Machine Learning, 23, 121–161 (1996)

Real-World Robotics: Learning to Plan for Robust Execution

SCOTT W. BENNETT bennett@sra.com
Systems Research and Applications Corporation, 2000 15th Street North, Arlington, VA 22201

GERALD F. DEJONG dejong@cs.uiuc.edu
Beckman Institute, University of Illinois, 405 North Mathews Avenue, Urbana, IL 61801

Editors: Judy A. Franklin, Tom M. Mitchell, and Sebastian Thrun

Abstract. In executing classical plans in the real world, small discrepancies between a planner's internal representations and the real world are unavoidable. These can conspire to cause real-world failures even though the planner is sound and, therefore, "proves" that a sequence of actions achieves the goal. Permissive planning, a machine learning extension to classical planning, is one response to this difficulty. This paper describes the permissive planning approach and presents GRASPER, a permissive planning robotic system that learns to robustly pick up novel objects.

Keywords: machine learning, robotics, uncertainty, planning

1. Introduction

For some time technical difficulties have prevented the field of planning from delivering on its early promise. Increasingly it has become accepted that domain-independent classical planning, in which one finds a sequence of actions which provably achieves a goal when applied to a problem's initial state is at best quixotic. The death knell for the approach was sounded by Chapman (Chapman, 1987) who showed that with some assumptions (which are rather reasonable in real-world domains), the general classical planning problem is prohibitively difficult. One new approach, known generally as *reactivity*, has made great strides to circumvent the obstacles of classical planning (Agre & Chapman, 1987, Firby, 1987, Schoppers, 1987, Suchman, 1987). Classical planning demands that the planner know and model the effects of its actions. For any sequence of actions a classical planner must be able to judge accurately their cumulative changes to a state. This ability to project states through operators may seem modest at first but the failure of classical planning can be viewed as traceable to this task. A central tenet of pure reactive activity is to do no projection. The "planner" (or better the "agent") makes no attempt to anticipate how the world will look after an action is executed. Instead, an action is selected based entirely upon the agent's sensor-supplied knowledge of the current world state. After the execution of an action, the world changes in some (possibly complex) way. The next action is selected in the same manner, based upon the updated sensor values. In a way, a purely reactive system employs the world itself to model the effects of actions at the expense of *gedanken* backtracking. The method of selecting actions, and the demands such a mechanism places upon its sensors, are the subject of much current research

(Agre & Chapman, 1987, Hammond, Converse, Marks, 1990, Brooks, 1987). In any case, the resulting system relies on its sensor abilities to drive action selection.

Models of reactivity, for all their recent popularity, are only one approach to the problems faced by classical planning. In this paper we introduce another. In some ways it is the dual of the reactive approach. Our approach, called *permissive* planning, like the reactive approach, gives up the notion of a provably correct plan. However, the concept of projection remains. Indeed, it is, if anything, more central than before.

In most real-world domains it is impossible to describe the world correctly and completely. It follows that internal system representations of the world must, at best, be approximate. Such approximate representations of world attributes are called *data approximations*. They may arise from imperfect sensors, or incomplete inferencing. We introduce the concept of *permissiveness* of a plan as a measure of how faithfully the plan's preconditions must reflect the real world in order for the plan to accomplish its goals. One plan is more *permissive* than another if its representations can be more approximate while continuing to adequately achieve its goals. We do not propose to quantify this notion of permissiveness. Instead, we employ a machine learning approach which enhances permissiveness of acquired planning concepts.

The approach involves acquiring and refining generalized plan schemata which achieve often-occurring general goals and sub-goals. Acquisition is through rather standard explanation-based learning (Serge, 1988, Mitchell, Keller & Kedar-Cabelli, 1986, DeJong & Mooney, 1986, Mitchell, Mahadevan & Steinberg, 1985). However, the refinement process is unique.

To drive refinement, the system constantly monitors its sensors during plan execution. When sensor readings fall outside of anticipated bounds, execution ceases and the plan is judged to have failed. The failure must be due to a data approximation; if there were no mismatch between internal representations and the real world, the plan would have the classical planning property of provable correctness. The plan's failure is diagnosed. Ideally, only a small subset of the system's data approximations could underlie the monitored observations. The system conjectures which of its internal expressions might account for the observations. Next, the system uses qualitative knowledge of the plan's constituent operators. The small conjectured error is symbolically propagated through the plan to plan parameters. The plan parameters are adjusted so as to make the planning schema less sensitive to the diagnosed discrepancy with the world. If the process is successful, the refined schema is uniformly more permissive than the original, which it replaces. Thus, through interactions with the world, the system's library of planning schemata becomes increasingly permissive, reflecting a tolerance of the particular discrepancies that the training problems illustrate. This, in turn, results in a more reliable projection process. Notice that there is no improvement of the projection process at the level of individual operators. Performance improvement comes at the level of plan schemata whose parameters are adjusted to make them more tolerant of real-world uncertainties in conceptually similar future problems. Adjustment is neither purely analytical nor purely empirical. Improvement is achieved through an interaction between qualitative background knowledge and empirical evidence derived from the particular real-world problems encountered.

The notion of permissive planning is not tied to any particular domain. It is a domain-independent notion that is, nonetheless, not universally applicable. There are characteristics of domains, and problem distributions within domains, that indicate or counter-indicate the use of permissive planning. Later, the requirements of permissive planning will be made more formal. Here we give an intuitive account of characteristics needed to support permissive planning. An application that does not respect these characteristics is unlikely to benefit from permissive planning.

For permissive planning to help, internal representations must be reasonable approximations to the world. By this we mean that there must be some metric for representational faithfulness, and that along this metric, large deviations of the world from the system's internal representations are less likely than small deviations.

Second, some planning choices must be subject to continuous real-valued constraints or preferences. These constraints and preferences are called *parameters* of the plan schema. They might be thresholds indicating bounds on acceptable values, or actual real-valued arguments to domain operators. These parameters are tuned to achieve permissiveness.

Finally, the planner must be supplied with information on how each operator's preconditions and arguments *qualitatively* change its effects. This information is used to regress the diagnosed out-of-bounds approximations of failed plans through the planning structure to parameters. Such propagation determines how parameters may be adjusted so as to decrease the likelihood of similar future failures.

Clearly, many domains do not respect these constraints. However, robotic manipulation domains form an important class in which the above characteristics are naturally enforced. Consider the data approximation constraint. A typical expression in a robotics domain may refer to real-world measurements. Object positions and dimensions, for example, require some representation for metric quantities. An example might be something like **(HEIGHT-IN-INCHES BLOCK3 2.2)**. Such an expression is naturally interpreted as an approximation to the world. Indeed, expressions such as this one are useless under a standard semantics. The conditions of truth require that the height of the world object denoted by **BLOCK3** be exactly 2.2 inches. Technically, no deviation whatsoever is permitted. If the height of **BLOCK3** is off by only 10^{-40} inches, the expression is false – just as false as if it were off by 5 inches or 50 inches. Clearly, such an interpretation cannot be tolerated; the required accuracy is beyond the numerical representational capabilities of most computers. Another nail is driven into the coffin for standard semantics by real-world constraints. Actual surfaces are not perfectly smooth. Since the top and bottom of **BLOCK3** most likely vary by more than 10^{-40} inches, the "height" of a real-world object, using a standard semantics, is not even a well-defined relation. In fact, no working system adopts a strict standard semantic interpretation for expressions such as the one above. There are several alternatives which will be discussed later. For now, it is sufficient to notice that expressions such as **(HEIGHT-IN-INCHES BLOCK3 2.2)** are extremely common in robotic domains and can be easily interpreted as satisfying our informal definition of an approximation: the metric for faithfulness is the real-valued height measure, and, presumably, if a reasonable system describes the world using the expression **(HEIGHT-IN-INCHES BLOCK3 2.2)** it is more likely the case that **BLOCK3** is 2.2001 inches high than 7.2 inches high. It is

essential that the expression not saddle the system with the claim that **BLOCK3** is precisely 2.2 inches high.

