Visual Control of Grasping and Manipulation Tasks

Billibon H. Yoshimi and Peter K. Allen
Department of Computer Science
Columbia University, New York, NY 10027 *

Abstract—This paper discusses the problem of visual control of grasping. We have implemented an object tracking system that can be used to provide visual feedback for locating the positions of fingers and objects to be manipulated, as well as the relative relationships of them. This visual analysis can be used to control open loop grasping systems in a number of manipulation tasks where finger contact, object movement, and task completion need to be monitored and controlled.

I Introduction

Hand-eye coordination is a demanding robotic task. While such coordination is common in higher level animals, it is still an elusive goal for multi-sensor robots. A key component of this is to fuse both vision and touch information in real-time. Such fusion will allow on-line grasp planning and manipulation. This paper addresses the vision aspect of this sensor fusion problem, specifically integrating visual feedback for control of grasping and manipulation. We motivate the application of vision to grasping with the following example taken from manufacturing and assembly tasks. In most manufacturing tasks, it is necessary to move parts together in useful configurations so as to make the assembly process more efficient. For example, moving fingers to surround a nut or moving a grasped nut to a bolt. In the case of grasping a nut, the robot must locate the nut, move its gripper to the vicinity of the nut, locate the best grasp points on the nut, servo the fingers of the gripper to those grasp points, and finally, verify that the nut is securely grasped. Usually this variety of task is performed by blind robots using world coordinates, jigs and other devices to remove object positioning ambiguity. Systems built using this kind of open-loop, pre-defined knowledge control usually require large start-up costs in pre-production measurement, setup and testing. These systems also exhibit inflexible, brittle qualities. If small manufacturing design changes are made, the cost of replanning the robot assembly line (including retooling, rejigging and revision of the robot control strategy) can be prohibitively expensive.

Our research is aimed at using vision to provide the compliance and robustness required by assembly operations without the need for extensive analysis, detailed knowledge of the environment or direct physical contact to control a complicated grasping and manipulation task. Using a visual sensor, the spatial arrangement of objects in the environment can be understood without disturbing the environment, and can provide a means for providing robust feedback for a robot control loop. This paper further explores the idea of visual control of grasping, within the specific context of using vision as a complement to other sensory information relevant to the grasping task.

*This work was supported in part by DARPA contract DACA-76-92-C-0007, NSF grants IRI-86-57151, CDA-90-24735, Toshiba Corporation, North American Philips Laboratories, Siemens Corporation and Rockwell International.

II Related Research

The grasping task is complex requiring several steps to setup, execute and monitor. Sobh and Bajcsy [9] take an interesting approach to solving this problem. They examined how finite state machines can be used to monitor the grasping process. They directly observe a gripper performing the manipulation and based on the configuration of the gripper and the part to be manipulated, they compute the sequence of moves necessary to perform the grasp. By monitoring the fingers and object as the task is performed they can determine the state of the task, which states are reachable from the current state, and the move necessary to complete the task. Their technique allows the system to dynamically allocate feature detectors to look for events which trigger state transitions. The system's ability to create and monitor goals also makes it an ideal platform for directing the type of method it can use to achieve those goals. Another approach, given by Hollinghurst and Cipolla [5] uses a combination of control techniques to perform visual grasping. They determine an object's initial position in the environment using an affine stereo transform. Since the affine transformation does not capture the non-linearities of the camera systems, the robot-gripper system's precision is only modestly accurate. To servo the gripper to the object from this position, they use a second control technique which converts the relative distance from the gripper to the object in image space into a compensating movement in real space. In this way, they get around the errors associated with the linearities of the affine stereo transform. Blake [2] has developed a computational model of hand-eye coordination that develops a qualitative theory for the classification of grasps that utilizes dynamic image contours. In other approaches, Murphy et al. [7] described a system for effecting stable grasps based on perceived object orientation. Sharma et al. [8] redefined grasping tasks in terms of the intersection of a 3D surface describing the working configurations of the gripper and robot system and the 3D surface describing the configurations of the object. Sharma's system plans path trajectories in terms of movements along these surfaces. Our work can be seen as computing how a subtask — or in Sobh and Bajcsy's case, a state transition — should be accomplished, and that we give a method for determining how well the subtask is being performed (i.e. feedback).

