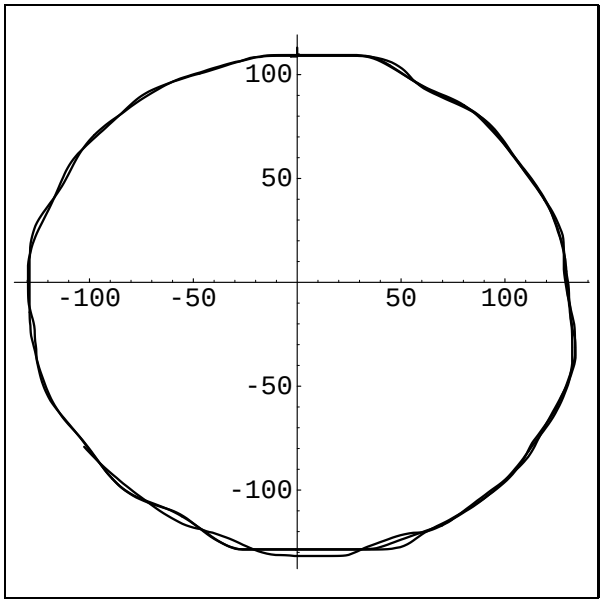

Figure 12: Tracking a rotating square target with a four-point model (measurements in mm).

Experiments were conducted in which a target was presented on a 27 inch monitor located one meter from the end-effector mounted camera. The target, a 7.3 cm tall square or triangle, moved around a rectangular path of 100 cm at approximately 8 cm/sec. The position commands sent to the robotic arm were collected and are graphically illustrated in Figures 15 - 17. Previous results [26] (see Figure 16) were compared to results using the P & P Algorithm (see Figure 17).

The previous system used a *predetermined* number of control points irrespective of the target's shape. These points were *manually* placed near the object contour in a highly regular configuration. The generic constraints used by the tracking algorithm created a bias toward equidistant points and equal angles between edges. The new system uses the P & P algorithm to automatically select control points. Because the P & P algorithm does not choose equally spaced points, the constraints used during tracking were modified to reward configurations with angles close to the initial angles and distances close to the initial distances.

The model-based tracking scheme with the manual selection of control

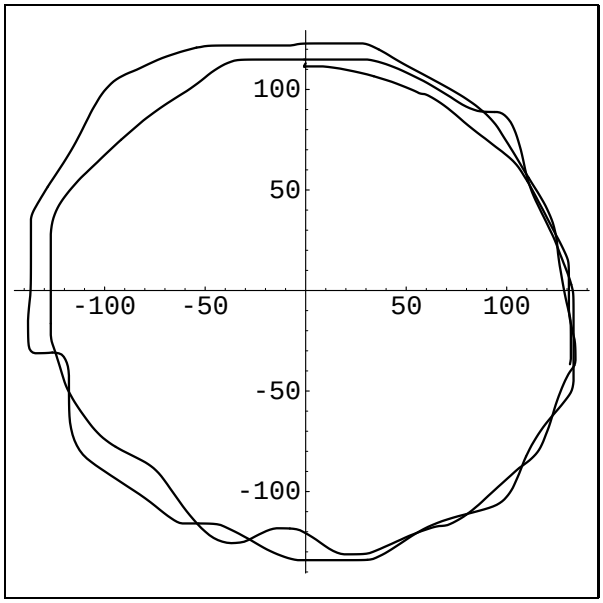

Figure 13: Tracking a rotating square target with an eight-point model (measurements in mm).

points worked well only when a small number of control points was selected and the points described the contour well. Since that system encouraged equidistance between control points and equal angles between edges, it performed best when the contour of the object being tracked could be approximated by an equilateral polygon (a highly regular shape) with as many vertices as the model had control points. For less regular shapes or control point configurations, performance degraded. For example, the system in [26] lost track of the square target after just one revolution when an eight-point model was used (see Figure 16). The old system was not tested with the (non-equilateral) triangular target, since this target is not a highly regular shape.