The second condition for permissive planning requires that continuous real-valued parameters exist in the system's general plans. Geometric considerations in robotic manipulation domains insure that this condition is met. As an example, consider a robot manipulator that wishes to move its arm past **BLOCK3**, which in turn rests on the table. Some path must be adopted for the move. From the geometrical constraints there is a minimum height threshold for the path. Since the arm must not collide with anything (in particular with **BLOCK3**), it must be raised more than 2.2 inches above the table. This height threshold is one of the plan parameters. Any value greater than 2.2 inches would seem to be an adequate bound on the parameter for the specific plan. However, it is easy to see that if 2.2 inches *is* adequate, so is 2.3 inches, or 5.0 inches, etc. Thus, the plan supports the parameter as a continuous real-valued quantity. Notice, that once the specific plan of reaching over **BLOCK3** is generalized by EBL, the resulting plan schema parameterizes the world object **BLOCK3** to some variable, say, *?x* and the value 2.2 to *?y* where (**HEIGHT-IN-INCHES** *?x ?y*) is believed, and the threshold parameter to *?z* where *?z* is equivalent to (+ *?y* ϵ) for the tight bound, or equivalent to (+ *?y* ϵ 0.1), for the second bound above, or equivalent to (+ *?y* ϵ 2.8), for the third bound above. The value of ϵ insures that the bound is not equaled and can be made arbitrarily small in a perfect world. As will become clear, in permissive planning, ϵ may be set identically to zero or left out entirely.

The final condition for permissive planning requires qualitative information specifying how the effects of domain operators relate to their preconditions and arguments.[1] This constraint, too, can be naturally supported in robotic manipulation domains. Consider again the plan of moving the robot arm past **BLOCK3**. The plan involves moving the arm vertically to the height *?z* and then moving horizontally past the obstacle. The required qualitative information is that the height of the robot arm (the effect of **MOVE-VERTICALLY**) increases as its argument increases and decreases as its argument decreases. With this rather simple information the generalized plan schema for moving over an obstacle can be successfully tuned resulting in a more permissive plan schema.

To illustrate, suppose the plan schema for moving past an obstacle is acquired from a successful specific example of moving past **BLOCK3**. To have been successful, reality must not have interfered with the system's approximations. Suppose **BLOCK3** is in reality 2.14962 inches high. The naive specific plan simply moves the arm to the minimum threshold (2.2 inches, the believed height of **BLOCK3**) successfully missing the obstacle by about a twentieth of an inch. Since it was successful, a schema might be generalized for later use. This involves variablizing the constants in standard EGGS (DeJong & Mooney, 1986) or EBG (Mitchell, Keller & Kedar-Cabelli, 1986) fashion. Suppose now that we wish to reach past another obstacle, say **BLOCK7** which we believe is 5.5 inches high, but is in reality 5.6 inches high. The plan fails when an unexpected force is encountered during the horizontal move action. The failure must be diagnosed to a set of likely offending approximations. For the example, we will assume that diagnosis uniquely identifies (**HEIGHT-IN-INCHES BLOCK7 5.5**) as the offending representation, conjecturing that its height is greater than represented. This might be done from the fact that the unexpected force is perpendicular to the direction of motion, and **BLOCK7** is believed to be the only object in the vicinity

of the arm at the time of collision. The result of the diagnosis is a symbolic expression Δ representing that the block is too high for the plan to succeed. Let $\mathbf{P}(\Delta)$ represent the probability that the general obstacle is too high. We do not attempt to assign an actual numerical probability to this expression; it is treated purely symbolically. If it is decided that the system must not tolerate the failure, then $\mathbf{P}(\Delta)$, whatever its value, is greater than the allowable threshold. The generalized plan must be adjusted to reduce $\mathbf{P}(\Delta)$, or the planning concept must be discarded. The next step is to qualitatively regress $\mathbf{P}(\Delta)$ through the plan to parameters that unambiguously, in a qualitative sense, reduce $\mathbf{P}(\Delta)$. If no such parameters can be found, the general planning concept must be discarded. According to the plan, the parameter *?z* above is identified as one whose increase uniformly reduces $\mathbf{P}(\Delta)$. That is, as one moves "excessively" higher over an obstacle of uncertain height, one is less likely to collide with it. If there is no conflicting bound on *?z* it is adjusted upward, and the refined plan replaces the original. Thus, in the future similar planning episodes the robot system will raise its hand higher than it believes necessary when passing over obstacles. The permissiveness of the plan has been increased. Increasing *?z* to decrease $\mathbf{P}(\Delta)$ may increase the probability of some *other* failure not encountered in the refining experience. Managing and exploiting tradeoffs in the context of a particular problem distributions is what permissive planning is all about.

From a different point of view, permissive planning amounts to blaming the plan for execution failures, even when in reality the representations, not the plan, are at fault. This is a novel approach to planning which results in a different, rather strange semantics for the system's representations. Current research includes working out a more formal account of the semantics for representations in permissive plans. Straightforward interpretations of the expressions as probabilistic seem not to be sufficient. Nor are interpretations that view the expressions as fuzzy or as having uncertainty or error bounds. The difficulty lies in an inability to interpret an expression in isolation. An expression "correctly" describes a world if it adequately supports the permissive plans that make use of it. Thus, an expression cannot be interpreted as true or not true of a world without knowing the expression's context including the system's planning schemata, their permissiveness', and the other representations that are believed.

We now return to the popular alternatives to uncertainty in robotics, alternatives to a standard semantics for expressions such as **(HEIGHT-IN-INCHES BLOCK3 2.2)**. Inability to precisely describe the world has always been a difficult problem in robotics domains. There are three common approaches. The first most obvious approach is to select representations that are, in a sense, good enough. This is the traditional AI notion of a micro-world. In the most straightforward version, the system implementor takes on the responsibility for insuring that no problems will result from necessarily imprecise descriptions of the domain. In general, this requires the implementor to characterize in some detail all of the future processing that will be expected of the system. Often he must anticipate all of the planning examples that the system will be asked to solve. If the physical robot system is not up to the accuracy that the examples require, the implementor must build a better vision system or purchase a more precise, more reproducible robot manipulator. This approach has enjoyed frightening popularity. While it is most often used in systems which research phenomena other than uncertainty, the implementors seldom more than tacitly acknowl-

edge the semantic implications. When employed by practical systems the approach results in an ultimately-doomed quest for increasingly exacting (and expensive) hardware. The great irony of industrial automation, where this approach is nearly universal, is that the mechanical positioning capabilities of robots must far exceed the humans that they replace. The characteristic brittleness and inflexibility of industrial robotics is a consequence of the semantically-rooted constraint that the implementor anticipate all future applications

The second approach involves monitoring and/or manipulating explicit representations of uncertainty (Zadeh, 1965, Brooks, 1982, Lozano-Perez, MAson & Taylor, 1984, Erdmann, 1986, Davis, 1986, Hutchinson & Kak, 1990). In a pure and unsimplified form this approach adequately models any knowable theory of uncertainties; it preserves the classical planning ideal of a provably correct plan. Unfortunately, a high computational price is incurred. A general ability to project states including objects with explicit general error bounds is necessarily no less difficult than if the objects are known precisely. Many systems incorporate simplifications, typically assuming that uncertainties are constant, independent of context, or otherwise constrained. This buys some efficiency at the price of generality. But the efficiency is never greater than if the objects had zero uncertainty so that the Chapman intractability results still apply.

Finally, spatial uncertainties in robotics may be dealt with by guaranteeing a conservative approach to inferencing. In a sense, a kind of worst (or sufficiently bad) case representation is assumed for objects. The approach, which seems only used for problems of path planning, includes techniques like quantizing the space (Wong & Fu, 1985, Zhu & Latombe, 1990, Malkin & Addanki, 1990) and imagining a repulsive potential field around obstacles (Khatib, 1986, Hwang, 1988). Interestingly, the approach can be more efficient than the zero uncertainty case. Since object boundaries are not guaranteed to be the tightest possible, they can be selected to be both conservative and simplifying. This benefit does not come for free. In different guises completeness or correctness can be sacrificed. In a sense, this is the closest of the popular approaches to the research reported in this paper. In a sense we also adopt a conservative representation, although the uncertainty tolerance is due to plan characteristics rather than explicit representations. This shift supports a context-sensitive conservatism which supports reasoning about general manipulation problems rather than only path planning in a static world.

2. Learning Increasingly Permissive Plans

This section defines a framework for permissive planning and describes an algorithm which, based on execution failures, incrementally increases plan permissiveness so as to decrease the likelihood of those failures. First, data approximations are defined. They provide the internal representations employed during planning. Because data approximations are approximations to the world, plan execution failures which may result can be linked to some set of these data approximations. Next, we define plan parameters. These parameters may be tuned to increase the permissiveness of the plan. The remainder of this section discusses the plan generation, execution, and refinement process for permissive plans.

2.1. *Data Approximations*

Data approximations are representations for approximate continuously valued data about the state of the world. They can either be *external* or *internal*. External data approximations are used to represent the uncertainty of data in the world. Internal data approximations are used to simplify complex sets of data to make reasoning more tractable. First, let us consider external data approximations.