III Vision as a Feedback Mechanism for Grasping

Human experience provides an existence proof for the ability of vision to assist in grasping and manipulation tasks. Vision provides rich knowledge about the spatial arrangements (i.e. geometry and topology) of objects to be manipulated as well as knowledge about the means of manipulation (i.e. the fingers of a robotic hand). Our goal is to visually monitor and control the fingers of a robotic hand as it performs grasping and manipulation tasks. Our motivation is the general lack of accurate and fast sensory feedback for most robotic hands. Many grippers lack sensing, particularly at the contact points with objects, and rely on open loop control to perform grasping and manipulation tasks. Vision is an inexpensive and effective method to provide the necessary feedback and monitoring for these tasks. Using a vision system, a simple uninstrumented gripper/hand can become a precision device capable of position and possibly even force control. Below, we outline some aspects of visual control that are well suited to the grasping problem:

1. Visual determination of grasp points. This is a preliminary step before grasping , and may not be as time critical as the actual manipulation itself.
2. Image-space reasoning and planning in unstructured and moving environments. Vision-based techniques can be useful where model-based knowledge may be unavailable or errorful. This is an example of the active vision paradigm.
3. Once a grasp has been effected, vision can monitor the grasp for stability. By perturbing the fingers, we can measure the object's movement in image space. If the object does not move as predicted, we use this information to update our perceptual grasp framework.
4. Visually monitoring a task will give us the feedback necessary both to perform the task as well as to gauge how well the robot performed the task, or if an error has occurred.

While visual control of grasping can be very helpful, we need to recognize some problems associated with it. The problems listed below need to be adequately addressed in order to successfully control grasping using vision, and are at the crux of why this is a difficult robotics problem.

1. Grasping and manipulation need real-time sensory feedback. Vision systems may not be able to provide the necessary analysis of the image and computation of an actuator movement fast enough.
2. In grasping with a robotic hand, multiple fingers need to be employed. This entails having the vision system follow multiple moving objects in addition to the possible movement of any object to be manipulated.
3. Grasping and manipulation usually require 3-D analysis of relative relationships of fingers and objects. Simple vision systems only provide a 2-D projection of the scene.
4. As fingers close in on an object to be manipulated, visual occlusion of both the object and fingers can easily occur.
5. An important component of most grasping tasks is the knowledge of forces exerted by fingers. Vision systems can not directly compute accurate force measurements.

Our current research is aimed at finding solutions to these problems. For some tasks, visual feedback methods may only supplement contact/force sensing. In the next section, we discuss a method that allows fingers and objects to be tracked. This method can be used to track multiple fingers and objects, alleviating some of the problems discussed above. The manipulation tasks we have experimented with are primarily 2-D in nature; however, by using a variety of methods, including stereo [5], these results can be applied to 3-D scenes.

The problems of occlusion and force estimation are subjects of our ongoing research, but they will not be discussed in detail in this paper. The problem of visual occlusion may be alleviated through the use of active vision and multi-camera techniques, in which the vision system moves to keep designated features and objects in view [1]. Additionally, we believe it may be possible to estimate grasping forces from vision alone, given an appropriate spring model of force/displacement and knowledge of the actuator forces.

Visual control of grasping is not a panacea. As stated before, fusion of vision and local contact/force information is needed for truly precise control of grasping and manipulation. The work described in this paper is aimed at highlighting what vision can provide. This work can be extended to model the interplay of vision and touch for more complex tasks, including the analysis of partially occluded regions of space and complicated multifingered grasps.

IV Tracking Fingers and Objects Using Snakes

The most important attribute for a controller using visual-feedback is having primitives which can accurately segment and track objects in image space. Depending on the complexity of the experiment, this can range from tracking a nut, to tracking the fingers of a robotic hand, to tracking a complete robot.

A useful tracking primitive is a snake. Kass et al. [6] originally developed snakes, a model for representing image contours which allows them to be easily manipulated by higher level processes. The central idea behind a snake is that it is a deformable contour that moves under a variety of image constraints (which tend to be local) and object-model constraints. The representation of a snake is $v(s) = (x(s), y(s))$ where s runs from 0 to 1 over the perimeter of the snake. The snake is controlled by minimizing the following function

$$E_{snake} = \int_0^1 E_{int}(v(s)) \quad + \quad E_{image}(v(s)) \\ + \quad E_{con}(v(s))ds \quad (1)$$

E_{int} represents the internal energy of the snake caused by bending or discontinuities. E_{image} are the forces caused by image features. E_{con} are external constraints such as lines, edges and terminations.

