To Appear: AAAI Fall Symposium on Machine Learning in Computer Vision,
Raleigh, NC, October 22-24, 1993

A Vision-Based Learning Method for Pushing Manipulation

Marcos Salganicoff

GRASP Laboratory
Department of Computer and
Information Science
University of Pennsylvania
Philadelphia, PA, USA
sal@grip.cis.upenn.edu

Giorgio Metta Andrea Oddera
Giulio Sandini

Laboratory for Integrated Advanced
Robotics (LIRA - Lab)
Department of Communication,
Computer and Systems Science
University of Genoa
Via Opera Pia 11A - I16145
Genoa, Italy
giulio@vision.dist.unige.it

Abstract—**We describe an unsupervised on-line method for learning of manipulative actions that allows a robot to push an object connected to it with a rotational point contact to a desired point in image-space. By observing the results of its actions on the object's orientation in image-space, the system forms a predictive forward empirical model. This acquired model is used on-line for manipulation planning and control as it improves. Rather than explicitly inverting the forward model to achieve trajectory control, a stochastic action selection technique [Moore, 1990] is used to select the most informative and promising actions, thereby integrating active perception and learning by combining on-line improvement, task-directed exploration, and model exploitation. Simulation and experimental results of the approach are presented.**

I. INTRODUCTION

Active perception can broadly be defined as the process of information gathering, organization and interpretation by the active and purposive control of sensors, effectors and computational resources in order to carry out a task or set of tasks. Closely related to the notion of active perception is the process of learning, since it holds the promise of being a general purpose method for acquiring task-specific perceptual and effector control strategies in an unsupervised way. While in principle, learning and active perception are well suited for each other, active perceptual tasks demand several properties from learning algorithms; in particular, it is desirable that the algorithms meet the following requirements:

1. That they be *On-line*, meaning that the system improve continually while the task is being attempted, rather than in a *batched* fashion which requires a separate *learning* and *execution* phase. Traditionally, many inductive learning have been batch methods, rather than on-line.

2. Closely related to the on-line requirement is the *continually adaptive* requirement, meaning that the system should adapt to changing task dynamics. Many inductive learning techniques are *one-pass* adaptive, meaning that they are allowed to adapt during the explicit learning phase described in item 1 above, but once the learning phase is over, they do not adapt to any subsequent changes in the task.

3. They should *converge rapidly.* Since an agent or organism has a finite lifetime, and each experimental interaction has cost in terms of time, energy and material, learning techniques should converge to a good sensing and action control policy rapidly in order to make the learning a viable alternative to hand coding of control policies.

4. They should provide an efficient *exploration strategy* which balances the need to explore and characterize task properties versus the need to achieve competence as rapidly as possible. Active perception systems are particularly well suited to benefit from intelligent exploration strategies since they can, by definition, determine which exemplars they create.

In this work we describe the use of a memory-based inductive learning technique which addresses these criteria and learns to perform the visuomotor task of pushing an unknown object with a single rotational point contact to an arbitrary goal point in the visual space.

II. PUSHING MANIPULATION

In many situations it is desirable to move an object from one location to another, but the object may be too large to be lifted by a single agent. Two possible solutions exist, either many agents may cooperate in lifting and moving the object [Bajcsy *et al.*, 1991], or it may be possible for a single agent to push the object instead of lifting it. We explore the pushing case where the contact between robot and the object is single point (see Figure 1.) and the pusher remains within the friction cone of the contact (i.e. only a rotational degree of freedom exists at the pusher/object contact point; this is enforced by notching the object at the contact point in the experiments).

Stable pushing and steering of an object to desired position in the workspace when there is only a point contact between the pusher and the object is a difficult visuomotor control problem since the relationship between the pusher and the object is unstable. Because of this, the object tends to rotate past the pusher if no corrective actions are taken. At the same time, a desired pushing direction must be achieved in order to arrive at the desired point in the robot workspace.

