The Fifth Yale Workshop on
Applications of Adaptive Systems Theory
Center For Systems Science
Yale University, May, 1987

SELF-TUNING OF ROBOT PROGRAM PARAMETERS

Lee E. Weiss David A. Simon Arthur C. Sanderson

Department of Electrical and Computer Engineering and The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania

A robot and its interaction with the task environment can be viewed as a physical process to be controlled. In this formulation, the robot program can be viewed as a task level controller. The program, which is synthesized from sets of parameterized motion control primitives, embeds control strategies to implement particular tasks. The automatic development of robot program code using CAD techniques or the manual development by experienced programmers is a difficult process which is often complicated by uncertain, incomplete, and varying models of the task environment. In this paper we address the strategy and parameter selection problem by describing an approach to self-tuning of robot program parameters. In this approach, the robot program incorporates motion control primitives with adjustable parameters and an associated cost function. A hybrid gradient-based and direct search algorithm uses experimentally measured performance data to adjust the parameters to seek optimal performance and track system variations. Alternative control strategies which have first been optimized with the same cost function can then be assessed in terms of their optimized behavior.

1 Introduction

The development of robot program code requires the transformation of abstract robotic task descriptions into desired robot motions. This transformation is a difficult problem which is often complicated by uncertain, incomplete and varying models of the task environment including the robot, manipulated objects, and sensors. The transformation process includes the selection of a control strategy to implement each task. In conventional programming environments the strategy is encoded in the robot program as logically connected sets of motion control primitives. Each primitive has an associated parameter list. For example, a representative primitive **MOVES** $(\vec{X}, V)$ tells the robot to move to position $\vec{X}$ with velocity V along a straight line trajectory. Primitives can also incorporate sensory feedback to accommodate uncertain and changing environments. Force sensing is often used in robotic assembly applications to accommodate contact motion constraints [12]. For example, a simple force monitored "guarded motion" primitive, **MOVES** $(\vec{X}, V, \vec{F}_{thresh})$ terminates the motion if force thresholds are exceeded. Primitives which explicitly specify dynamic sensory feedback strategies such as active stiffness control are also feasible [3].

While simple strategies may consist of a single program control command or motion primitive, more complex strategies are formed by logically sequencing multiple primitive commands. For example, consider the peg-in-hole mating task. If the peg and the hole are not chamfered and the tolerance between them is tight, then one possible multi-primitive strategy is as follows. First, tilt the peg slightly to increase the range of relative positions where initial entry in the hole is guaranteed [5]. Second, move the peg along the axis of the hole with a force monitored motion. The monitor tests lateral and axial forces. If lateral forces are exceeded, realign the peg in the hole using a force feedback primitive to minimize lateral forces, then go to the second step. If axial forces are exceeded then terminate the motion. In contrast, if the peg and hole are chamfered, and the end-effector has sufficient passive compliance (e.g., from an RCC wrist) then simpler single move primitive strategies may be appropriate. A current trend in product design for automated robotic assembly is to incorporate parts geometries which can be *reliably* mated with such simple strategies, thus simplifying the programming requirements.

Automatic synthesis of motion control strategies by CAD systems, and selection of motion primitive parameters has proven to be a very difficult problem due to complex and incomplete models of the system [2,6]. In practice, robot programming has been left to experienced programmers who typically rely on intuition and trial-and-error experimentation to select a strategy and manually tune the program parameters. A good programmer will have abstract notions of optimization in mind as basis for designing the program. For example, in assembly tasks, the programmer seeks a strategy and a set of parameter values which accomplishes the task quickly, while attaining nominal mating forces, and minimizing the probability of mating failures. Humans, however, are not very efficient at searching parameter and symbolic spaces for optimal solutions. Manual searching is tedious and typically the resulting performance could be improved. The process of parameter tuning by a human programmer, or automated parameter selection by a CAD system, is further complicated by the fact that real robotic systems drift with time due to variations in the robot and environment, and thus require periodic parameter readjustment.