Several researchers have investigated the use of snakes for tracking contours. They have been used in many different image-processing applications ranging from tracking of medical imagery to facial expression modeling. The work most directly related to ours involves the real-time computation of snakes and includes Williams and Shah [11], Cipolla and Blake [3], Curwen and Blake [4] and Hollinghurst and Cipolla [5].

We use a discrete formulation of snakes (basically replacing the continuous $v(s)$ with v_i's (a set of control points with corresponding coordinates (x_i, y_i)). [1] The energy equation for our snake is:

$$E = \sum_{i=0}^{m}(\alpha E_{cont}(v_i) \quad + \quad \beta E_{curv}(v_i) + \gamma E_{image}(v_i) \\ + \quad \delta E_{balloon}(v_i)) \qquad (2)$$

where m is the total number of control points, α, β, γ, and δ are weights associated with each of the snake energies, and E_{cont},E_{curv},E_{image},and $E_{balloon}$, are the four energy terms affecting the shape of the snake. The first three terms in the summation are similar to those used by Williams and Shaw (referred to later as WS). We added the fourth term, $E_{balloon}$, to compensate for some of the problems associated with the WS snake energy formulation.

[1] In this discussion, we will define the vector $\vec{v}_i$ as the backward difference $(v_i - v_{i-1})$.

A Snake Energy Terms

To facilitate the calculation of snake energies, we make the following assumptions about the world:

- The objects we want to track are contiguous, slow-moving, and uniformly-colored.
- The world is 2-D with no occlusions.
- The snake points are ordered such that if the snake elements are traversed in increasing order $(v_i,\ v_{i+1},\ \ldots)$, the object we are tracking with the snake will be surrounded with a counter-clockwise wrap (i.e. the object volume is to the left of the snake.)

The computation of the snake proceeds in the following fashion: first, a set of control points is determined to approximate the tracked object's contour. This can be done by hand or by simple segmentation based upon geometry or intensity. Secondly, each control point is used to compute an energy term that evaluates the energy change caused by moving the control point within a local pixel neighborhood of size $2w$ x $2w$. At each iteration, the control point is moved to the new position that reduces the total energy of the snake.

E_{cont} is the global regularization term, one which regulates the shape and sampling of the snake. Our implementation favors uniform snake segment lengths:

$$E_{cont}(\vec{v}_i) = \max\left(|\bar{d} - |\vec{v}_i||, |\bar{d} - |\vec{v}_{i+1}||\right) \qquad (3)$$

where $\bar{d}$ is the average segment length. This formulation removes directional bias from the continuity constraint. To prevent oscillation between adjacent minima, we added the following additional modification:

$$E_{cont}(v_i) = \begin{cases} E_{cont} & \text{if } (Econt - .1\bar{d}) > 0.0 \\ 0.0 & \text{otherwise} \end{cases} \qquad (4)$$

This formulation creates a small region of equal minimum energy around a snake control point. At this point, E_{cont} becomes less important and the other energy terms dominate the energy function.

E_{image} is the term which determines of how image energy affects a snake. Typically, this is an attractive force in which image features (edges, corners, intensity extrema etc.) are used to attract the snake. We have defined E_{image} as the negative magnitude of the image gradient or $-(|\delta x_i| + |\delta y_i|)$. This measure can be computed at frame rates in the adjacent neighborhood of the control point.

E_{curv} is a measure of a snake's curvature at a given control point. This term is used to create smoothly varying contours. The following formulation results in a constantly increasing function which increases as curvature increases.

$$E_{curv}(v_i) = \left|\frac{\vec{v}_i}{|\vec{v}_i|} - \frac{\vec{v}_{i+1}}{|\vec{v}_{i+1}|}\right|^2 \qquad (5)$$

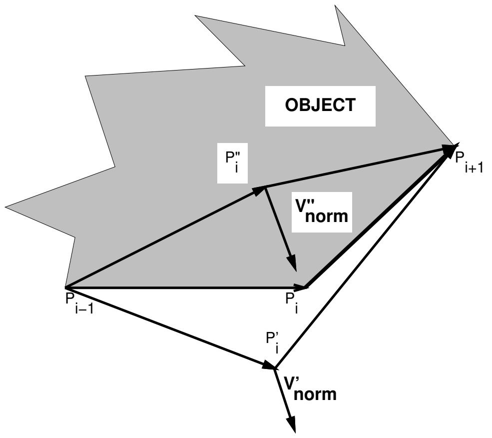

Figure 1: Example demonstrating the directions favored balloon forces for different vertex positions

While the E_{curv} term is not curvature, (κ), it allows us to determine how perturbing a control point cause the snake to bend more or bend less compared to neighboring points.