An additional complication is that the object motion resulting from pushing actions is a function of the frictional distribution of the object [Mason *et al.*, 1989] on its surface of support and the mass distribution of the object, which are difficult to measure using only passive visual perception. These quantities can, in general, only be measured with active perceptual procedures [Campos and Bajcsy, 1990; Lynch, 1993]. However, even if these

The research was supported by ESPRIT Projects FIRST and SECOND, the Special Projects on Robotics of the Italian National Council of Research, an NSF Postdoctoral Associateship for MS (CDA-9211136), and by NSF/ESPRIT IRI-9303980

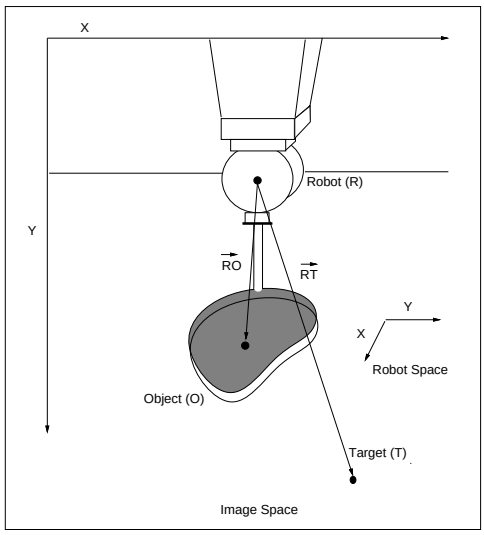

Fig. 1: The pushing task. The pusher and object are connected only with a point contact so that the object can rotate relative to the pusher. The objective is to push the object to the desired point in the image space.

quantities can be estimated, friction is difficult to model analytically in general because of its non-linear behavior.

Rather than estimate parameters and utilize an analytic model of friction, we develop a simple and direct solution by measuring the effects of pushing actions on the image-space orientation of the object relative to the pusher, and then use a learned forward model to select actions.

The pushing and sliding manipulation problem has been studied extensively by Mason [Mason, 1986] from an analytical viewpoint, as well as from a learning perspective [Mason *et al.*, 1989]. Lynch [Lynch, 1993] has explored using visual measurements of object reaction to pushing actions in order to explicitly estimate the center of friction of the object. Zrimec [Zrimec, 1990] implemented a system which generated qualitative models of the effects of pushing actions through experience which were then used for planning.

A. *Steering by Controlled Instability*

Since point-contact pushing is an intrinsically unstable process, one immediate objective might be to null the rotation of the object in order to stabilize it relative to the pusher. However, if the only objective is to zero the object's rotation, then control of steering is impossible, since when this condition is achieved no directional correction is possible and pusher trajectory is fixed. When pushing an object, we desire to null its rotation *only* when it is aligned with the idealized linear trajectory which takes it to the goal point in image-space. When the object is misaligned, the objective is the controlled rotation of the object relative to the pusher in order to bring it in line with the ideal trajectory. This rotation is a controlled instability, since object rotation is a manifestation of the instability of the task.

The above notions can be captured by devising the following trajectory generation procedure. In the image space, let $\vec{RT}$ be the vector between the current center of mass of the locations of flow vectors associated with the robot end-effector and the desired target location in the image space, and $\vec{RO}$ be the vector between the robot and the center of mass of the locations of flow vectors assigned to the object (see Figure 1; segmentation of pusher and robot is discussed in the section describing the experimental implementation.) For trajectory generation, the direction and magnitude of desired rotational velocity $\omega_d = \frac{\Delta\theta_{RO}}{\Delta t}$ of the object is a function of the angle $\theta_{RO,RT}$ between the ideal pushing trajectory $\vec{RT}$ and the current estimate of orientation of the object direction $\vec{RO}$ multiplied by a turn rate gain coefficient k_s:

$$\omega_d = k_s \theta_{RO,RT} \tag{1}$$

Since it is difficult to control large rotation rates in practice, the magnitude of ω is bounded by putting it through a saturation function:

$$\omega_d^{'} = \begin{cases} \omega_{max} & \text{if } \omega_d > \omega_{max} \\ \omega_{min} & \text{if } \omega_d < \omega_{min} \\ \omega_d & \text{otherwise} \end{cases} \tag{2}$$

This desired rotational rate $\omega_d^{'}$ then provides a reference rotation rate which must be achieved by selecting an appropriate robot action v_y in the robot frame using the learned forward state-transition model $\hat{f}$. The x velocity of the robot is constant in the robot-space, thus ensuring that the robot always moves forward and that the pushing trial terminates. The combination of (v_x, v_y) determines the effective pushing direction of the robot, $\vec{P}$.