In this paper we address one aspect of the strategy and parameter selection problem by describing an approach to self-tuning of robot program parameters. In this approach, the robot program incorporates motion control primitives with *adjustable* bounded value parameters and an associated *cost function*. Search algorithms, which use experimental performance measure evaluation, and which do not rely on explicit robot and environment models, adjust the parameters to seek optimal performance and to track system variations. Alternative control strategies, which have first been optimized *with the same cost function*, can then be assessed in terms of their optimized behavior. In this paper, we discuss this approach for force monitored primitives applied to parts mating tasks.

2 Self-Tuning Formulation

While the self-tuning approach has been widely studied at the servo level to optimize control system tracking performance [10], only recently have researchers explored self-tuning ap-

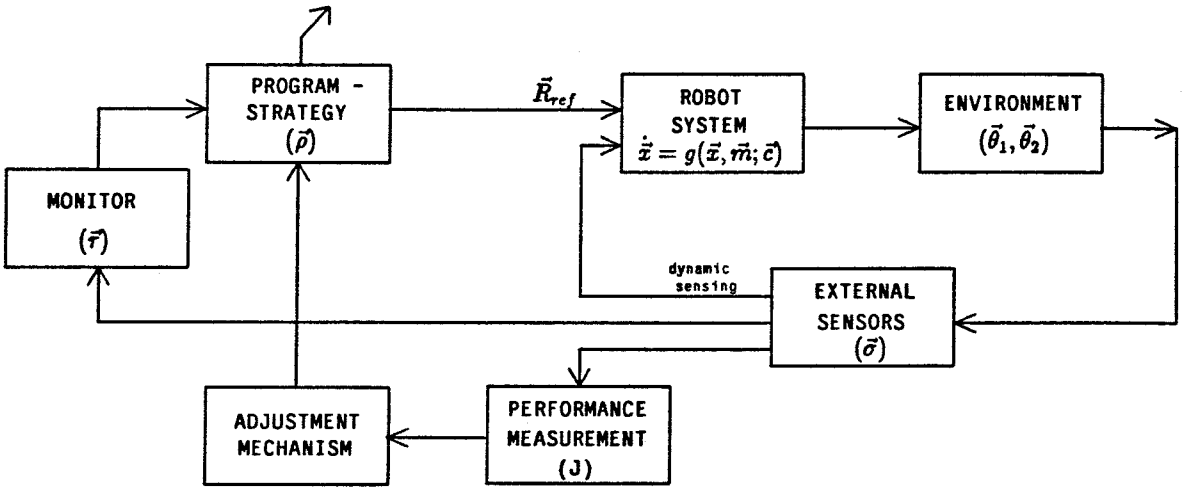

Figure 1: Self-Tuning Approach

proaches at the program or strategy level. These approaches, including the system described in this paper, seek to optimize primitive parameters at the program level. This formulation may facilitate both manual and CAD programming techniques which currently use such primitives as their fundamental building blocks. Simons et al. [9] demonstrate the application of stochastic automata to tune the positioning parameters of a quasi-static force feedback strategy. While the formulation of their approach is similar to ours, stochastic automata are best suited to parameter spaces which have a small number of discrete values. Whitney [11] applies optimal Kalman filter theory to adjust the position parameter values for fine-motion control, and suggests varying the velocity as a function of confidence in the position estimates. This approach however requires explicit position estimation and does not generalize this tuning approach to other parameter values.

In the self-tuning approach described here and illustrated in figure 1 we view the robot/task environment as a physical process to be controlled. The underlying robot dynamics may be described by:

$$\dot{\vec{x}} = h(\vec{x}, \vec{m}) \tag{1}$$

where $\vec{x}$ is the state variable vector and $\vec{m}$ is the vector of kinematic model parameters. Closed-loop control is achieved by a set of position feedback control parameters, $\vec{c}$:

$$\dot{\vec{x}} = g(\vec{x}, \vec{m}; \vec{c}) \tag{2}$$

which responds to some preplanned reference trajectory. Such a system description is sufficient for simple robot positioning tasks. However, for many tasks which involve interaction with the environment, the closed-loop dynamics of the robot/task environment involves other parameters:

$$\dot{\vec{x}} = f(\vec{x}, \vec{m}; \vec{c}, \vec{\theta}_1, \vec{\theta}_2, \vec{\sigma}, \vec{\tau}, \vec{\rho}) \tag{3}$$

where

$\vec{\theta}_1$ = geometric constraint parameters,

$\vec{\theta}_2$ = force constraint parameters,

$\vec{\sigma}$ = sensor sampling parameters,

$\vec{\tau}$ = computational delay parameters,

$\vec{\rho}$ = task-level control parameters,

and the reference inputs, $\vec{R}_{ref}$, are expressed as position and force trajectories. Execution of the robot program requires specification of $\vec{\rho}$, and resulting performance of the system depends on $\vec{\rho}$.

Design of the control strategy for these cases is difficult due to the uncertainty of the task parameters ($\vec{\theta}_1, \vec{\theta}_2, \vec{\sigma}, \vec{\tau}$), and the complexity in accurately modeling them. This uncertainty occurs as trial-to-trial variations and slowly varying changes with time. As a result, performance of the system for a given task may vary for a given implementation, between trials, as a function of time. For a given task-specific performance measure

$$J(\vec{x}; \vec{\rho}) \tag{4}$$

we would like to achieve

$$\min_{\vec{\rho}} J(\vec{x}; \vec{\rho}) \tag{5}$$

for best average performance and tracking of performance over time. For example, for a parts mating task the cost function may have the form :

$$J(\vec{x}; \vec{\rho}) = E\left[\frac{1}{(1-P_e)}\left(k_1 \Delta T + K_2^T(\vec{F}_{ref} - \vec{F}_{ss})^2 + K_3^T\,[max((\vec{F}_{peak} - \vec{F}_{ref}), 0)]^2\right)\right] \tag{6}$$

where ΔT is task cycle time, $\vec{F}_{ref}$, $\vec{F}_{ss}$, and $\vec{F}_{peak}$ are vectors of the *desired* or *reference* steady-state forces, the *measured* steady-state forces, and the peak overshoot forces respectively, P_e is the probability of a mating failure, and k_1, K_2, K_3 are constant scalars or vectors which weight the relative importance of each performance component. This process control description of the system suggests the use of optimization techniques to tune the program parameters. In this paper we will describe an implementation of a system which combines gradient-based and direct search techniques to accomplish this. While the robot program code used in our experiments have been manually derived, these optimization techniques could be used to tune the parameter sets of CAD generated code.

3 Search Algorithms

A design goal for a self-tuning system is to achieve stable

response with fast convergence properties. The attainment of these goals is especially difficult if the performance space is characterized by multi-modal behavior. As will be shown in the next section, the performance space of monitor-based control primitives is often characterized by a complex multi-modal function which causes simple gradient based optimization methods to fail. In our approach we use a hybrid gradient-based and direct search algorithm to achieve stable response. The tuner was also designed to converge in a reasonable number of iterations. As suggested above, the performance measure is evaluated experimentally by the robot at various points in parameter space, which can be a time consuming process. Thus, search techniques which do not require a large number of performance measure evaluations are preferred for this application.

Driven by the above requirements, a two-stage self-tuning process has been designed. The goal of the first stage is to obtain a rough estimate of the optimal parameter set using a relatively small number of experimental trials. This coarse resolution adjustment is followed by a second stage fine resolution tuning to locate a true global minimum.

In the first stage, least squares regression analysis is used to fit experimentally collected performance data to a simple analytic model of the performance surface. This model, which is described in the next section, does not incorporate the complex multi-modal component of the actual surface, but only the underlying "low-frequency" trends. The resolution of adjacent parameter test values is set to minimize the total number of experiments required, while maintaining a good least-squares fit to the low frequency component of the actual performance space. This coarse resolution has been selected by experimentally evaluating the improvement in mean-squared error versus resolution over a large number of optimization trials. The minimum of the resulting analytic model is then found using a "quasi-Newton" optimization algorithm known as Successive Quadratic Programming (SQP) [1]. This algorithm was chosen because it deals well with simple constraints on the independent variables which arise from the constrained parameter space.