The system using the P & P algorithm for automatic point selection performed substantially better. Ten trials were measured. In the first five, the arm tracked the moving square. In the second five, the triangular target was tracked. Results from the first trial with each target are presented in Figures 15 and 17 respectively. The control point selection algorithm invariably selected ten points for the square and six points for the triangle that appropriately described the shapes. The tracker maintained tracking of the objects

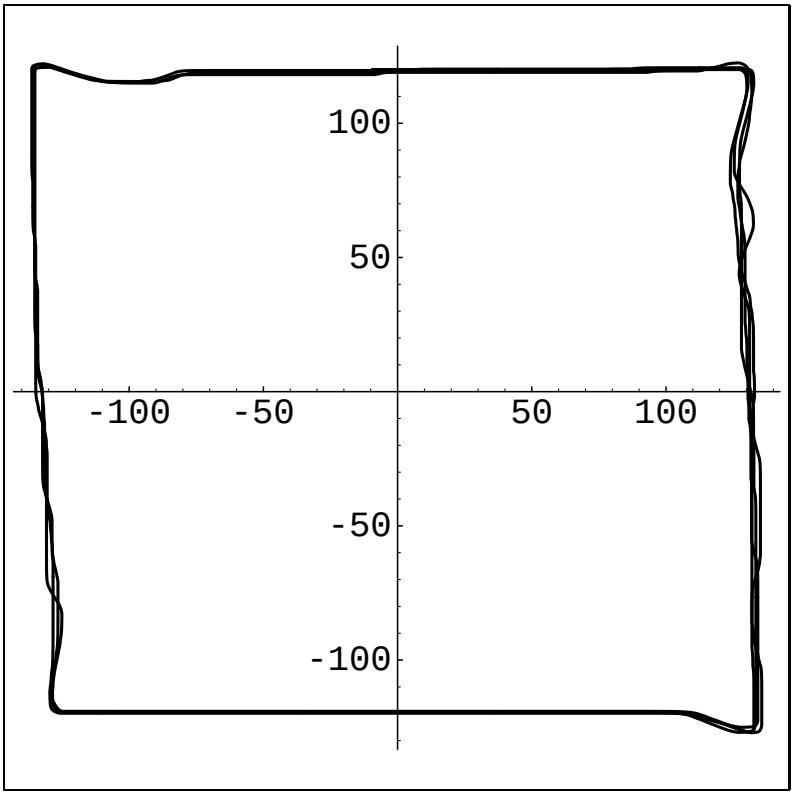

Figure 14: Tracking a deforming square target with a four-point model (measurements in mm).

for several revolutions. In this experiment, the P & P tracker exhibited its ability to maintain tracking at fairly high speeds of different target shapes (square, triangle).

In order to show the generality of the approach, we used the method in another domain (pedestrian tracking). With exactly the same formulation like in the case of visual servoing, our system can successfully track motion of a walking pedestrian, even when the pedestrian's image deforms in unexpected ways such as those caused by thrusting out one's arms or kicking a leg forward in an exaggerated manner (Figures 18 and 19). It is also fairly robust with respect to occlusions such as when two pedestrians pass in opposite directions or a single pedestrian passes behind a large tree. Potentially, more than one pedestrian could be tracked simultaneously. Although such a system should be equally robust with respect to occlusions caused by two

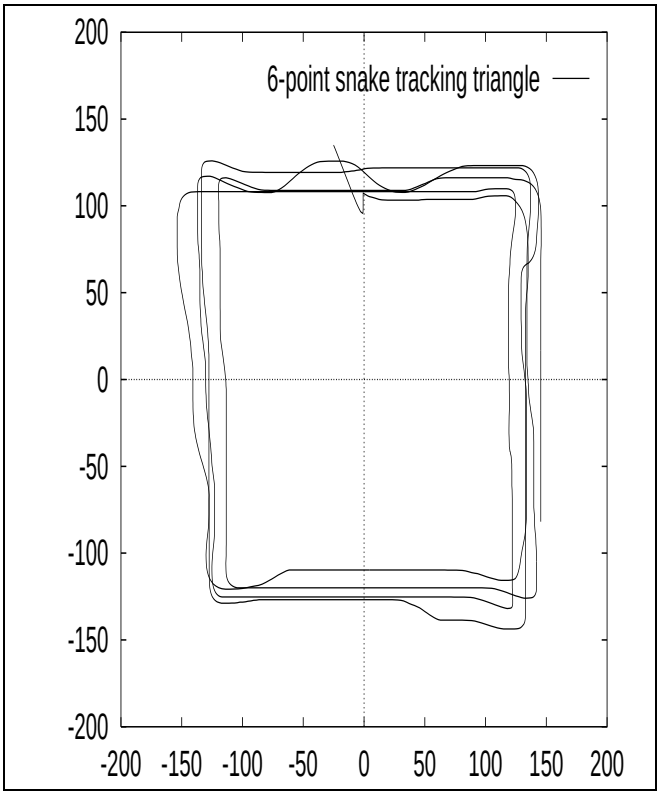

Figure 15: Tracking of a triangular target with the P & P algorithm (measurements in mm).

tracked pedestrians passing one another, it would probably not be possible to tell whether the active deformable models had continued to track the same individual. Such a system might have difficulty distinguishing between two pedestrians approaching one another and then returning the way they came and two pedestrians walking past one another.