B. *Learning the Forward Model*

The learning process consists of approximating the forward function $\hat{f}$ from a set of observed input/output pairs. The input consists of pairs θ_{RO} and v_y, where θ_{RO} is the angle of $\vec{RO}$ relative the the x-axis of the image space, and v_y is the y velocity of the robot action in the robot space, both sampled at time t. The resulting output consists of the observed change in orientation $\Delta\theta_{RO}$ observed at time $t+1$. At sample time t, the learning set consists of a set of tuples of the form $(((\theta_{RO}, v_y)_0, (\Delta\theta_{RO})_1), \ldots, ((\theta_{RO}, v_y)_{t-1}, (\Delta\theta_{RO})_t))$.

The state-transition function to be learned is

$$\Delta\theta_{RO} = \hat{f}(\theta_{RO}, v_y) \tag{3}$$

We use a simple one nearest-neighbor (1-NN) technique [Duda and Hart, 1973] to approximate the function, in which the output of $\hat{f}$ is taken to be the value of the exemplar which is nearest to the query point in the input space using a standard Euclidean distance metric. The data is indexed in an *k-D* tree [Friedman *et al.*, 1977] in order to make queries more efficient ($O(log(N))$, where N is the number of exemplars in the database). The insertions are handled by inserting new exemplars into the leaves that they index to, according to the current tree. The data can be copied and a new tree built while the old tree continues to be used. In practice, the tree rebuilding time is small in comparison to the real-time necessary to gather data, and does not present a problem.

C. *Planning, Exploration and Exploitation*

Given the current estimate of the forward pushing function $\hat{f}$ and the current observed input state, a decision must be made as to the next control action to be

issued. During learning with active perception, actions may need to accomplish dual purposes. The first purpose may be *performatory*, or to directly accomplish the change in orientation needed to achieve the current desired trajectory. The second purpose may be *informative*, or to execute actions which yield information about the task. In particular we would like to have a strategy which exploits the model in regions where it is known (well approximated) and it can achieve the desired performatory goals, while exploring actions whose outcomes are not completely characterized and have a some better possibility of achieving the desired next state when no known good actions can be found using the current model. A strategy with these desirable properties is the intelligent experimentation scheme of Moore [Moore, 1990]. This strategy selects actions according to the following heuristic. Let

$$\sigma(\theta_{RO}, v_y) = Cd((\theta_{RO}, v_y)_{\{1\}}, (\theta_{RO}, v_y)) \qquad (4)$$

where C is an exploratory constant, (θ_{RO}, v_y) is the combination of the current observed object orientation and the current randomly selected action under evaluation, $(\theta_{RO}, v_y)_{\{1\}}$ is the first nearest-neighbor to the evaluation point, and d is the Euclidean Norm. Moore's heuristic is based on the assumption that the distribution of predicted outcomes of $\hat{f}$ is Gaussian with mean of $\hat{f}(\theta_{RO}, v_y)$ and that the further the nearest-neighbor from the query point, the wider the variance of its prediction. In other words, if the nearest-neighbor used in the prediction by $\hat{f}$ is very close to the query point, then the prediction should be fairly good, and the spread of outcomes of the prediction should be narrowly distributed around the output value predicted by the nearest-neighbor. However, if we are basing our prediction on a very distant nearest-neighbor, the variance should be quite wide, since the nearest-neighbor is far, and should not be relied upon for a tight prediction. Given that the $\sigma(\theta_{RO}, v_y)$ is selected using equation 4, and a desired goal interval for the next output state is $\Delta\theta_{RO} \pm \delta$, the probability of the output of a candidate action v_y landing in the interval is

$$p = \mathbf{G}\left(\frac{\Delta\theta_{RO} - \hat{f}(\theta_{RO}, v_y) + \delta}{\sigma(\theta_{RO}, v_y)}\right) - \qquad (5)$$

$$\mathbf{G}\left(\frac{\Delta\theta_{RO} - \hat{f}(\theta_{RO}, v_y) - \delta}{\sigma(\theta_{RO}, v_y)}\right) \qquad (6)$$

where $\mathbf{G}(x)$ is the integral under the Gaussian $\mathcal{N}(0,1)$ from $(-\infty, x]$. Therefore, some number of random actions are generated during each control cycle given the desired $\Delta\theta_{RO}$, and the action with the highest p is executed.