The second stage of the self-tuner uses the minimum found in the first stage as the starting point for a "fine-tuning" high resolution direct search. Various approaches have been evaluated, each of which uses a small window in parameter space centered around the current minimum estimate. In this region, we have found that the amplitude of the multi-modal periodic component is often relatively small. Unfortunately, efficient approaches such as Hooke-Geeves [4] pattern search and gradient-based algorithms resulted in unpredictable and often unstable behavior. Such behavior is inevitable when a complex multi-modality is not explicitly accounted for in the model, even in a region where its amplitude is relatively small. Thus, to keep our approach simple we used a less efficient non-patterned direct search which samples all the points within the current window. The direct search selects the smallest performance value within the window and sets this point to the new minimum. This process of defining a window about the current minimum, collecting a set of experimental performance data within the window, and selecting the best operating point, continues until the performance change from one iteration to the next is below a specified threshold. Examples of this self-tuning method are presented in the following sections.

Once the self-tuner has found an optimal operating point for a given task, the robot can be put into "operational" mode. In this mode, shifts is the optimal operating point can be caused by tool wear, part tolerance changes, and drift within the robot. Therefore, it is desirable to be able to track small performance changes while the robot is operating. While we have not as yet experimentally evaluated a tracker, fine-tuning techniques could be applied *over a relatively small window* if it can be assumed that these variations are slowly time-varying.

In order to study the self-tuning approach on actual robot tasks, an experimental test-bed has been set up. It consists of an IBM 7565 robot with AML programming/control environment, a robot gripper with adjustable compliance along the tool Z axis, and a force sensor capable of measuring forces along the tool Z axis. Additional processors were interfaced to the AML robot system including an IBM-PC and a SUN-2 workstation. The IBM-PC was used to implement several complex monitoring strategies which were not available in AML, while the search algorithms were implemented on the SUN. Additional hardware was also built to perform peak force detection which was also not available in AML.

4 Force Monitor Tasks

The application of a force monitor primitive is illustrated by the task of affixing two parts, one of which has an adhesive surface. More specifically, the task is to bring the parts into contact as quickly as possible, while achieving a final steady-state force of F_{ref}. To understand this task, the robot tool-tip was programmed to contact a rigid surface using the force monitored motion primitive command, **MOVE** $(\vec{X}, V, F_{thresh})$. This primitive instructs the robot joint level controllers to move to position $\vec{X}$ with velocity V, but stop if the measured force in the direction of motion exceeds F_{thresh}. The actual force achieved is a function of V, F_{thresh}, and the sensor monitor sampling period. For this task, the performance measure is:

$$J = k_1 \Delta T + k_2 (F_{ref} - F_{ss})^2 \qquad (7)$$

Note that J only contains cycle-time and steady-state force error components, since overshoot was not present and "mating failures" are not relevant. Expected values were not used in equation (7) because we observed that the performance at any point was highly repeatable for this simple experiment.

The goal of the self-tuning algorithm is to vary V and F_{thresh} to minimize J. A plot of experimentally measured J vs. V and F_{thresh} is shown in figure 2. For this plot, the coefficients k_1 and k_2 have been adjusted so that the maximum value of the cycle time component is approximately the same as the maximum value of the steady-state force error component. While this particular coefficient setting is arbitrary, in general, specified limits to cycle time and forces should be considered. For example, if the cycle time at the optimal operating point is larger than constraints specified by assembly queuing requirements, then the cycle time coefficient should be increased. Conversely, if the parts being assembled are extremely fragile, then more weight should be assigned to the force components.