$E_{balloon}$ is the term which exploits the intrinsic properties of the object to improve tracking performance. Our formulation contains an interesting combination of local and global computations which allows us to use one local-area image operation and snake information to compute the energy distribution for an area surrounding a snake control point. The $E_{balloon}$ term uses the object's color to help determine whether or not a pixel is part of the object. The idea is to examine the neighborhood of a control point and to try to push the contour away from the interior of the object if the control point is inside the object, and to contract the contour towards the interior of the object if the control point is outside the object's boundary ("ballooning"). We use a simple neighborhood threshold around a control point to determine if the region surrounding a control point is inside or outside the object's boundary. If over ϵ% of the pixels in a region have the same characteristic, the control point associated with that region is classified as having that characteristic. The objects we are tracking are homogeneous in color, and by calculating the number of pixels in the neighborhood of the control point that are object pixels, we can associate a direction (in or out) to move the snake's boundary.

The actual movement direction is found by computing an approximation to the contour normal at the control point under investigation. The normal direction is the bisector of the angle formed between the two snake points adjacent to the control point, $\angle p_{i-1}p_ip_{i+1}$ in Figure 1. The IN or OUT designation determines if we move along the positive or negative normal direction.

Figure 1 demonstrates the three situations where $E_{balloon}$ evaluates differently. These cases are:

1. INSIDE: p''_i lies inside the object. Sampling a small region surrounding p''_i shows that the color of the region is identical to the color of the object. The contour underestimates the true shape of the object. V''_{norm} is the normal to the boundary at point p''_i. The IN side of the plane is on the left side of the directed line segment, $\overline{p_{i-1}p''_i}$. The OUT side of the plane is on the right. We give all potential control points on the IN side a weight of +1 (high energy) and those of the OUT side, -1 (low energy). This weighting favors any movement which would take the point away from the object.
2. OUTSIDE: p'_i lies outside the object. Sampling a small region surrounding p'_i shows that the color of the region is unlike the color of the object. The contour overestimates the true shape of the object. V'_{norm} is the normal to the boundary at point p'_i. The IN side of the plane is on the left side of the directed line segment, $\overline{p_{i-1}p'_i}$. The OUT side of the plane is on the right. We give all potential control points on the IN side a weight of -1 (low energy) and those of the OUT side, +1 (high energy). This weighting favors any movement which would take the point toward the object.
3. EDGE BORDER: p_i lies on the border between the object and the outside world. In this case, the entire region is given the same weighting, 0.

The goal of $E_{balloon}$ is to keep the snake from shrinking in upon itself. Snake formulations without the $E_{balloon}$ force tend to collapse in upon themselves when an object disappears or when an object moves too quickly.

B Collision Detection

To properly control grasping We have also implemented a brute-force, collision detection algorithm (to determine when snakes collide). In the experiments that follow, we track both fingers and objects to be grasped. We need to compute contacts between fingers and objects, and this can be done by determining if two snakes, S_1 and S_2, are adjacent to one another by the following algorithm.

1. Find the minimum distance (point-to-point) for each pair of points (p_1, p_2), where $p_1 \in S_1$ and $p_2 \in S_2$.
2. Find the minimum distance (point-to-line) for each pairing of points to lines (p_1, l_2), where $p_1 \in S_1$ and l_2 is composed of 2 adjacent points in S_2.
3. Find the minimum distance (line-to-point) for each pairing of points to lines (l_1, p_2), where l_1 is composed of 2 adjacent points in S_1 and $p_2 \in S_2$.

4. Take the minimum of the three values found in the previous steps. If the value is less than 1 then there is a high probability (within image noise limits) that the snakes are adjacent to one another.