2.1.1. *External Data Approximations*

An external data approximation involves a set of quantities for which the system is given approximate values typically via imperfect sensors. Let Q_E be a vector $\{q_1, q_2, q_3, ..., q_n\}$of quantity variables and A_E be a vector of their corresponding approximate values $\{a_1, a_2, a_3, ..., a_n\}$. Every q_i exists along a continuous dimension $D(q_i)$. We adopt the following model for external data approximations. For each quantity q_i there is a measurable value knowable to the system denoted $M_V(q_i)$. There is also an actual world value not knowable to the system denoted $A_V(q_i)$. For a sensor to yield a valid approximation we require that $P(M_V(q_i) = a_i | A_V(q_i) = t)$ is monotonic in $|t - a_i|$. In other words, given the actual value of t of a sensed quantity $q_i(A_V(q_i) = t)$, the likelihood of measuring value a_1 is greater than measuring value a_2 in just those cases that a_1 is closer to t than is a_2. More precisely:

$$\forall_{i=1}^{n} P(M_V(q_i) = a_1) > P(M_V(q_i) = a_2) \iff |a_1 - t| < |a_2 - t| \quad (1)$$

In the case of external data approximations, the value vector A_E is the best information the system has about the values of the quantity variables Q_E. The only way to improve this information is to interact with the world. For purposes of planning with the data represented by the approximations, the system behaves as if $Q_E = A_E$. The qualitative definition of a data approximation is never employed during planning, only when analyzing failures.

For instance, suppose one is using a tape measure to measure the length of a piece of wood. The reading obtained fits the criteria for an external data approximation. The quantity q_i being measured is the length of the wood and a_i is its measured approximate value. The quantity q_i exists along the continuous dimension $D(q_i)$ of length. It is reasonable to believe that the error between the measured and *true* value is likely to be small. The only way better information can be obtained is through further interaction with the world: obtaining and using a better measuring device, measuring more times and averaging the results, etc...

In robotics, external data approximations can be used to represent values read from sensors, which are inherently uncertain. For instance, the position of a block, as sensed by a visual system, would be represented with an external data approximation.

2.1.2. *Internal Data Approximations*

With an internal data approximation, the system chooses the values A_I of the quantity variables Q_I with a data approximation procedure. This is often motivated by the need

to simplify the representation so reasoning can be performed more efficiently. Internal data approximations can be adjusted through the system's reasoning alone. One can think of internal data approximations as being like external data approximations except that the actual value of the variable represents the optimal setting as defined by a system utility function.[2] The measured value for a quantity corresponds to the value chosen by the system in an attempt to maximize utility.

Recall our earlier tape measure example. Suppose that we would like to use the measurements we are making to decide how much wood is needed for a project. We might consider rounding up to the nearest inch in making the measurements. This makes it easier to compute the wood needed at the cost of possibly requesting too much. On top of the external data approximation, due to the uncertainty of the measurement, we have added an additional internal approximation to permit easier planning of the wood requirements.

In robotics, geometric object models are examples of internal data approximations. A simplified geometric representation can be far more efficient to reason about than the complex raw data returned by a vision system. However, in seeking a simplified representation, accuracy is sacrificed. The system has thus introduced further uncertainty.

2.2. *Plan Parameters*

Permissive plans employ continuous numeric parameters which can be tuned to affect plan permissiveness. It is important that these parameters depend on the context in which the plan is applied. Suppose we have a parameter which specifies the height which a manipulator must be above the workspace to safely navigate without collisions. The possible settings for this parameter are a function of the highest object in the workspace. Therefore, it depends on context. This means a generalized plan schemata must choose its parameter values at plan application time.

Parameters are chosen based on a *quality* function maintained in the schemata which indicates the current "best" parameter choices for the context in which the schemata is being applied. The quality function is represented by a collection of preferences which qualitatively describe the behavior of the function over a set of intervals.

Let Q_P be the set of plan parameters and A_P be the set of their respective values. Every $q_i \in Q_P$ is defined along a continuous dimension $D(q_i)$. Let there be a low bound $L_{low}(D(q_i), C)$ and a high bound $L_{high}(D(q_i), C)$ on the values which q_i may assume along dimension $D(q_i)$ in context C. A *context* is a partial world state specification. Figure 1 gives a pictorial representation for the dimension $D(q_i)$. The plan's parameter values A_P must also be dependent on context. The system chooses A_P based on a set of preferences $P(q_i, C)$ for each q_i in context C.

A preference $p \in P(q_i, C)$ is a 3-tuple $< V_{low}(C), V_{high}(C), r >$ where $V_{low}(C)$ and $V_{high}(C)$ form an interval dependent on context, and $r \in$ {*increasing, decreasing, constant*} is a relation describing the behavior of a *quality* function $F_{Q,q_i,C}$ in that interval. Additionally:

$$L_{low}(D(q_i), C) \leq V_{low}(C) \leq V_{high}(C) \leq L_{high}(D(q_i), C) \tag{2}$$
$$P(q_i, C) \textit{ is inconsistent } \iff \exists p_i, p_j \textit{ such that}$$

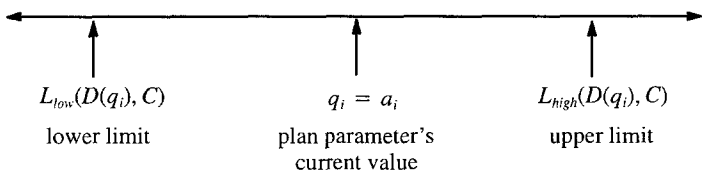


Figure 1. A Graphic Representation of Dimension $D(q_i)$

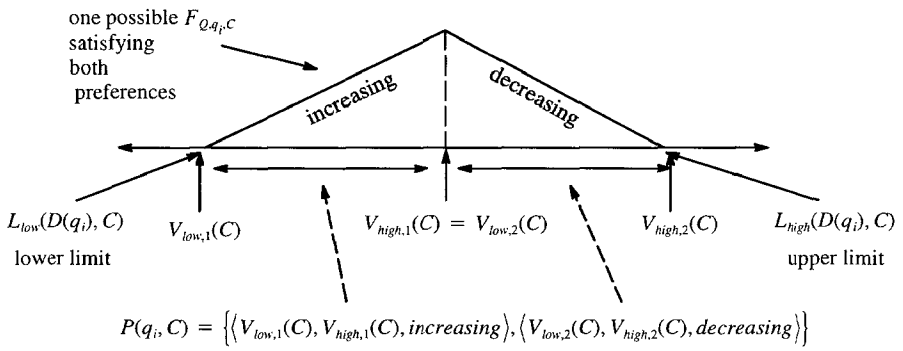


Figure 2. A Graphic Representation of Two Consistent Preferences Comprising A Quality Function

$$
\begin{gathered}
p_i = < V_{il}(C), V_{ih}(C), r_i > \in P(q_i, C) \wedge \\
p_j = < V_{jl}(C), V_{jh}(C), r_j > \in P(q_i, C) \wedge \\
r_i \neq r_j \wedge \\
\begin{pmatrix} V_{il}(C) < V_{jl}(C) < V_{ih}(C) \vee \\ V_{il}(C) < V_{jh}(C) < V_{ih}(C) \end{pmatrix}
\end{gathered}
$$

Figure 2 shows two consistent preferences comprising a quality function. An incompletely specified preference is one which has an undetermined value for $V_{low}(C)$ or $V_{high}(C)$. If incompletely specified preferences are used, as they are in our GRASPER implementation discussed in Section 3, the system must provide a procedure for assigning values to the undetermined $V_{low}(C)$'s and $V_{high}(C)$'s. The procedure should attempt to find assignments such that the resulting set of preferences is consistent as defined above.

The overall quality function F_Q determines the choice made by the plan for A_P. $F_{Q,q_i,C}$ is only specified in terms of the qualitative behavior of its regions as defined by the preferences $P(q_i, C)$. A plan parameter q_i may not be able to take on all values a_i, where $L_{low}(D(q_i), C) \leq a_i \leq L_{high}(D(q_i), C)$, in a given context C. We therefore represent the set of values q_i can take on in context C as $R(q_i, C)$. Let $MAX(F_{Q,q_i,C}, X)$ be a set of potential maxima for the function $F_{Q,q_i,C}$ applied to the set of values X. For $x \in X$,

$L_{low}(D(q_i, C)) \leq x \leq L_{high}(D(q_i), C)$ because the function is only defined over that interval. With only qualitative information about the quality function only potential maxima can be identified. The plan must choose a value for $a_i \in MAX(F_{Q,q_i,C}, X)$. Several strategies are possible. The GRASPER implementation chooses the value closest to the current setting for q_i. In general, the plan chooses a value for one or more parameters $q_i \in MAX(F_{Q,q_i,C}, R(q_i, C))$.