The multi-modal surface characteristic is clearly seen in figure 2. This characteristic results from the finite sampling interval (20 msec) at which the monitor trip conditions are tested. Since a trip condition may be *satisfied* at any time within the interval, there can be a delay between the time at which the trip condition is physically satisfied, and the time at which the monitor system acknowledges the condition. This delay causes a complex, non-linear periodic function in performance space. We have developed an analytic model of this monitoring task to verify our experimentally measured results. This model is beyond the scope of this paper, however details can be found

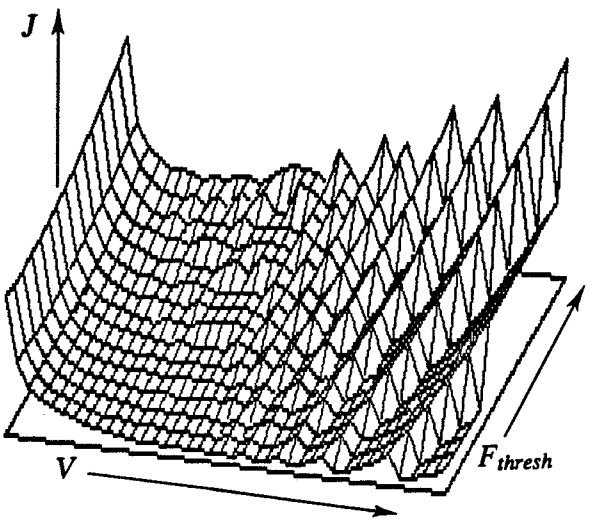

Figure 2: Performance Map for the Adhesive Task

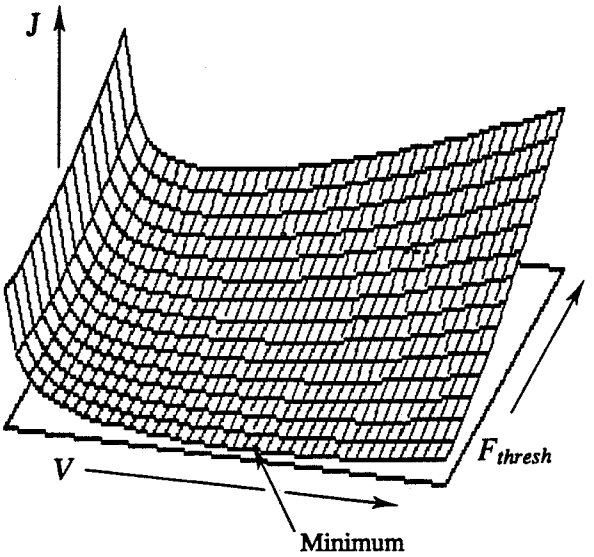

Figure 3: Surface Model for the Adhesive Task

in [8]. This multi-modal behavior is typical of robots using sensory monitored motions and is thus an important case study. For example, we have observed similar multi-modal behavior on a PUMA 560 robot running the VAL-II control system.

To optimize the force monitoring task, the self-tuning algorithms of Section 3 are applied. The coarse resolution tuner fits the experimentally collected performance data shown in figure 2 to the function S given by:

$$S = \vec{\beta} \cdot [1,\ V,\ F_{thresh},\ VF_{thresh},\ V^2,\ F_{thresh}^2,\ 1/V,\ F_{thresh}/V]^T \qquad (8)$$

The vector of coefficients, $\vec{\beta}$, are estimated using the method of least-squares to minimize the error function

$$E = \sum_{i=0}^{n} [J(\vec{\rho}_i) - S_{\vec{\beta}}(\vec{\rho}_i)] \qquad (9)$$

where $\vec{\rho}_{0\ldots n}$ are the parameter values, V and F_{thresh}, at each of the collected data points, and J is the actual measured performance. The resulting surface model is shown in figure 3. The minimum of the surface found by the SQP Algorithm is marked in the figure.

The form of the surface model, S, is based on an understanding of the underlying physical processes of the task [8]. No explicit models of the robot were used to derive it. The first six terms of this surface form a quadratic in V and F_{thresh}, and are used to model the steady state force error component of the surface. The remaining two terms are inversely dependent on V, and were added specifically to model the cycle time component of the performance measure. It should be noted that the algorithms function properly without the final two terms, however an improvement in overall convergence time is realized if these terms are included.

The operating point found during the coarse resolution tuning stage is used as the starting point for the direct search fine-tuner. The window size used in the direct search was set so that at least two full cycles of the periodic variation are included within a window at all points in the search space. This ensures that the search does not prematurely terminate in a valley of the periodic variation. The sample spacing used in the direct search was set based upon the desired resolution of the search, and upon several performance sensitivity experiments.