Further development of the transportation-related system will require overcoming the inherent limitations of using a difference image to provide image forces for the active deformable model. These problems include short and long time-scale changes in the background caused by lighting changes or continuous regular movement of objects in the scene, for example, the rustling of leaves in the wind. The system is also vulnerable to the effects of camera self-motion. A slight jitter in the camera mount could cause many patches of noise in the difference image. Although these patches will generally be

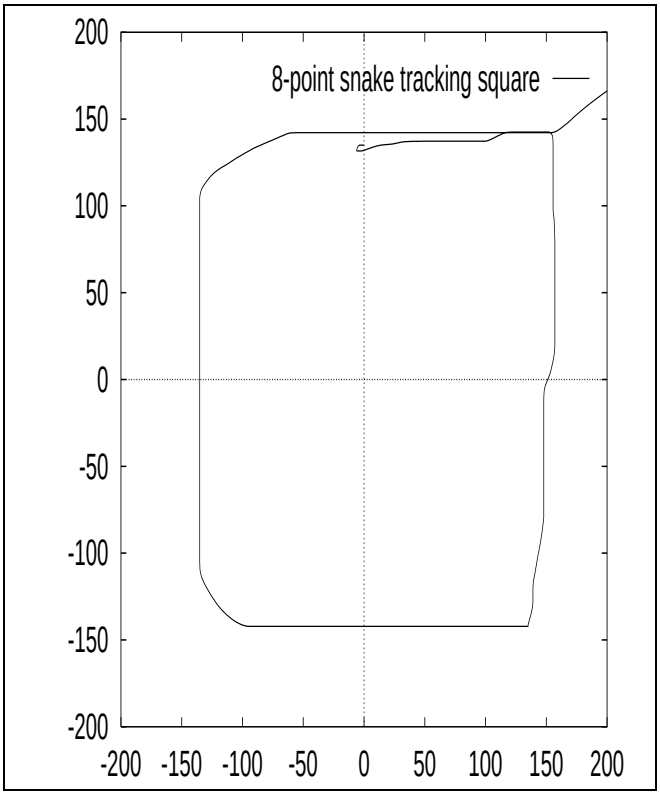

Figure 16: Tracking of a square target without the P & P algorithm. The target was lost after one revolution (measurements in mm).

ignored once contour tracking has begun, they do disturb the initial placement of the snake. Richards *et. al.* [23] describe two enhancements to the difference image framework to overcome these difficulties. First, by slowly modifying the ground image in a controlled way, changes in the background can be incorporated in the ground image. Second, to overcome the placement problem, additional processing of image regions can be done to identify portions of the image consistent with the appearance of a pedestrian. We plan to incorporate these improvements in our system. Consideration should also be given to methods which would make it possible to mount the camera in a moving vehicle.

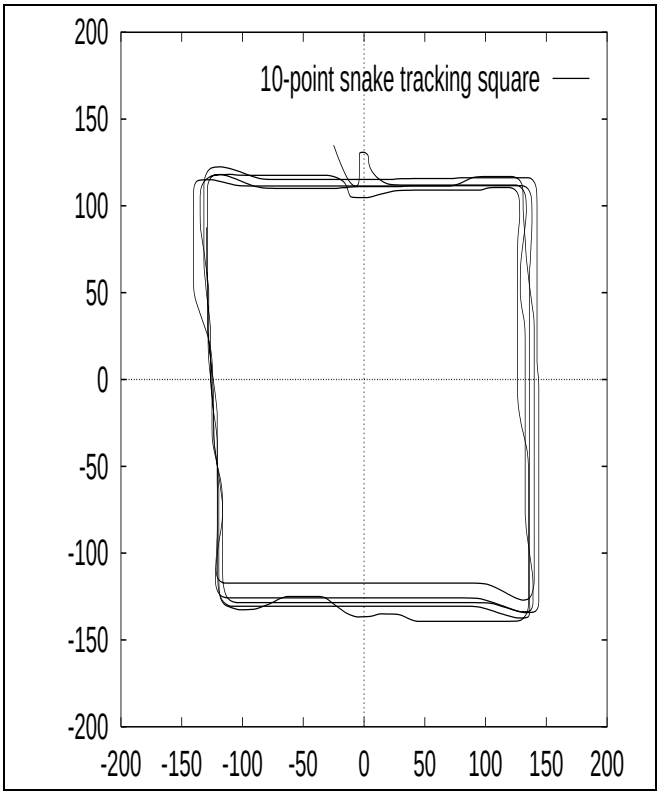

Figure 17: Tracking of a square target with the P & P algorithm (measurements in mm).