C Experimental Procedure

Our experimental system is shown in figure 2. A static monocular camera system is used to monitor and control a multi-fingered robotic gripper. The gripper for this experiment is the FMA (Flexible Micro Actuator) gripper [10], provided by Toshiba Corporation and pictured in figure 3). Each finger of the FMA gripper is controlled by three servo valves which either inflate or vent air to three chambers in the finger. By varying which valves are closed in each finger at any one time, it is possible to change the position of each finger to one of eight positions. The fingers are sensorless. It is impossible to determine if an action has taken place, whether a pressure has been exerted, or what position a finger is in at any time interval. The FMA gripper also exhibits an extraordinary amount of compliance in performing tasks. Since its fingers are compliant, it is possible to perform tasks using the FMA gripper which are not possible using a simple parallel jaw apparatus. In addition, the fingers have high friction coefficients allowing them to grip and exert forces on objects with little slippage, qualities which make the FMA gripper excellent for grasping tasks.

For our experiments, the neighborhood size on image operations was 5 x 5, the balloon threshold (ϵ) was set at 90%, and the energy weighting terms α, β, γ and δ were respectively 1.0, 1.0, 1.1 and 1.0 (these were established experimentally and provided good, stable tracking). The initial positions of the snake control points are placed by hand as a convenience; other methods of estimating the initial contour are possible. We have also changed the method used by WS to update snake parameters. In WS, snake updates were performed segment by segment in order from the lowest numbered snake segment to the highest. This method tends to bias the snake along the direction of travel. Since modifications were performed directly on the snake, lower numbered snake control points directly affected the energy calculations of higher numbered points. Instead of updating the snake control points sequentially, we investigated all control points simultaneously to determine the best positions to move. Once all the points were evaluated, we made one trip around the snake making sure that the new positions of snake control points were valid (non-overlapping). This new formulation seems to be amenable to parallel implementation (a possibility we may wish to explore in the future).

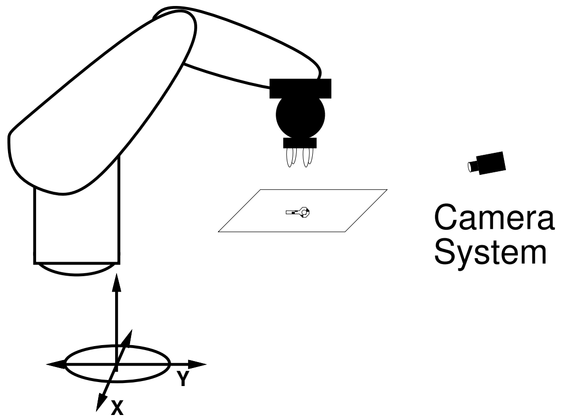

Figure 2: Experimental setup for the visual control of grasping experiment

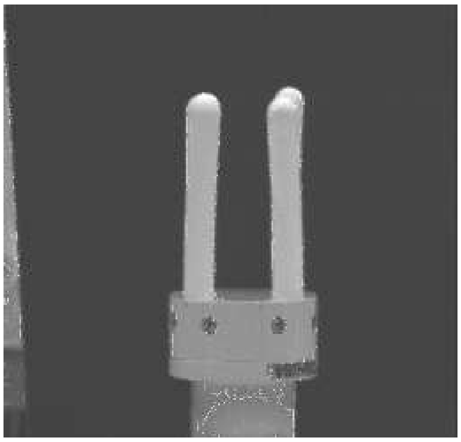

Figure 3: Toshiba Gripper

V Touch Experiment

The first experiment involved using snakes to track a finger on a gripper. The goal was to evaluate snake dynamic tracking performance as the finger moved over a sequence of image frames. In this experiment (shown in figure 4), the finger moved in an upward direction, until contact was make with the block. Snakes tracked the movements of both the finger and the block. The distance between the finger snake and block snake was used to determine when contact has been made between the finger and block.

The finger and the block positions were both estimated initially by hand-picked snake control points. Both normal and Sobel-edge imagery were computed at frame rates for input to the snake fitting algorithm. Between successive images, the snake fitting routine was called a maximum of 20 times for each snake. In many several cases, the snakes settled into positions where several snake segments oscillated between local minima. Using a hard-limit on the number of iterations does not appear to have hampered the snake's ability to track reliably in these situations, since the local minima happened to be within a pixel of each other. Figures 5, 6, and 7 show the

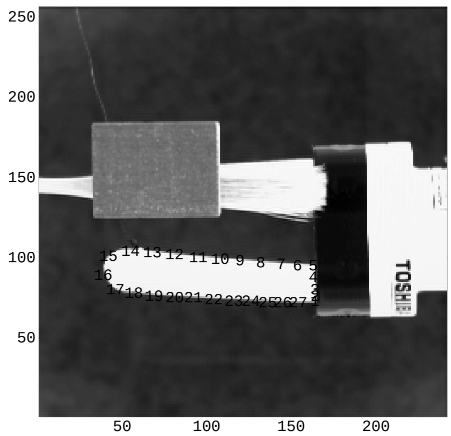

Figure 4: Initial position for touching experiment. Finger snake control points are numbered. The snake attached to the block is not shown.