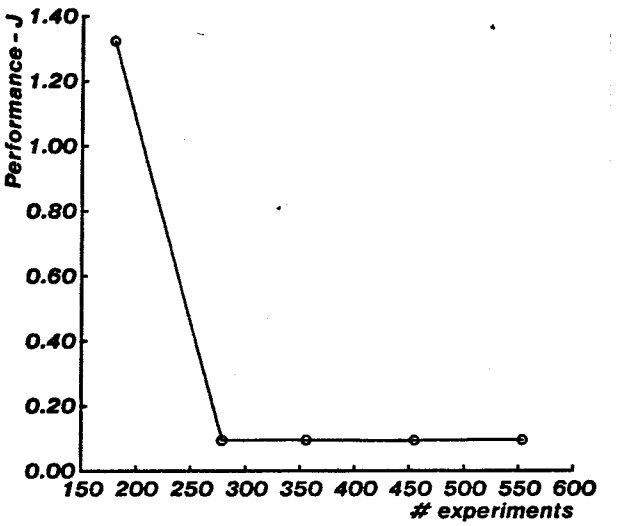

Figure 4: Performance at the Completion of Self-Tuning Cycles

Figure 4 shows the performance measure value at the end of each of the self-tuning cycles. The leftmost data point corresponds to the output of the coarse tuner, while the remaining data points correspond to the output of the fine-tuner. As seen in the figure, the minimum is found with only one iteration of the fine-tuner. Within the resolution of the fine tuner, this minimum is the "true" global minimum as verified by an independent high resolution direct search of the entire parameter space. The behavior of this self-tuner has been studied over a variety of performance measurement gains, reference forces and tool compliances. Typically, the minimum is found within one or two iterations of the fine-tuner, which suggests the appropriateness of the simple model fit used in the coarse tuner.

In this section we demonstrated the self-tuning approach for a simple yet practical force monitoring strategy. Next, we demonstrate how this approach can be applied to facilitate selection of the strategy itself.

5 A Snap-Fit Task Example

We are currently studying the self-tuning approach on a variety of snap-fit tasks. Reliable snap-fit operations are a major requirement for mating of parts which have been designed for automated assembly. There are a number of feasible strategies to accomplish such operations. In our approach, the strategy selection process is based on comparing the performance of strategies which have first been optimized *with the same cost function*. The optimized performance measure provides a unifying metric by which alternative strategies can be compared. We are also exploring if there is a single generic strategy, amongst the alternative strategies, which is optimum over a broad range of "designed for assembly" snap-fit operations. The existence of a generic strategy would facilitate the design of off-line CAD programming systems. The design for assembly constraint is specified because it is well known that for more general assembly and manipulation tasks the concept of a generic strategy is not feasible. Even small variations in parts geometry will change the appropriate strategy [7]. Our research demonstrates, however, that even for fixed geometric constraints, the optimum strategy is a function of the performance requirements. This suggests that generic strategies would be difficult to specify, even for designed for assembly operations.

The "snap-ball" task illustrated in figure 5, provides a good model of a generic snap-fit task, and was used a case study in our work. The task is to insert the ball into the socket as fast as possible, such that the resulting final force is F_{ref}, while ensuring that overshoot forces remain small. Figure 6 shows a typical plot of insertion force vs. time for the snap-ball task. As seen in the plot, there is a sharp decrease in insertion force as the ball enters the socket. This negative change in force is one characteristic of snap-fit operations.