III. Results

A. Simulation Results

In order to verify the feasibility of the approach a simple simulation was implemented. In the simulation k_s=.5, C=1.0, v_x=5.0cm, v_y are uniformly distributed over $\pm 10 cm/sec$, 100 random actions are evaluated per control cycle, and δ = .05 radians. The unknown underlying forward function is a linear function of $\theta_{RO,\vec{P}}$, where $\vec{P}$ is the resultant pushing direction of the arm. Figure 2 shows the performance of the learner's control during the first 9 attempts. Beginning with no *a-priori* knowledge of the pushing dynamics of the object, the system learns to push effectively after the first *two* trials, and has converged to very good performance after nine trials. In figure 3 inter-task transfer can be seen, as the first attempt to push to $(-50, 150)$ succeeds by using the forward function estimate generated from learning to push to the previous location in Figure 2. Since memory based learning stores all exemplars, even experiences with actions which failed for the first location may subsequently be useful for other locations [Aboaf *et al.*, 1989].

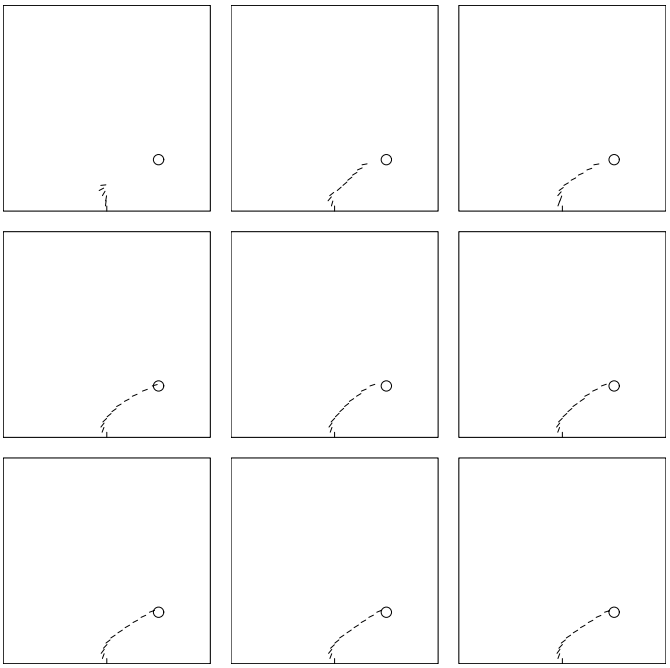

Fig. 2: Simulated Trials. Learning to push to the circle at (50,50) in the work space, beginning with no *a-priori* knowledge about the object dynamics.

B. Experimental System

The experimental setup consisted of a manipulatory and a perceptual component. The manipulatory component consists of a Unimation Puma260 Robot, Unimation Controller, a SparcStation IPC running RCCL [Lloyd, 1986] Sbus/VME Mapper and software which allows for high-speed communication between the Sparc IPC and the Unimation Controller. The perceptual component consists of a VDS EidoBrain 7001 Image Acquisition and Processing system and CCD Camera. Communication is accomplished using TCP/IP sockets, which are adequate for the .8 second update intervals. A manipulation server process exists on the Sparc which servos the most recent rate commands from the VDS at 28 msec intervals and takes care of communication protocols.

The object was notched at the pusher contact point in order to allow only rotational motion between the pusher and the object. The experiment was monitored by an observer and when a trial failed completely and the object rotated past the pusher, it was terminated. Additionally, if the object reached either extreme or the bottom of the image, the trial was halted.