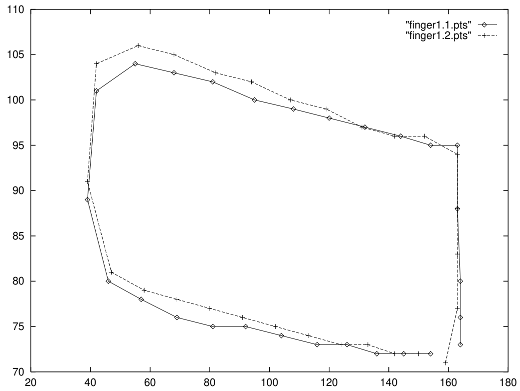

Figure 5: First and second moves. (not drawn to scale)

positions of the snakes as they track the finger over 6 frames from the fingers initial position shown in figure 4 to the finger's final resting position on the block shown in figure 8. Notice that though out this operation, the snake representation succinctly describes the pose of the finger. In addition, we have also verified that snakes can be used to detect contact between different objects. In figure 9, we show the final resting positions of the two snakes. In this situation, our collision detection algorithm reported a distance of 0 pixels between the two snakes (a point on the finger snake actually lies on a line segment contained by the block snake).

VI Grasping and Extraction Experiment

The following experiment tested visual control in two different situations. In the first part of our experiment, the gripper closed two fingers around an object while at the same time the vision system monitored the finger-object-finger system to determine when a stable grasp has taken place. In the second part, the robot translated the grasped object along its Z-axis. During this translation, the vision system

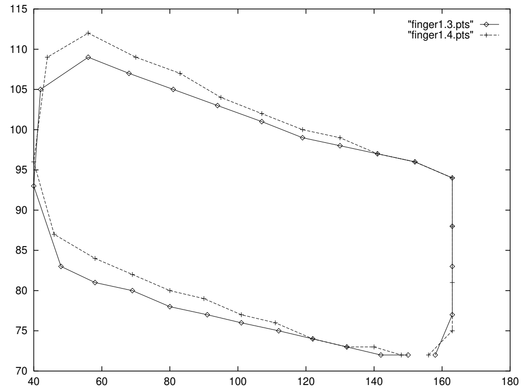

Figure 6: Third and fourth moves. (not drawn to scale)

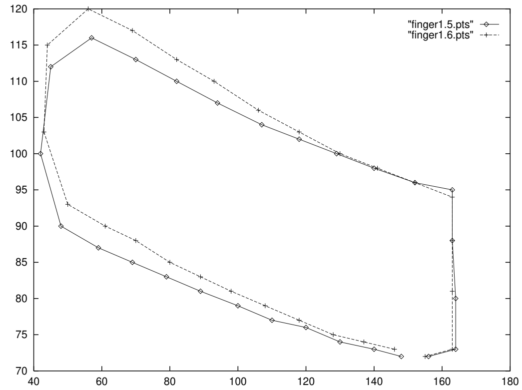

Figure 7: Fifth and sixth moves. (not drawn to scale)

monitored the positions of the fingers and object to determine the status of the translation. The two parts of this experiment highlight the utility of a visual control system. First, the grasping and translation tasks both require the vision system to determine the position and orientation of the fingers and the object to be manipulated. Second, in both of