8 Discussion

Although the results of the experiments described in Section 7 demonstrate the promise of a system combining the active deformable models for visual tracking with a visual servoing system, they also illustrate drawbacks of the current implementation.

Two factors affect the quality of tracking which must be discriminated. The initial set of experiments conflates changes produced by the sheer number of control points with effects caused by the match between the number of control points and with the points of high curvature on the object boundary.

For example, performance degraded significantly when an eight-point model was used to track a four-sided figure. However, there are two reasonable explanations for this difference. 1) The extra computation required to mini-

Figure 18: A six-point active deformable model tracking a pedestrian.

mize an eight-point model reduced total model update time by a factor large enough to create a qualitative drop-off in overall performance. Or, 2) the match between object shape and model was not good enough to achieve a stable minimum.

It should be noted that an important strength of the minimization algorithm (its local character) is also a weakness in this case. In no sense does the algorithm trade-off higher curvature in one region to achieve lower curvature in another. It relentlessly attempts to reduce curvature (or approach a default angle) at every control point. Further, because the minimization only considers a small number of alternative positions for the control point, it cannot make dramatic changes in configuration to arrive at a globally optimal configuration.

The current system would also benefit from a theoretical basis for the selection of the gains applied to the different elements of the energy function.

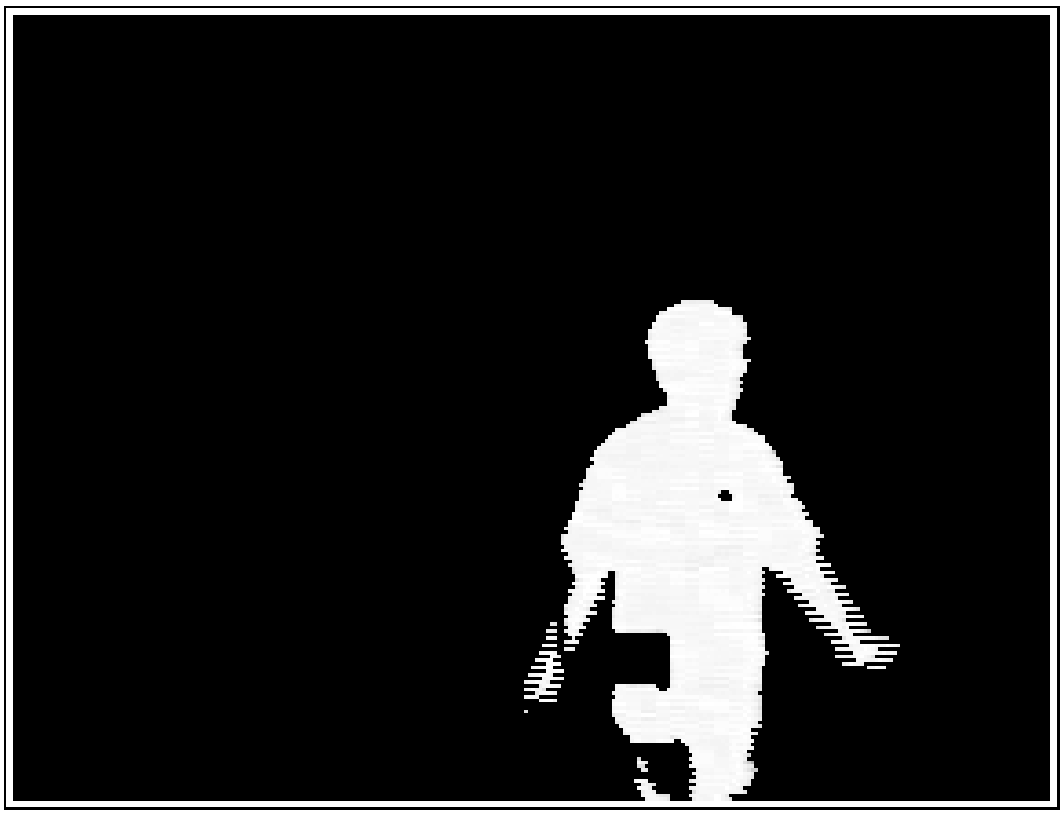

Figure 19: The difference image which provides image forces for the active deformable model.