The figure ground segmentation was accomplished by computing the optical flow [Horn and Schunk, 1981] with

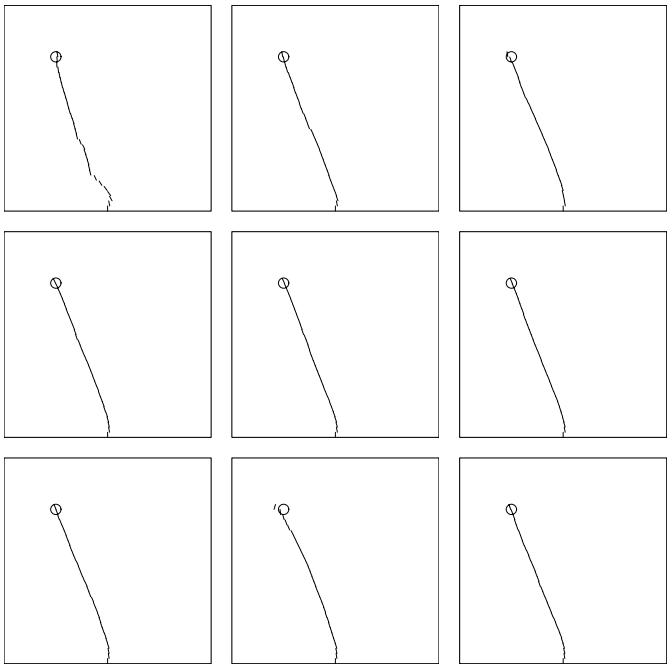

Fig. 3: Simulated Trials. Pushing to point (-50,150) in the work space after completing the first 9 trials in Figure 2 Knowledge from those trials is exploited for pushing to the new location to enable success in the first attempt.

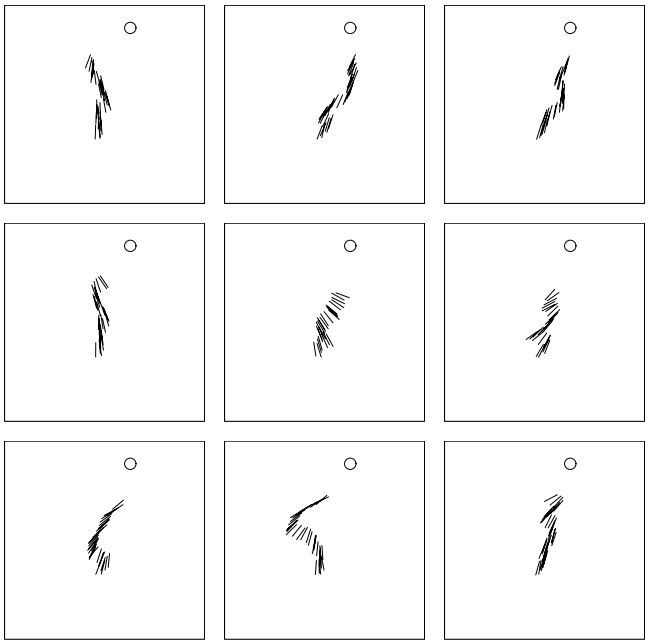

Fig. 4: Experimental Trials. Pushing to point (44,62) in the image-space during the first 9 trials.

recursive temporal filtering over the incoming image sequence and thresholding the flow vectors based on magnitude. Locations with above threshold optical flow are labelled as foreground and others as background.

Once the foreground has been labelled, the segmentation between the pusher and object must be performed. During a brief calibration motion the manipulator is swept through its pushing workspace with no object present, holding the y-component of the end-effector fixed while the x-axis position is moved in the positive x direction. Simultaneously, the end-effector position is tracked in the image space. The robot x-axis positions and their associated image y-axis values are stored and simple linear fit is done to calculate the relation between the two. Later, during the execution of the pushing task, the manipulator position is used to compute the vertical position of the end effector in the image using the linear fit parameters. All flow vectors below the horizontal at this vertical position are associated with the object and vectors above it with the robot. Assuming the object can be held $\pm\frac{\pi}{2}$ of the image y-axis (approximately in front of the pusher) this provides an extremely reliable and simple segmentation method.