We assume limited interactions between plan parameters. L_{low}, L_{high}, and all V_{low}'s and V_{high}'s are dependent on context and may be a function of other plan parameters. We enforce that they must maintain their relative ordering under any possible valuation of the other plan parameters. In general: given an ordered, consistent set of preferences $\{P_1(q_i, C), P_2(q_i, C), ..., P_n(q_i, C)\}$ where each
$P_j(q_i, C) = < V_{low,j}(C), V_{high,j}(C), rel_j >$ then:

$$\forall C[(k < l) \Rightarrow [\ L_{low}(D(q_i), C) \leq V_{low,k}(C) \leq V_{high,k}(C) \leq V_{low,l}(C) \leq V_{high,l}(C) \leq L_{high}(D(q_i), C)]] \quad (3)$$

2.3. *Planning with Data Approximations*

Constructing a plan for a specific goal making use of data approximations is the same as with traditional planning techniques. The data approximations permit a simplified model of the world, simplifying the reasoning involved. No explicit reasoning about the fact that data approximations were employed takes place during plan construction or application. Generalized plan schemata can be constructed using the EGGS algorithm (Mooney & Bennett, 1986). The choice of parameter values for the generalized plan schemata, however, will remain dependent on the context in which the generalized plan is applied.

The price for approximation of the world is the increased potential for failure due to discrepancies. However, the strength of the permissive planning approach is in the ability to tune plan parameters to succeed in spite of these discrepancies. To achieve this we require that the system have an ability to recognize failures and to tune plan parameters so as to reduce the likelihood of encountered failures. The next two subsections discuss these requirements. First, we define expectations which are constructed during the planning process and allow failures to be recognized during execution. Next, we discuss how plans are refined so as to become increasingly permissive.

2.4. *Expectations*

We are interested in improving plan performance with respect to the real world. Failures occur when feedback from the real world contradicts the prediction of the simplified model which employs data approximations. Expectations are predictions of the simplified model for an action in terms of conditions which can be efficiently monitored in the world. In order to detect execution failures, all actions carried out in the world must be monitored. The important parts of a monitor are illustrated in Figure 3. First, the monitor must specify

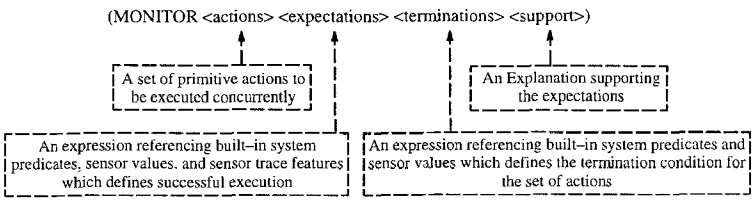


Figure 3. Syntax for Monitored Actions

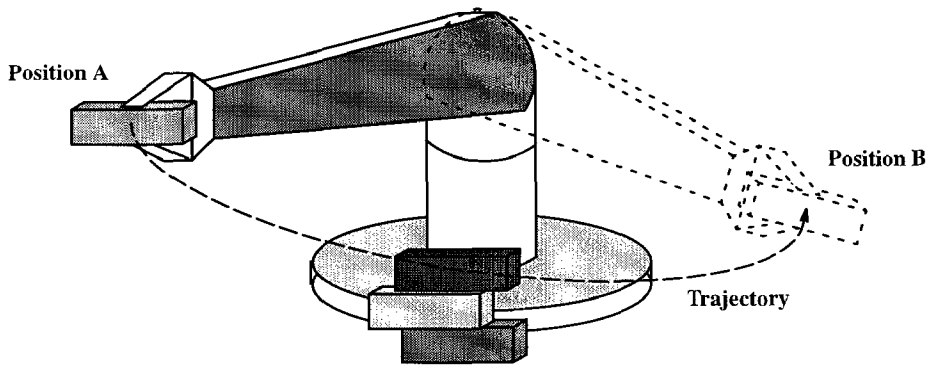


Figure 4. An Action for which Expectations Are Specified

the set of concurrent primitive actions to be carried out. Next, a set of constraints on the readings for sensors must be provided which define expected operation of the actions. A set of terminations is also provided in terms of sensor values which define termination of the actions. If the terminations are satisfied and none of the expectations were violated, the actions have succeeded. Lastly, there must exist a justification for the expectations in terms of the simplified model. As part of the planning process, all actions must have a justified set of expectations.

For instance, suppose that the robotic manipulator shown in Figure 4 is to be moved from position A to position B along some trajectory. The move is composed of a set of primitive joint moves for the manipulator which are to take place in a coordinated fashion. These form the set of concurrent primitive actions. The expectation is that the manipulator strikes nothing during the movement. This could be confirmed by force sensors on the manipulator. If none of the force sensors read sufficiently high to indicate the presence of an external force then it is assumed that the manipulator struck nothing during the move. The termination for the action set is position B as specified by the readings of a set of joint encoders on the manipulator's joints. The support for the expectations involves reasoning about the spatial occupancy of nearby objects in the simplified model. The expectation of sensing no external forces during the move could not be satisfied if the space occupied by

the stack of blocks in Figure 4 intersected the space occupied by the manipulator during the trajectory.

2.5. *Refining Failing Plans for Increased Permissiveness*

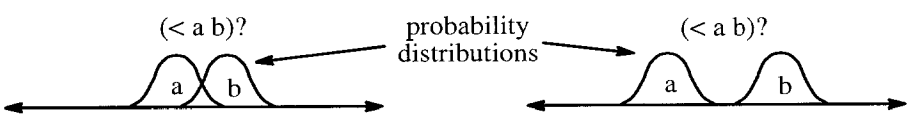
Failures are defined by expectation violations. These occur when the supporting proof for an expectation is valid in the model but is contradicted by real-world experience. That difference triggers the first phase of failure recovery: generating a qualitative explanation of how to tune plan parameters to reduce the chance of the failure in the future.

2.5.1. *Qualitative Tuning Explanations*

In order to diminish the chance uncertainty-related failures, it is necessary to decide which plan parameters to tune and how to tune them. In our model, failures can always be attributed to poor data approximations. In order to devise a strategy for tuning parameters so as to decrease the likelihood of a failure, it is necessary to reason about the relationships which exist between data approximate quantities, the failing expectations, and tunable plan parameters.

We employ a qualitative model of the relationship between continuous quantities.[3] Let $Q_+(a,b)$ signify that the magnitude of quantity b positively influences the magnitude of quantity a. Similarly, $Q_-(a,b)$ means the magnitude of quantity b inversely influences the magnitude of quantity a. That is, if $a = f(b)$ the minimum we must know to create such a relation is the sign of $\frac{\partial f}{\partial b}$. If $a = f(b)$ and $\frac{\partial f}{\partial b} > 0$ then $Q_+(a,b)$ holds. If $a = f(b)$ and $\frac{\partial f}{\partial b} < 0$ then $Q_-(a,b)$ holds.

Quantitative predicates employed by the system have one of two basic intents. Either they are *calculation predicates*, whose purpose is to compute some value (e.g. a predicate for subtraction), or they are *test predicates*, which are designed to fail for certain sets of inputs (e.g. a predicate for performing a less-than comparison). There is no way to vary the probability of success of a calculation predicate since they always succeed. A test predicate's probability of success, is sensitive to the probability distribution of its argument quantities. In the diagram below, the less-than test on the right has



a higher probability of succeeding given the illustrated probability distributions for its arguments than the one on the left. While probability distributions are difficult to define and work with, recall the simpler qualitative view of the probability distribution defined for data approximations in Section 2.1.1: probability density decreases monotonically as one moves either higher or lower away from the central value. Therefore, if we can tune one of the arguments to a test predicate we can affect its chance of success. Let DA(q) signify that q is a data approximate quantity. Now let us introduce the notation $PQ_+(p,q)$ to express

that the magnitude of the quantity q directly influences the magnitude of the probability of success of test predicate p. Similarly, $PQ_{-}(p, q)$ indicates that the magnitude of quantity q negatively influences the magnitude of the probability of success of test predicate p. This provides a mechanism for connecting the probability of a predicate being satisfied with the magnitude of a quantity. The rule below is one of several which follow from our definition for data approximations:

$$PQ_{-}(a < b, a) \Leftarrow DA(q), Q_{+}(b, q). \tag{4}$$

The rule states: if q is a data approximate quantity and hence uncertain and directly influences the magnitude of a quantity b, the likelihood of $a < b$ succeeding is inversely proportional to the magnitude of a. Let PP(q) indicate that the quantity q is a plan parameter and hence is tunable. Let $PS_{\uparrow}(p)$ signify that the probability of success of predicate p is increasing. Let $Q_{\uparrow}(q)$ signify that quantity q is increasing. Therefore we could use the following rules to increase the probability of a predicate p give than q is a plan parameter:

$$PS_{\uparrow}(p) \Leftarrow PQ_{+}(p, q), Q_{\uparrow}(q). \tag{5}$$

$$Q_{\uparrow}(q) \Leftarrow PP(q). \tag{6}$$

The second rule above asserts that we can increase plan parameters to achieve goals (because they are tunable).