Presently, these gains must be empirically determined for each application, by observing the behavior of the active deformable model in action and adjusting parameters to overcome performance deficiencies.

Empirically determined gains have given satisfactory results, but a theoretical framework for gain selection would allow for the automatic determination of gains, which will be necessary for deployment if such systems are to be used successfully in commercial manufacturing settings.

9 Future Work

There are a number of promising areas for the further development of this system. These include further exploration of the performance of the algo-

rithm described here and enhancements to the system. These enhancements may either increase the robustness of the system or extend its capabilities.

One issue which should be further explored is the necessity of using difference images as the input to the placement and energy minimization algorithms. If we can assume that more prior information is available about the shape, color or texture of the object, then an alternative placement algorithm could be developed. If color or texture is known, then a different segmentation routine could be used. If shape is known, then a Generalized Hough Transform could be applied to an edge-detected image. The energy minimization algorithm relies on the difference image to provide image segmentation for the E_{model} term in Equation (11). It is also used as an input to the edge detection process, but this design decision was made solely to increase ease of implementation. When new placement routines are available, the minimization algorithm should be tested with raw grey scale image data.

More experiments should also be done to determine whether the mean of control points is the most useful definition of the center of active deformable model. Although the mean is very simple to compute, it directly reflects the location of the control points—not the location of the entire shape. Consider that there are many sets of control points that define the same boundary (when control points are allowed to be collinear, which they frequently are). These sets of control points do not, however, have the same mean. If the location of a model is defined as the center of mass of the shape defined by its boundary, then the location of the model is invariant across these different sets of control points.

System robustness can be improved by arriving at a reliable measure for system failure. One such measure for the energy minimization technique described in this chapter is a "cross-over" in the active deformable model. As mentioned previously, when the model is not a simple polygon, the E_{model} term no longer works in concert with the other energy terms, which frequently leads to uncontrollable expansion of the model. If a computationally inexpensive check could be devised for violation of this condition, the system could be stopped, and new control points selected.

Finally, the ability of the system to move relative to the target object can be enhanced by making better use of the information available in the momentary configuration of the active deformable model. Currently, only the location of the mean of the model control points is recovered. By using the relative positions and distributions of the control points, the control input can be extended to take into account apparent scaling or skewing of the model

points. For example, increases in the model scale should correlate inversely with decreases in the distance from the object to the camera. Theoretical groundwork for this extension exists in the previous work of Colombo [9] and Andrew Blake's group [8].

10 Conclusions

We have presented an approach to visual servoing using active deformable models to track image contours. We use these models to track the boundaries of the object's image in the difference image. By tracking the object's contour, we avoid some difficulties associated with visual servoing techniques which track features, such as the occlusion of features or changes in the features due to object deformations. Moreover, because we close the control-loop by using partial solutions from an iterative technique, the movement of the manipulator actually simplifies the task of the process which tracks the object using active deformable models. To illustrate the potential of our algorithms, we implemented them on the MRVT system and presented a detailed description of their real-time performance.

Acknowledgments

This work has been supported by the National Science Foundation through Contracts #IRI-9410003 and #IRI-9502245, the Center for Transportation Studies through Contract #USDOT/DTRS 93-G-0017-01, the Minnesota Department of Transportation through Contracts #71789-72983-169 and #7 1789-72447-159, the Department of Energy (Sandia National Laboratories) through Contracts #AC-3752D and #AL-3021, the 3M Corporation, the Army High Performance Computing Center and the Army Research Office through Contract #DAAH04-95-C-0008, the McKnight Land-Grant Professorship Program, and the Department of Computer Science of the University of Minnesota. Michael Sullivan has also been supported by an NSF Graduate Fellowship in Visual Perception and Motor Control.

References

[1] P. K. Allen, B. Yoshimi, and A. Timcenko. Real-time visual servoing. In

Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 851–856, April 1991.

[2] A. A. Amini, T. E. Weymouth, and R. C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):211–218, 1990.

[3] A. Blake and A. Yuille. *Active Vision*. MIT Press, Cambridge, 1992.

[4] A. Blake, R. Curwen, and A. Zisserman. A framework for spatiotemporal control in the tracking of visual contours. *Int. Journal of Computer Vision*, 11(2):127–145, 1993.