C. *Experimental Results*

Some representative sequences of the system's performance in learning to push an object with unknown mass and friction to an image location are shown in Figures 4 and 5. In the actual trials, qualitative control was learned, rather than exact control as in the simulation. The desired turn rate interval to be achieved was either $(-\infty, -.2]$ or $[.2, \infty)$ depending on whether the desired turn direction was negative or positive, respectively. This was done because the estimates of the angles $\theta_{RO,RT}$ in the visual space were coarse and noisy since the optical flow was computed on a subsampled version of the image to speed up computation of flow. Requiring a minimum $\Delta\theta_{RO} = \pm.2$ ensured the angular changes were above the noise floor at the expense of some oscillation in the control. When failures occurred, it tended to be due to errors in measurement of angles which lead to an incorrect sign being specified for the desired turning direction. Because of the large delay (800ms) in computing the 64×64 flow vectors, and the temporal filtering, the actual trajectory tended to oscillate about the desired trajectory.

IV. DISCUSSION AND FUTURE WORK

The utilization of optical flow simplifies the arm/background segmentation problem significantly, assuming a static background. Unlike other approaches for manipulator control in the image space [Mel, 1991] which require identifying and tracking markers on arm joints such as LEDS, or grey level thresholding which is quite sensitive to ambient illumination, flow measures are much more flexible since they do not require explicit tracking, control of illumination or uniform backgrounds.

The segmentation between pusher and object, while effective, is currently done in a rather ad-hoc fashion. Other more general approaches for this problem should be developed. In particular, knowledge of the pusher speed and direction might be utilized as constraints for separating flow vectors arising from the object from those of the pusher.

Nearest-neighbor learning approaches tend to have poor noise immunity and suffer from outliers in the learning sets. In particular, the use of robust regression techniques [Huber, 1981] would serve to enhance performance especially when very noisy visual measures such as normal flow are utilized. Secondly, the number of nearest-neighbors used in the functional estimate is selected by

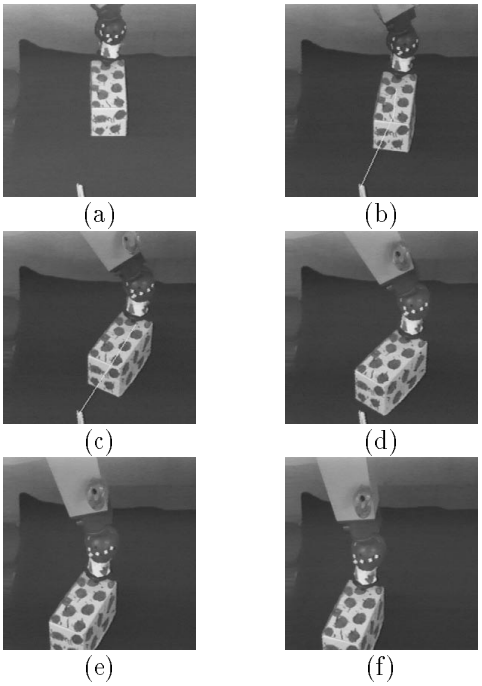

Fig. 5: An image sequence for pushing to a point in the image space to the left.

the user, but can be automatically determined by a cross-validation technique [Stone, 1977].

The optical flow computations are currently only being used for segmentation, but in fact, provide additional information about the rotational rate of the object relative to the pusher [Salganicoff *et al.*, 1993] which can be exploited for control purposes in pushing and insertion. The use of memory-based learning methods for predicting these direct rotational measures, which have the advantage of being independent of object contour, should be explored. The current implementation is only one-pass adaptive (see section I), which is characteristic of memory-based learners. However it can be extended to a continuously adaptive technique by the addition of explicit time-weighted or locally-weighted forgetting algorithms which delete observations from the exemplar database selectively [Salganicoff, 1993]. This would permit it to adapt to new objects.

Finally, the variance in equation 4 is based on a heuristic. Instead, the variance of the function $\hat{f}$ around the mean at a query location could be empirically estimated and used to generate minimum-risk actions for achieving the next state in an approach similar to [Christiansen and Goldberg, 1990].