It must also be possible to propagate the qualitative probabilities of predicates. Let $ANT(p_1, p_2)$ indicate that p_2 is an antecedent of a rule for which p_1 is a consequent. One sound rule for propagation of qualitative probabilities across rules can then be expressed:

$$PQ_{+}(p_1, q) \Leftarrow ANT(p_1, p_2), PQ_{+}(p_2, q), \forall x[[ANT(p_1, x) \wedge x \neq p_2] \Rightarrow \neg PQ_{-}(x, q)] \tag{7}$$

The general rules required to construct a qualitative tuning explanation fall into four categories:

general qualitative inference rules – inference rules necessary to reason about increasing and decreasing quantities

Example:

$$Q_{\uparrow}(x) \Leftarrow Q_{+}(x, y), Q_{\uparrow}(y). \tag{8}$$

qualitative predicate definitions – rules providing qualitative definitions for system predicates relating quantities

Example:

$$Q_{+}(x, y) \Leftarrow [x = y + z]. \left(because(x = f(y)) \wedge \left(\frac{\partial f}{\partial y} = 1 > 0\right)\right) \tag{9}$$

approximation definition rules – rules defining the behavior of test predicates using data-approximate quantities

Example:

$$PQ_{+}(a < b, b) \Leftarrow DA(q), Q_{+}(a, q). \tag{10}$$

qualitative probability rules – rules about the propagation of qualitative probabilities

Example: Rule 7 above.

The qualitative tuning explanation is a sound proof of how to positively influence the probability of success of the predicate which supported the failing expectations. The procedure for constructing the tuning explanation and tuning the plan parameters as a result is as follows:

1. Compute the set P of generalized preconditions and effects for the plan justification structure using the EGGS or EBG algorithms.
2. Take all generalized variables which are quantitative arguments to every predicate $p \in P$ as quantity variables for the qualitative reasoning process.
3. Find all qualitative influences among these quantity variables. This is possible since, if two plan quantities are related, we know the exact functional relationship.
4. Construct a proof based on the qualitative rules discussed above for how to unambiguously qualitatively increase the probability of success of the predicate supporting the expectations to the failed action.
5. Collect the set of quantity increases and decreases justified by the fact that plan parameters PP(q) are tunable. This amounts to finding applications of the rules: $Q_{\uparrow}(q) \Leftarrow PP(q)$ and $Q_{\downarrow}(q) \Leftarrow PP(q)$.
6. For each of the tunable plan parameters in the set collected above, add a new preference to the set of preferences for that parameter. This preference will be incomplete, specifying as one of it bounds a general expression for the point at which the failure occurred and as its relation increasing or decreasing as given in step 5. The other bound will be realized in conjunction with the neighboring preferences.

Next, we give an illustration of step 6 for adding preferences and updating the associated quality function. The overall refinement procedure is demonstrated in more detail with regard to the GRASPER implementation in Section 3.

2.5.2. *Tuning Plan Parameters: An Example*

The actual tuning of a plan parameter q_i is based on the addition of new preferences to the set $P(q_i, C)$. Consider the situation depicted in Figure 5. The goal is to grasp Block2, a relatively flat intricately shaped piece. The workspace also contains Block1, a bigger

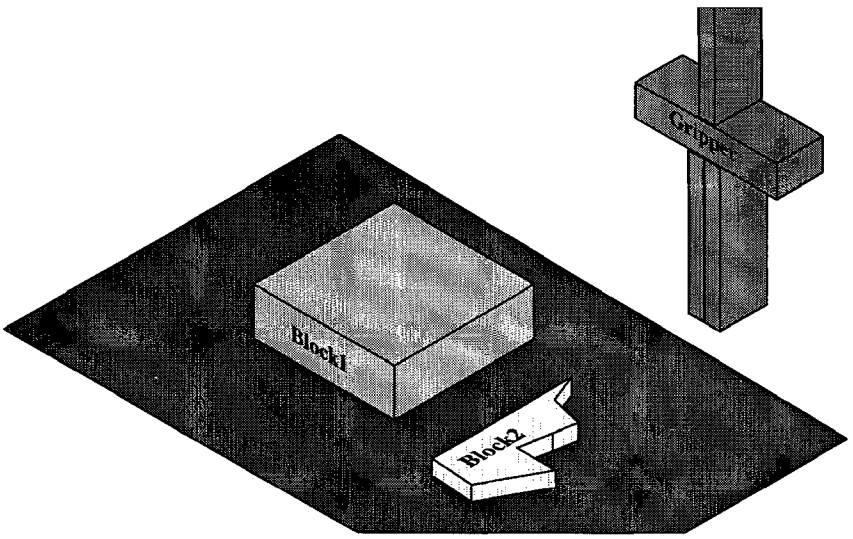


Figure 5. The Gripper In Its Initial Configuration

Current Preferences	Preferences Realized As	Quality Function
<?,?,*constant*>	a = $L_{low}(D(q_{ow}))$, b = $L_{high}(D(q_{ow}))$ <a,b,*constant*>	a b

Figure 6. Initial Preference Set and Quality Function for the Opening Width Plan Parameter

block which is fairly close to Block2. The only grasping faces on Block2 which can lead to a successful grasp require that one of the gripper fingers be inserted between Block1 and Block2. The system's plan for accomplishing the grasping goal involves selecting a grasping site, moving to a position above the center of the grasping site, orienting to the correct angle for the grasp, opening the gripper to surround the object, moving the gripper down to the table to surround the object, closing the gripper on the object, and lifting the object. Although many parameters are involved in this plan, our example focuses on the parameter for determining proper opening width. The initial preference for the opening-width parameter is shown in Figure 6. In our example we will assume that the initial preference is always for a constant function over the range of the bounds on the parameter. This makes sense in that initially no preferences would imply that all settings (within the bounds) of values for the parameter are equally good. Once the first preference is learned by the system, the initial default preference is removed. The flat quality function defined by the default preference has a continuous range of potential maxima which exist between the

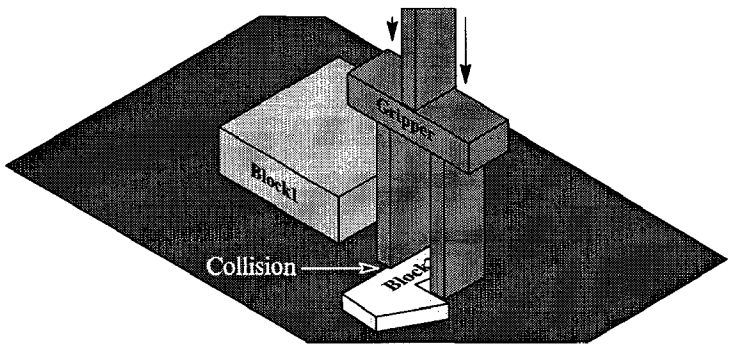


Figure 7. The Gripper Collides With Block2 As It Moves Down To Surround It

Current Preferences	Preferences Realized As	Quality Function
<a,?,*increasing*>	a = $L_{low}(D(q_{ow}))$, b = $L_{high}(D(q_{ow}))$ <a,b,*increasing*>	a b

Figure 8. Preferences and Quality Function for Opening Width After Collision Failure with Block2

bounds. In our example, the system will select one closest to the current opening width of the gripper. One could have been picked randomly as well but this method has the additional benefit of resulting in less gripper movement. The gripper was initially in a closed position so the nearest setting for the opening-width parameter is one exactly equal to the width of Block2 at the grasping site. The bound a in Figure 6 corresponds with this width.