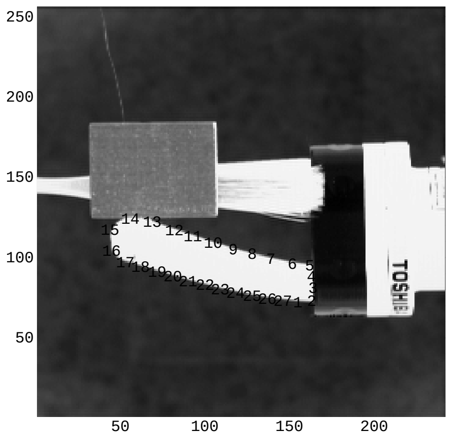

Figure 8: Final positions of finger and block snakes for the touching experiment

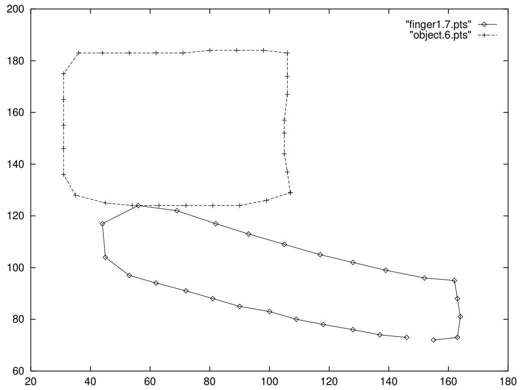

Figure 9: Positions of block and finger snakes in final position.

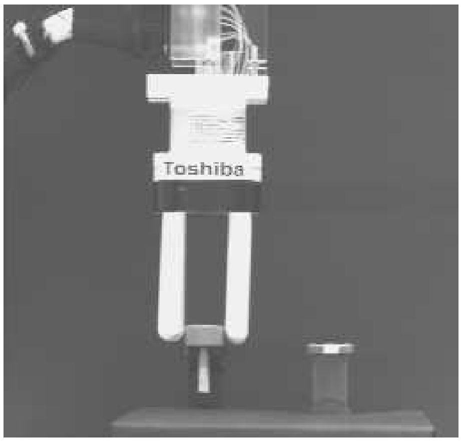

Figure 10: Initial configuration of gripper and bolt to be extracted. Although difficult to see at this resolution, the fingers are not in contact with object.

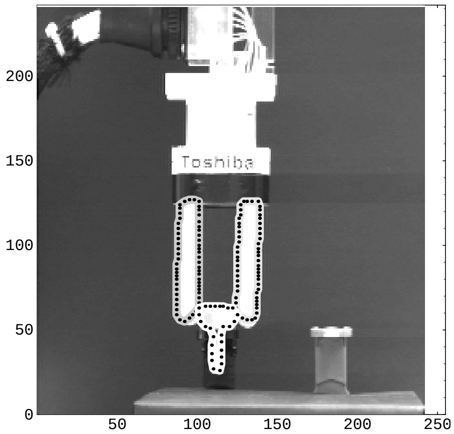

Figure 11: Initial positions of objects and snakes for grasping and extraction experiment

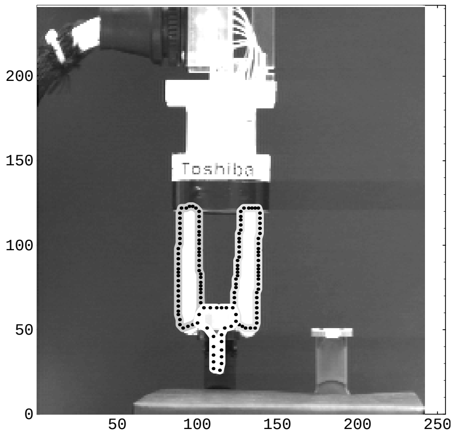

Figure 12: The final position of fingers, bolt and snakes after the grasping portion of the experiment.

these tasks, the vision system is required to generate a report on how well the robot is performing the task and/or how close the task is to completion. We have constrained this experiment to lie in 2 dimensions. In addition, the initial position of the gripper and the object translation path are known.

In the first part of the experiment, we use two of the fingers to grasp the bolt. Figure 10 shows the initial configuration for this experiment. Figure 11 shows the positions of the fingers and bolt as well as their associated snakes in image space. The goal of the grasping section is to monitor the finger snakes and bolt snake as the fingers close in on the bolt. As the fingers are slowly closed, we monitor the distance between the finger snakes and the bolt snake with our collision detection procedure. To close the fingers, we gradually increased the pressure to the fingers between image samples. When the collision detection algorithm reported less than 1 pixel separating both fingers from the bolt, we determined that grasping had occurred. At this point, we increased the pressure a small fraction (since the contact imposed by adjacency is usually a low-force contact, incapable of holding the grasped object) to form a stronger grasp. Figure 12 shows the final position of the fingers and their snakes at the end of a typical grasping operation.