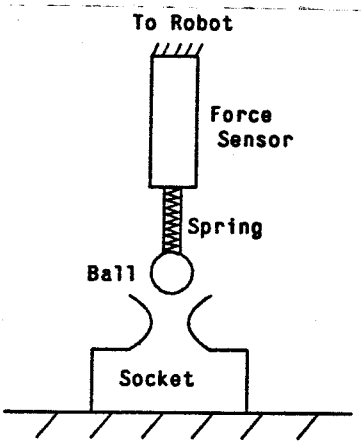

Figure 5: Snap Ball Task

To illustrate the strategy selection problem, two alternative guarded motion strategies are described. The first strategy uses the aforementioned force monitor motion primitive, **MOVE** $(\vec{X}, V, F_{thresh})$, with adjustable parameters V and F_{thresh}. For this strategy, the monitor trips in the region before the maximum pre-insertion force (MPIF) is reached and relies on the robot's momentum to complete the insertion. The second strategy uses the IBM-PC to perform a monitor test not available in AML. This strategy uses the primitive, **MOVE** $(\vec{X}, V, df/dt_{thresh})$, with adjustable parameters V and df/dt_{thresh}, where df/dt_{thresh} is a threshold on the rate of change of force. For this strategy, we limit df/dt_{thresh} to negative values so that the monitor trips after the MPIF, and insertion is assured.

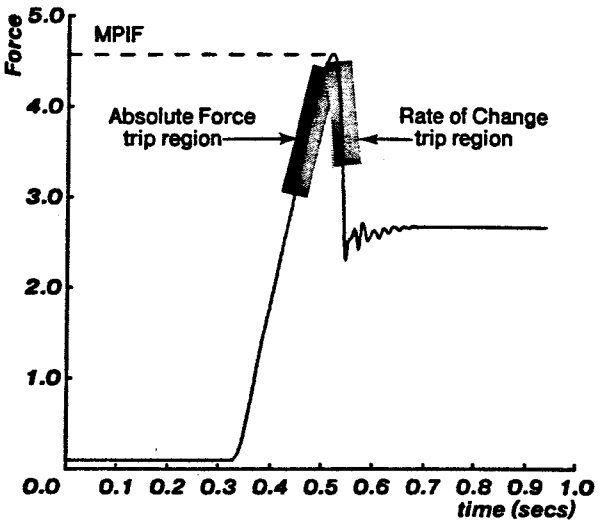

Figure 6: Insertion Force vs. Time for Snap-Ball Task

Self-tuning of the snap-ball task must account both for mating failures and for the significant stochastic variation which is observed for this task. However, since only a "rough" estimate of the minimum is desired for the *coarse* tuner, the stochastic variation is not taken into account in this stage. For the coarse tuner, J is evaluated once at each point in parameter space according to:

$$J(\rho) = k_1 \Delta T + k_2 (F_{ref} - F_{ss})^2 + k_3 [max((F_{peak} - F_{ref}), 0)]^2 \quad (10)$$

Mating failures, which occur when the ball does not enter the socket, are accounted for by incorporating success/failure constraint boundaries. The least-squares fit is performed only over the *successful* data points in order to generate the surface model. The resulting surface is optimized by an augmented SQP algorithm [8], which finds minima within the success region or on a region boundary. The fine tuner then uses equation (6) to calculate the expected performance by averaging J over successive trials, where P_e is estimated as the ratio of failures to the total number of trials.

In order to study the effects of performance measurement weighting on the selected strategy, the snap-ball task was tuned with two different performance measures. The first performance measure had a large value of F_{ref}, and the performance components were weighted such that their maximum values were equal. This measure would be appropriate when parts can tolerate large mating forces and cycle time is important. For this case, optimization of the absolute force strategy yielded the best performance. This is not surprising primarily because motion is terminated earlier than for the rate of change strategy. Therefore the optimum velocity for the absolute force strategy can be larger.

For the next case, the performance measure was assigned a smaller value of F_{ref}, a coefficient of zero on cycle time, and equally weighted overshoot and steady-state components. This performance measure would be appropriate for mating fragile parts. For either strategy to attain low steady-state and overshoot forces, the parameters must be set such that the robot speed is slow as it goes through the MPIF point. For this performance measure, the rate of change strategy exhibited superior performance primarily because mating successes are guaranteed even at low speeds. However, for the absolute force strategy the robot's lower momentum may be insufficient to overcome friction and complete the insertion. For this strategy, the stochastic nature of the system made it difficult to

find a parameter set for which consistently successful insertions occur at lower speeds. Thus, the P_e component dominates and degrades the performance of the absolute force strategy.