The plan is executed and fails with an unexpected collision when moving down to surround Block2. Figure 7 shows the situation at the time of failure. The collision occurred near Block2 and at a point lower than the height of Block1. The qualitative tuning explanation therefore indicates that increasing the opening width for surrounding Block2 causes the fingers to be farther from Block2 at that point in the grasp operation and decreases the likelihood of a collision failure with Block2. The collision occurred at point a, a bound on the quality function shown in Figure 6. The tuning explanation suggests that at that point, increasing the parameter is desirable. A new preference is therefore added to the opening-width parameter. Figure 8 shows the new preference and resulting quality function. The algorithm for realizing preferences now assigns the increasing preference to extend over the complete interval because no additional preferences are present. The plan is now more permissive in that the move-down-to-surround portion of the plan should function in spite of errors associated with the data approximations to Block2. However, the system has only learned one of the preferences that are important in this situation. The plan is now applied

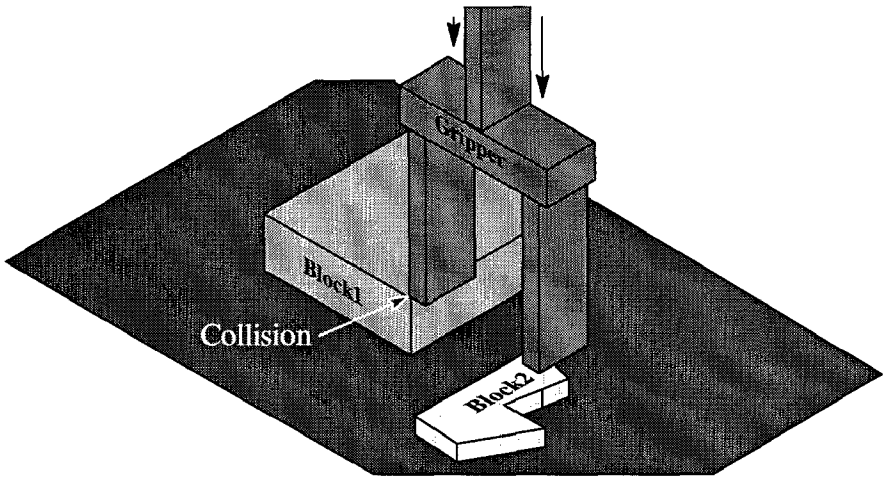


Figure 9. The Gripper Collides With Block1 As It Moves Down To Surround Block2

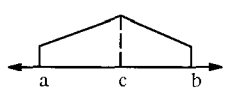
Current Preferences	Preferences Realized As	Quality Function
<a,?,*increasing*> <?,b,*decreasing*>	$a = L_{low}(D(q_{ow})), b = L_{high}(D(q_{ow}))$ c=(a+b)/2 <a,c,*increasing*> <c,b,*decreasing*>	

Figure 10. The Preferences and Quality Function for the Opening Width Parameter After a Collision with Block1

again to grasping Block2. The value for the opening-width parameter is now chosen to be the maximum possible opening width prior to striking Block1. This is the upper bound b for the opening-width parameter which corresponds to the only potential maximum of the new quality function shown in Figure 8.

The plan is executed and fails once again. This time the failure occurs near Block1 and at the height of Block1 as shown in Figure 9. The qualitative tuning explanation indicates that decreasing the value of the opening width would decrease the chance of a collision with Block1. As a result, another preference is added indicating that lesser values are to be preferred than b, the point at which the failure occurred. The new set of preferences form a new quality function shown in Figure 10. In order to concretely realize the two incomplete preferences, a third general point c must be created between the bounds a and b. Here, the algorithm creates a general expression for c by taking the midpoint between the values returned by the expressions for a and b, the nearest set of bounds. As a result, the system now has increased the plan permissiveness with respect to errors in the representations for both Block1 and Block2. The failures encountered in this example were with the specific part

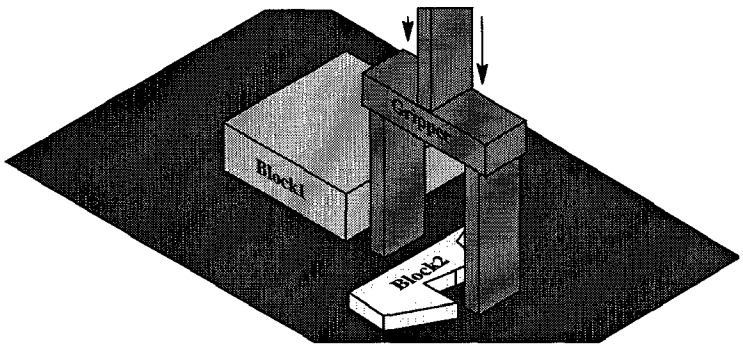


Figure 11. A Successful Surround Operation is Achieved in the Plan

of the plan that moves down to surround the object. It is the permissiveness of this portion of the plan that was increased by tuning the opening-width parameter. Other aspects of the plan which lead to problems will be tuned as well by other encountered failures. The point c now forms the only potential maximum of the new quality function for the opening-width parameter. Consequently, in applying the new plan, a midpoint between the two extremes is chosen for the opening-width parameter. This leads to the successful surround operation in the plan illustrated in Figure 11.

3. The GRASPER System

In order to demonstrate and test our approach we chose a complex real-world domain where uncertainty plays a role: robotic grasping. The goal in this domain is to learn plans which control a robotic manipulator to successfully grasp objects in its workspace. In fact, planning of grasps for arbitrarily shaped objects is still an open problem in robotics. Uncertainty is one primary difficulty in this domain. Sensors don't return precise information. Visual sensors seeking to identify or help represent the objects are very sensitive to placement of the light source. For example, an observer blocking some of the light may actually be effecting visual sensing of the objects. Force sensors used on the manipulator also are subject to errors so that the precise position which the manipulator first contacts an object is not known exactly. The robotic manipulator also cannot be completely precise in its movement. Intractability also plays a significant role in this domain. The system must represent the world in order to construct plans for carrying out its actions. For example, in using visual sensors to model an object or in recognizing objects and retrieving a pre-stored model, that model exists at some resolution. The greater the resolution, the more information that must be considered in planning to grasp the object. Therefore, in order to allow plans to be constructed in a reasonable amount of time, object models must be simplified. As discussed in the previous section, this amounts to introducing some error in return for planning efficiency. Altogether, the robotic grasping domain provides a challenging testbed for learning techniques.

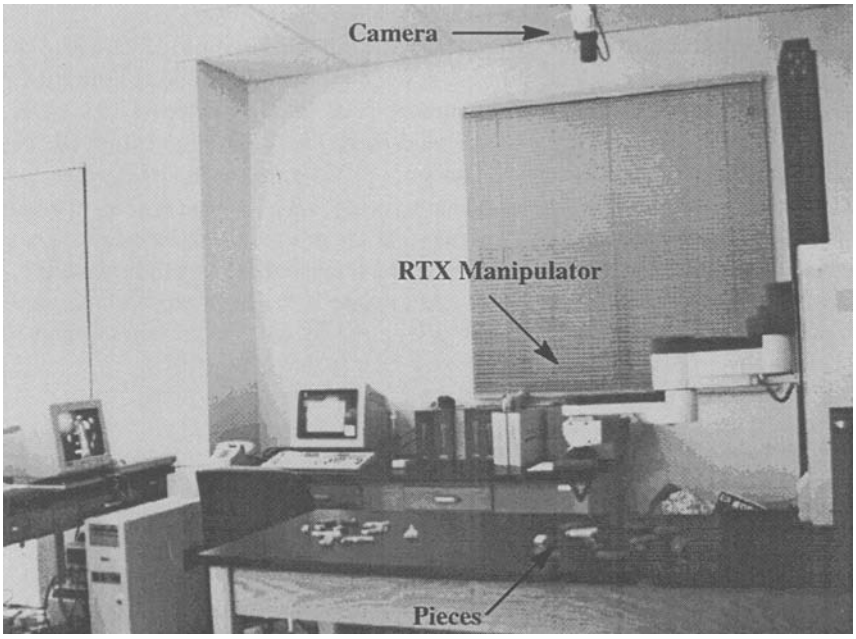


Figure 12. GRASPER Experimental Setup

Figure 12 shows the laboratory setup. The current implementation of the architecture is called GRASPER and is written in Common Lisp running on an IBM RT125. GRASPER is interfaced with a frame grabber connected to a camera mounted over the workspace. The camera produces bitmaps from which object contours are extracted by the system. The system also controls an RTX scara-type robotic manipulator. The RTX has encoders on all of its joint motors and the capability to control many parameters of the motor controllers including motor current. This gives the system a rudimentary capability of detecting collisions with the RTX gripper. If enough current (force) is applied to the motor to overcome friction of the joint and the position encoder indicates no movement, an obstacle has been encountered. This type of sensing gives feedback during execution of a plan when the camera's view of the workspace would otherwise be obscured. This precise control of the manipulator is ideal for carrying out monitored actions in the world.

Our current goal for the GRASPER system in the robotics grasping domain is to successfully grasp the plastic pieces from puzzles designed for young children. Since the pieces are relatively flat and of fairly uniform thickness, an overhead camera is used to sense piece contours. The GRASPER system could be applied to arbitrary three-dimensional objects as well given sufficient sensing hardware to recognize or model the objects in 3D. These pieces have interesting shapes and are large enough, yet challenging, to grasp. The goal is to demonstrate improving performance at the grasping task over time in response to failures. Some of the failures the current implementation learns to overcome, when using isolated grasp targets, include learning to open wider to avoid stubbing the fingers on objects, and learning to prefer more parallel grasping faces to prevent unstable grasps. Detailed examples of these two failures follow later in this section.