The next part of our experiment monitored the position of the finger-bolt-finger system while the fingers were translated along the robot's Z-axis. This part of the experiment is a simplified version of a general procedure for monitoring extractions. We verify the state of the extraction process by monitoring the positions of the bolt and fingers. If the bolt and fingers do not move in the same direction or if the fingers lose contact with the bolt, we say that the bolt is not grasped and the system must try again. In figure 13, we show the results of the extraction procedure given the initial starting state shown in figure 12.

VII Conclusions

This paper has motivated the use of visual control for grasping tasks. While this is a difficult research problem, it appears that there may be useful meth-

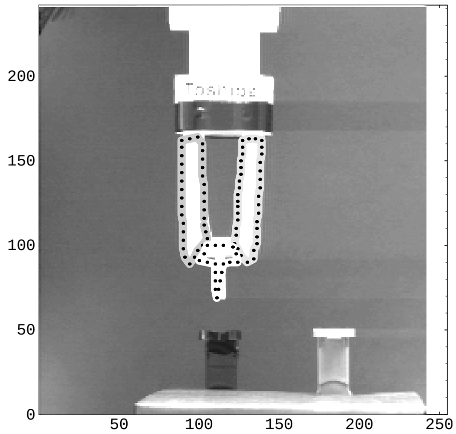

Figure 13: Final positions of the fingers, bolt, and snakes at the end of the extraction experiment.

ods that are fast and robust enough to close a feedback loop using a vision sensor to control grasping and manipulation. These methods are also consistent with sensor integration from other modalities such as force and contact on the gripper itself. In the absence of such sensing (which is common with simple robot grippers), visual control methods may suffice for a number of simple manipulation tasks.

The method described here needs to be extended in a number of ways. First, we are currently extending these results to 3-D analysis by use of stereo (2 monitoring cameras) and active techniques with a single camera [1]. Second, the experimental results shown here can track the finger-bolt-finger snakes at about 1/7 Hertz.; however, we have optimized this method to achieve a speed of approximately 1 Hertz by intelligent buffering of images and transfers to the host. At 1 Hertz, we are approaching the speeds which are necessary to integrate the vision with other sensory feedback. The speed is also a function of the number of control points chosen for the snake. The use of vision for control of grasping has the added benefit of being a relatively simple add on to an existing gripper system. It is also easily reconfigured, and can work in uncalibrated environments (which is the case in our experiments).

References

[1] S. Abrams and P. Allen. Dynamic sensor planning. In *IEEE International Conference on Robotics and Automation*, pages 2–605 – 2–610, May 2-7 1993.

[2] A. Blake. Computational modelling of hand-eye coordination. In *Active Perception*. Lawrence Erlbaum Associates, Inc., 1993.

[3] R. Cipolla and A. Blake. Surface orientation and time to contact from image divergence and deformation. In *ECCV 92*, pages 187–202, 1992.

[4] R. Curwen and A. Blake. Dynamic contours: Real-time active splines. In *Active Vision*, pages 39–57. MIT Press, 1992.

[5] N. Hollinghurst and R. Cipolla. Uncalibrated stereo hand-eye coordination. Technical Report CUED/F-INFENG/TR126, Department of Engineering, University of Cambridge, 1993.

[6] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proceedings of First International Conference on Computer Vision*, pages 259–269, 1987.

[7] T. Murphy, D. Lyons, and A. Hendriks. Visually guided multi-fingered robot hand grasping as defined by schemas and a reactive system. In *USC Workshop on Neural Architectures and Distributed AI: From Schema Assemblages to Neural Networks*, pages 1–5, 1993.

[8] R. Sharma, J. Herve, and P. Cucka. Analysis of dynamic hand positioning tasks using visual feedback. Technical Report CAR-TR-574, Center for Automation Research, University of Maryland, 1991.

[9] T. M. Sobh and R. Bajcsy. Autonomous observation under uncertainty. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1792–1798, May 1992.

[10] K. Suzumori, S. Iikura, and H. Tanaka. Development of a flexible microactuator and its application to robotic mechanisms. In *IEEE International Conference of Robotics and Automation*, pages 1622–1627, April 1991.

[11] D. Williams and M. Shah. A fast algorithm for active contours and curvature estimation. In *CVGIP: Image Understanding*, pages 14–26, 1992.