[5] S. A. Brandt, C. E. Smith, and N. P. Papanikolopoulos. The Minnesota Robotic Visual Tracker: a flexible testbed for vision-guided robotic research. In *Proc. of the 1994 IEEE Int. Conf. on Systems, Man and Cybernetics*, pp. 1363–1368, San Antonio, 1994.

[6] J. J. Brault and R. Plamondon. Segmenting handwritten signatures at their perceptually important points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, pp. 953-957, 1993.

[7] F. Chaumette and P. Rives. Vision-based-control for robotic tasks. In *Proc. of the IEEE Int. Worshop on Intelligent Motion Control*, pp. 395–400, 20-22 August 1990.

[8] R. Cipolla and A. Blake. Surface orientation and time to contact from image divergence and deformation. In *ECCV '92*, pp. 187–202. Springer-Verlag, 1992.

[9] C. Colombo, B. Allotta, and P. Dario. Affine visual servoing: A framework for relative positioning with a robot. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 464–471, 1995.

[10] P. I. Corke and R. P. Paul. Video-rate visual servoing for robots. Technical Report MS-CIS-89-18, Grasp Lab, Department of Computer and Information Science, University of Pennsylvania, February 1989.

[11] E. D. Dickmanns and A. Zapp. Autonomous high speed road vehicle guidance by computer vision. In *Proc of the 10th IFAC World Congress*, July 1987.

[12] E. D. Dickmanns. Computer vision for flight vehicles. *Z. Flugwiss, Weltraumforsch*, 12:71–79, 1988.

[13] J. T. Feddema and C. S. G. Lee. Adaptive image feature prediction and control for visual tracking with a hand-eye coordinated camera. *IEEE Trans. on Systems, Man, and Cybernetics*, 20(5):1172–1183, 1990.

[14] B. K. Ghosh. Image based estimation problems in system theory: motion and shape estimation of a planar textured surface undergoing a rigid flow. In *Proc. of the American Control Conf.*, pp. 1322–1326, 1993.

[15] R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision*. McGraw-Hill, Inc., New York, New York, 1995.

[16] B. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *Int. Journal of Computer Vision*, 1(4):321–331, 1987.

[17] A. J. Koivo and N. Houshangi. Real-time vision feedback for servoing of a robotic manipulator with self-tuning controller. *IEEE Trans. Systems, Man and Cybernetics*, 21(1):134–142, 1991.

[18] K. N. Kutulakos and C. R. Dyer. Recovering shape by purposive viewpoint adjustment. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 16–22, 1992.

[19] R. C. Luo, R. E. Mullen Jr., and D. E. Wessel. An adaptive robotic tracking system using optical flow. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 568–573, 1988.

[20] N. P. Papanikolopoulos, P. K. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision. *IEEE Trans. on Robotics and Automation*, 9(1):14–35, February 1993.

[21] I. Pavlidis, M. J. Sullivan, R. Singh, and N. P. Papanikolopoulos. Improving the performance of model-based target tracking through automatic selection of control points. In *Proc. of the Intl. Symposium on Robotics and Manufacturing*, pp. 711-716, 1996.

[22] I. Pavlidis and N. P. Papanikolopoulos. A curve segmentation algorithm that automates deformable-model-based target tracking. Technical Report TR 96-041, Department of Computer Science, University of Minnesota, 1996.

[23] C. A. Richards, C. E. Smith, and N. P. Papanikolopoulos. Detection and tracking of traffic objects in IVHS vision sensing modalities. In *Proc. Fifth Annual Meeting of ITS America*, pp. 453-461, 1995.

[24] A. C. Sanderson L. E. Weiss and C. P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal of Robotics and Automation*, 3(5):404–417, October 1987.

[25] M. J. Sullivan, C. A. Richards, C. E. Smith, O. Masoud, and N. P. Papanikolopoulos. Pedestrian tracking from a stationary camera using active deformable models. In *Proceedings of the Intelligent Vehicles '95 Symposium*, pp. 90-95, 1995.

[26] M. J. Sullivan and N. P. Papanikolopoulos. Using active deformable models to track deformable objects in robotic visual servoing experiments. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pp. 2929-2934, Minneapolis, Minnesota, April 22-28, 1996.

[27] M. M. Trivedi, C. Chen, and S. B. Marapane. A vision system for robotic inspection and manipulation. *Computer*, 22(6):91–97, June 1989.

[28] B. H. Yoshimi and P. K. Allen. Visual control of grasping and manipulation tasks. In *Proc. of the 1994 IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, pp. 575–582, Las Vegas, 1994.

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