3.1. System Architecture

The system is organized as illustrated in Figure 13. There are three modes of operation. Input to the system is through the planning/explanation-based learning component shown in Figure 14. In the simplest mode, a goal is presented to the system which already corresponds to a plan in the knowledge-base whose preconditions are satisfied in the current state of the world. In this case, that plan is passed directly to the executive to be carried out. Secondly, a goal could be presented to the system which doesn't correspond to any known plans. In this case, an explanation is generated for how the goal can be achieved in the current state, the explanation is then generalized and packaged into a general plan which is then saved and passed on for execution. Lastly, a goal and observed action sequence can be given to the planning/ EBL component which then generates the explanation from both the goal and the observed actions. This is the preferred mode of operation of explanation-based learning systems because it can make explanation construction an easier task. In this last mode, the explanation is then generalized, packaged into a plan, saved, and the plan is passed on to the executive.

The executive instantiates the execution sequence associated with the plan with the appropriate bindings obtained from evaluation of the plan's preconditions in the current state. These actions will be monitored, having sensor expectations associated with them that define success or failure. Should a failure occur, as defined by the expectations, the plan

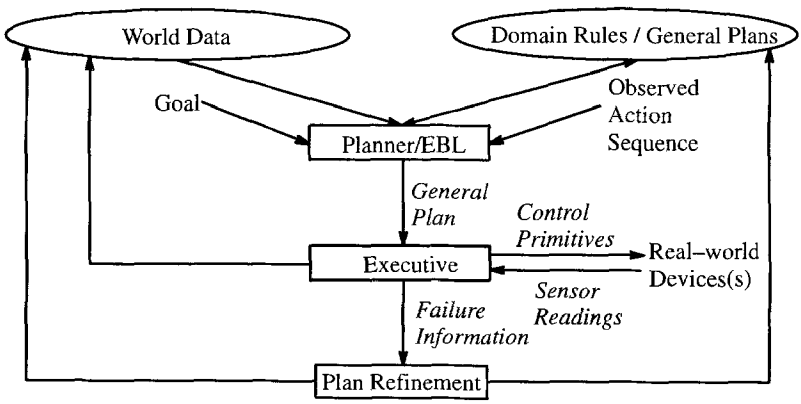


Figure 13. Approximation Architecture

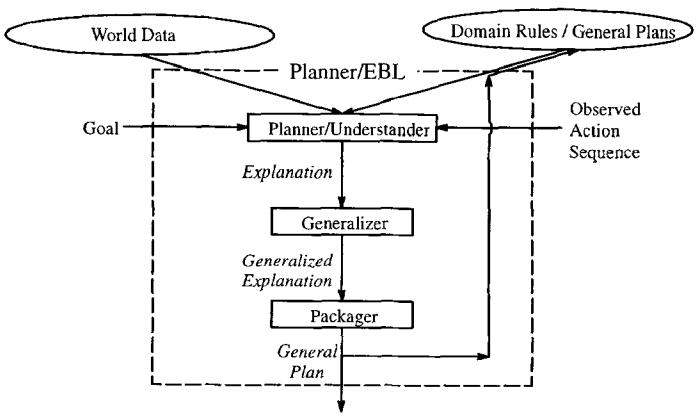


Figure 14. Approximate Explanation-based Learning

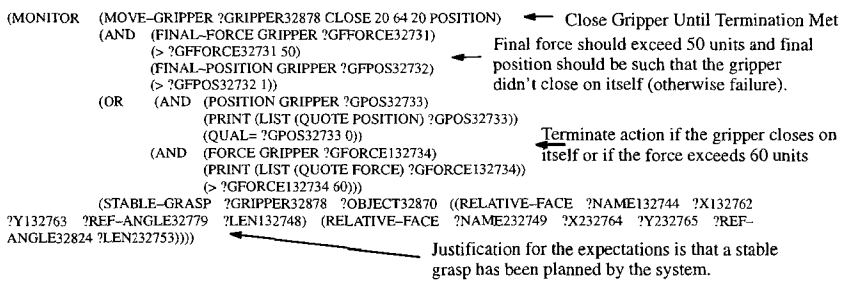


Figure 15. An Example of a Monitored Action

refinement module is called to increase plan permissiveness appropriately. If no errors occur, the system is ready for the next goal or goal/observation pair.

3.2. Execution & Monitoring for Robotics

In order for the system to improve its approximations when they don't perform well in the world, the system must have a monitoring capability. It is important that the system be able to represent what actions are to be carried out, what the expected outcome of those actions is, why that outcome is expected, and when the specified actions should be terminated. In the robotic grasping domain, the set of actions to be monitored are a set of motor commands to the manipulator. These may occur as individual motor moves as with a command to move the arm up the column in the case of our SCARA-type manipulator. A group of simultaneously applied motor commands may also be monitored. For instance, in moving the manipulator while grasping an object, force has to be continually applied to squeeze the object while the other joint moves are being carried out. In this case, expectations may apply both to whether the object is sensed between the fingers and whether an external force is sensed by the arm during the motion. Figure 15 gives a concrete example of a monitor employed for closing the robotic gripper on an object. The single action specified is for the gripper to begin closing from its current position. The expectation is that the final force of the gripper on the object exceed 50 units and that the gripper not close on itself. The action terminates when the gripper exerts a force greater than 60 units on the object or the gripper closes on itself. The expectation of feeling the object between the fingers with force greater than 50 units is justified by an explanation for why the planned grasp is stable (so the object will not slip away as force is applied). The specification of expectations as well as their justification facilitates the parameter tuning process.

The expectations and terminations are specified in DNF form and may reference predicates known to the system as well as sensors available on the manipulator. For actual execution on the robot, the sequence of monitored actions specified by a plan is compiled into a Common Lisp program for rapidly checking the sensors while the actions proceed. Many tradeoffs exist in the monitoring process. For instance, since sensors take time to read, the

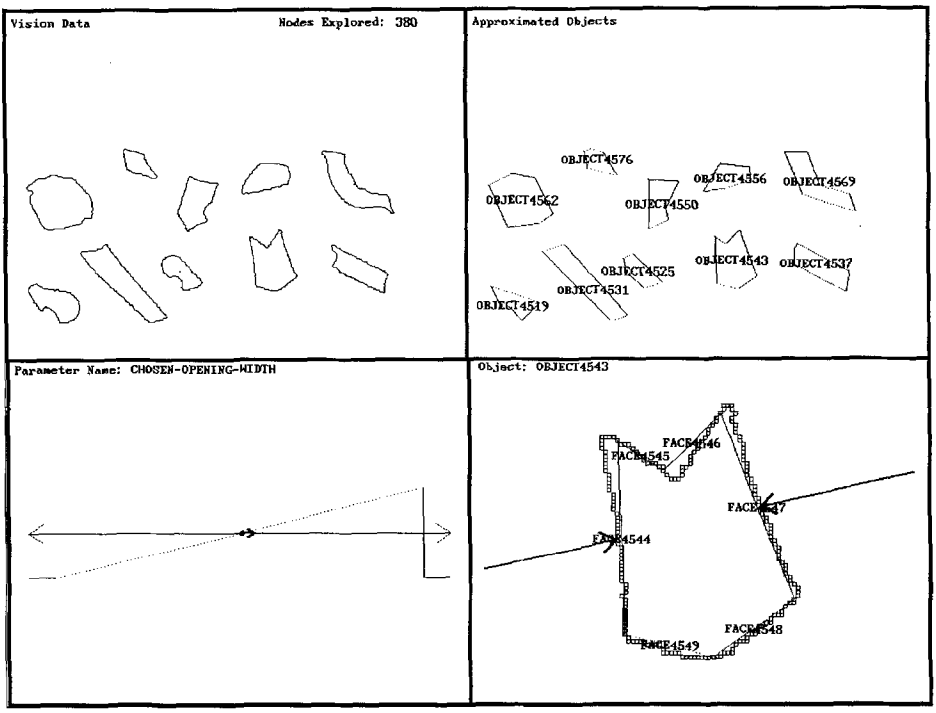


Figure 16. System Status Display During Grasp of Object4543

faster the actions are carried out, the smaller is the number of sensors readings which can be obtained. Furthermore, one can read more types of sensors during execution but each will have a lesser number of readings because of the time constraint. However, because of the permissive planning approach the plans become less sensitive to sensor error.