Other strategies may exist which are optimal for both performance measures. However, this simple example suggests that robust strategies which are optimal over a broad range of tasks and performance requirements may be difficult to find. While, careful analysis of these requirements can narrow the range of potential strategies, on-line evaluation of alternative approaches may ultimately be required to select the best one.

6 Conclusion

The self-tuning formulation provides a useful method for automatic adjustment of motion primitive parameters in robot programs. Self-tuning at the *program level* is appropriate for both manual and automatic programming techniques which use these primitives as their fundamental building blocks. However, even simple monitor based strategies are difficult to optimize due to the complex multi-modal behavior which they exhibit. Dynamic control strategies which can accommodate force constraints, such as impedance control [3], may not exhibit this multi-modal behavior. Such control schemes are worth exploring since smoother performance spaces could be optimized using more efficient search techniques.

We also show that optimization of alternative strategies using the same cost function provides a basis for strategy selection. While it is well known that variations in parts geometry change the preferred strategy [7], we have demonstrated that variations in desired performance lead to changes in the optimal strategy as well. This fact will complicate the design of off-line CAD programming systems even for parts which have been designed for automated assembly. Rather than rely on a simple menu of generic strategies, the CAD systems will require more complex representations of tasks and desired performance specifications.

Further work needs to be done in several areas. A tracking system based on the self-tuning approach needs to be implemented and evaluated. Alternative control strategies, particularly impedance control, need to be evaluated to determine if smoother performance surfaces exist for which more efficient search techniques can be applied. The self-tuning approach should be evaluated on other basic design for assembly tasks such as "twist", "push", "screw", etc. to determine the feasibility of extending this approach.

References

[1] Biegler, L.T. and Cuthrell, L.E. "Improved Infeasible Optimization for Sequential Modular Simulators - II: The Optimization Algorithm" *Computers and Chemical Engineering*, Vol. 9, No. 3, pp 257-267, 1985

[2] Dufay, Bruno, and Latombe, Jean-Claude "An Approach to Automatic Robot Programming Based on Inductive Learning", *International Journal of Robotics Research*, Vol. 3, No. 4, Winter 1984

[3] Hogan, Neville "Stable Execution of Contact Tasks Using Impedance Control" *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 2, pp 1047-1054, April, 1987

[4] Hooke, Robert and Geeves, T.A. "Direct Search: Solution of Numerical and Statistical Problems", *Journal of the Association for Computing Machinery*, Vol 8, pp 212-229, April 1961

[5] Inoue "Force feedback in precise assembly tasks", August 1974, AIM-308. Cambridge, Mass.: Massachusetts Institute of Technology Artificial Intelligence Laboratory. Reprinted in Winston, P.H. and Brown, R.H. eds. 1979. *Artificial Intelligence: An MIT Perspective*, MIT Press

[6] Lieberman, L., and Wesley, M. "Autopass: An Automatic Programming System for Computer Controlled Mechanical Assembly" *IBM Journal of Research and Development* Vol 21.

[7] Lozano-Perez, T., Mason, M.T., and Taylor, R.H. "Automatic Synthesis of Fine-Motion Strategies for Robots" *International Journal of Robotics Research*, Vol. 3, No. 1, Spring 1984

[8] Simon, D. A. *Masters thesis, Carnegie-Mellon University*, May 1987

[9] Simons, J., Van Brussel, H., and De Schutter J. "A Self-Learning Automaton with Variable Resolution for High Precision Assembly by Industrial Robots." *IEEE Transactions on Automatic Control*, Vol. Ac-27, No. 5., October 1982

[10] Weiss, Lee E. "Dynamic Visual Servo Control of Robots: An Adaptive Image-Based Approach", *PhD thesis, Carnegie-Mellon University*, April 1984

[11] Whitney, D.E. and Junkel, E.F. "Applying Stochastic Control Theory To Robot Sensing, Teaching, and Long Term Control." *12th International Symposium on Industrial Robots*, June 1982, Paris

[12] Whitney, D.E. "Historical Perspective and State of The Art in Robot Force Control." *1985 IEEE International Conference on Robotics and Automation*, St. Louis, Missouri