V. CONCLUSION

We have demonstrated an approach for learning to push unknown objects to arbitrary locations in an image space by observing the effects of past actions on the configuration of the object. By looking directly at the empirically observed effects of actions in the image-space the approach avoids the difficult problem of estimating and modelling the mass and frictional properties of the object being pushed, as well as the camera parameters. The necessary sample size for task success is decreased by using a biased random experimentation approach which enables the agent to avoid repeating failures and instead focus on actions which are promising.

REFERENCES

[Aboaf *et al.*, 1989] E. W. Aboaf, C. G. Atkeson, and D. J. Reinkensmeyer. Task-level robot learning. In *Proceedings, 1988 IEEE International Conference on Robotics and Automation*, pages 1290–1295, Scottsdale, AZ, IEEE Press, 1989.

[Bajcsy *et al.*, 1991] Ruzena Bajcsy, Richard Paul, Xiaoping Yun, and Vijay Kumar. A multiagent system for intelligent material handling. In *Proceedings of the International Conference on Advanced Robotics*, pages 18–23, Pisa, Italy, 1991.

[Campos and Bajcsy, 1990] M. Campos and R. K. Bajcsy. *A robotic haptic system architecture*. Technical Report MS-CIS-90-51, University of Pennsylvania, Dept. of Computer and Information Science, Philadelphia, Pa., 1990.

[Christiansen and Goldberg, 1990] A. D. Christiansen and K. Y. Goldberg. Robotic manipulation planning with stochastic actions. In *Proceedings of DARPA Workshop on Innovative Approaches to Planning Scheduling and Control*, Morgan-Kauffman, Los Altos, CA, November 1990.

[Duda and Hart, 1973] P.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley and Sons, NY, NY, 1973.

[Friedman *et al.*, 1977] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3:209–226, 1977.

[Horn and Schunk, 1981] B.P. Horn and B.G. Schunk. Determining optical flow. *Artificial Intelligence*, 17, 1981.

[Huber, 1981] P.J. Huber. *Robust Statistics*. J. Wiley, New York, 1981.

[Lloyd, 1986] John Lloyd. *Implementation of a Robot Control Development Environment*. Master's thesis, McGill University, Montreal, Canada, 1986.

[Lynch, 1993] K. Lynch. Estimating the friction parameters of pushed objects. In *Proc. of the 1993 IEEE/RSJ Int'l Conference on Intelligent Robots and Systems*, pages 186–193, Yokohama, Japan, 1993.

[Mason *et al.*, 1989] M. T. Mason, A. D. Christiansen, and T. M. Mitchell. Experiments in robot learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 141–145, Ithaca, NY, Morgan-Kauffman, 1989.

[Mason, 1986] M. T. Mason. Mechanics and planning of manipulator pushing operations. *International Journal of Robotics Research*, 5(3):53–71, 1986.

[Mel, 1991] B. Mel. *Connectionist robot motion planning: A neurally inspired approach to visually guided reaching*. Academic Press, San Diego, CA, 1991.

[Moore, 1990] A. W. Moore. Acquisition of dynamic control knowledge for a robotic manipulator. In *Proceedings of the seventh international conference on machine learning*, 1990.

[Salganicoff, 1993] M. Salganicoff. Density-adaptive learning and forgetting. In *Proceedings of the Tenth International Workshop on Machine Learning*, Amherst, MA, Morgan-Kauffman, June 1993.

[Salganicoff *et al.*, 1993] M. Salganicoff, G. Metta, A. Oddera, and G. Sandini. A direct approach to vision guided manipulation. In *Proceedings of the 1993 International Conference on Advanced Robotics*, Tokyo, Japan, November 1993. to appear.

[Stone, 1977] C.F. Stone. Cross-validation: a review. *Math. Operationforsch. Statist. Ser. Statist.*, (9):127–139, 1977.

[Zrimec, 1990] T. Zrimec. *Towards Autonomous Learning of a Behavior by a Robot*. PhD thesis, Dept. of Electrical Engineering and Computer Science, University of Ljubljana, Ljubljana, Slovenia, 1990.