3.3. Detailed Examples of Learning Episodes

This section details two learning episodes with GRASPER. The first highlights diagnosis of a failure to surround the piece and the second a failure to choose the best grasping site.

3.3.1. Example 1

Figure 16 shows the system's status display during a grasping task. First, the system uses the camera to acquire contour information about objects in the workspace. These contours are shown in the upper left corner of the figure. Next, the contours are approximated with n-gons (internal data approximations) which result in (n2-n)/2 possible unique grasping face

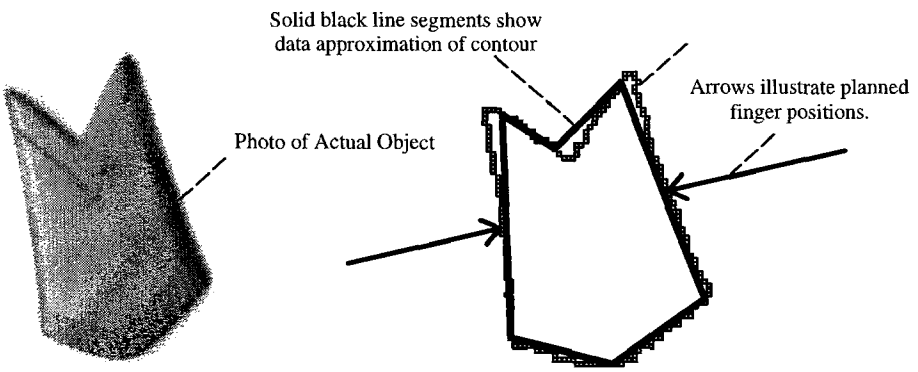


Figure 17. Grasp Target and Planned Finger Positions

pairs. These approximated object contours appear in the upper right quadrant of Figure 16. The algorithm chooses the value of n such that an approximation to the object is possible within a certain error threshold. The data approximated object representations as well as the current information about the state of the robot manipulator are asserted in the initial situation. The target object is then selected and an explanation is generated for how to achieve a grasp of the target. Figure 17 highlights the selected target object. The heavy line indicates the data approximation to the object contour while the lighter pixels show the actual sensed object contour points. The arrows indicate the positions of the leading edges of the fingers for the grasp position given by the produced explanation. The explanation for achieving *grasp-object* involves a total of about 300 nodes with a maximum depth of 10 levels.

Parameters are implemented by rules which choose their value dependent on context. When new preferences are added for a parameter, the associated rules are updated so as to choose potential maxima of the new quality function. Generalized plans refer to the consequents of the rules which choose parameter values. In this case, initially there were no preferences for the plan's opening width parameter other than it be a legal value between the width of the target object and the minimum of the maximum gripper opening and the distance to the nearest object. One of the initial rules for the opening width parameter is shown in Figure 18. This rule pertains to the case where the gripper is currently open less than the minimum opening to satisfy the surround goal in the plan. It then chooses the minimum opening which satisfies the goal because this is the closest potential maxima of the (initially flat) quality function. The rule therefore affects the separation of the arrows shown in Figure 17. After the explanation was generated, and its associated operator sequence executed, the monitored action shown in Figure 19 encounters a violation of the expected sensor readings. Figures 20 and 21 show traces of force and position, respectively, plotted against time (in seconds) into the overall operator sequence.

The original explanation for the *no-gripper-collision-object* goal indicated in the above monitored action is now suspect due to the violated expectations. A sketch of the specific

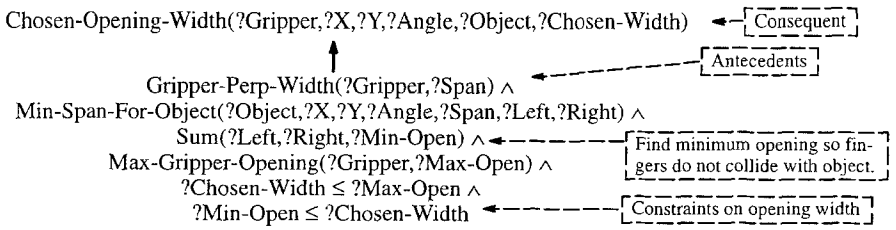


Figure 18. One of the Initial Parameter Rules For Opening Width

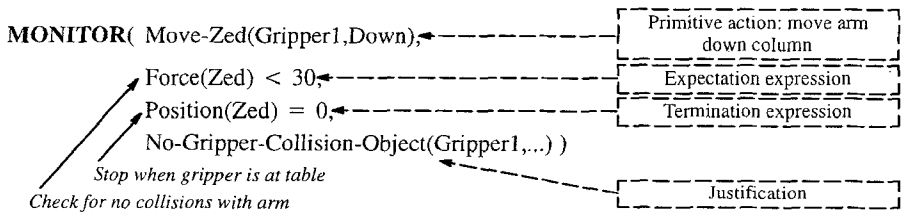


Figure 19. The Failing Monitored Action

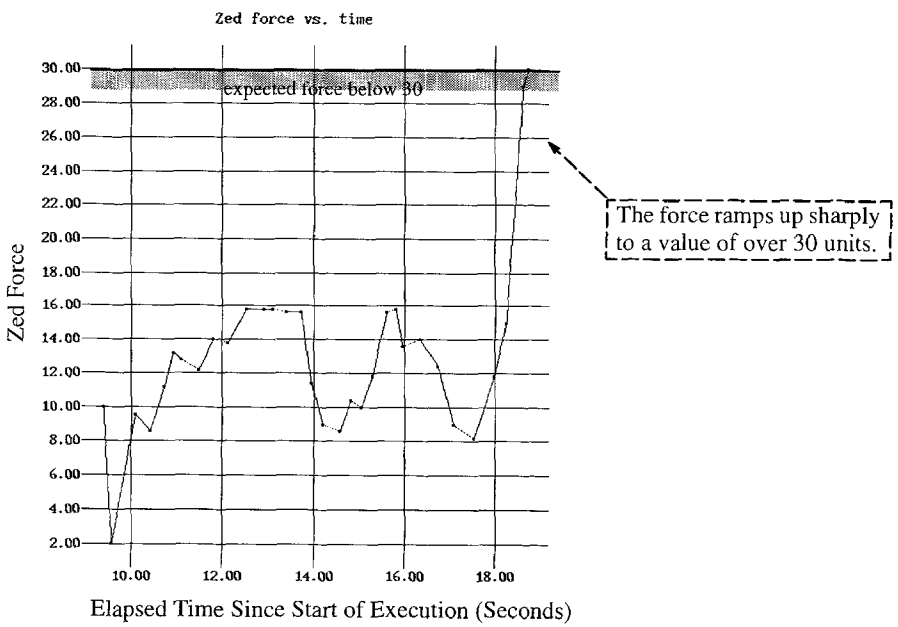


Figure 20. Zed Force vs. Time into Action Sequence

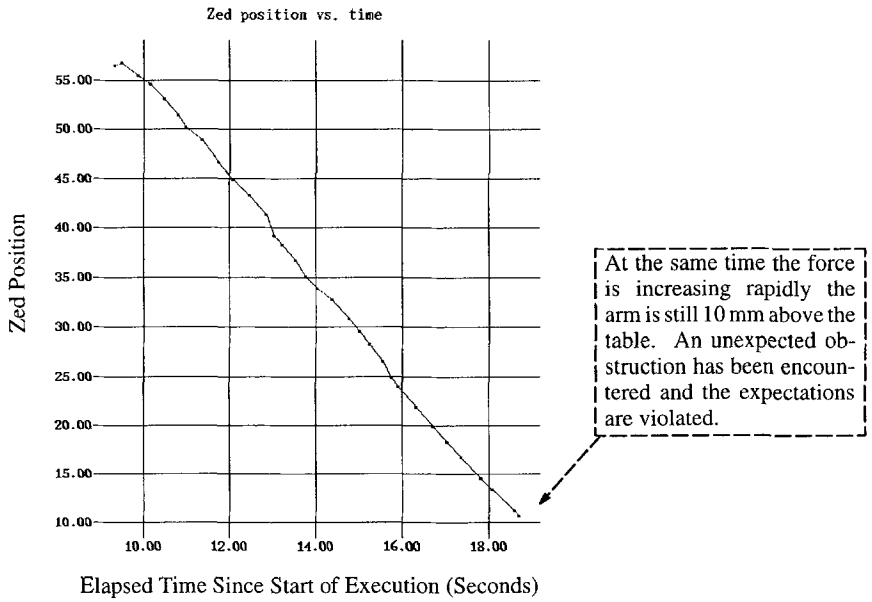


Figure 21. Zed Position vs. Time into Action Sequence

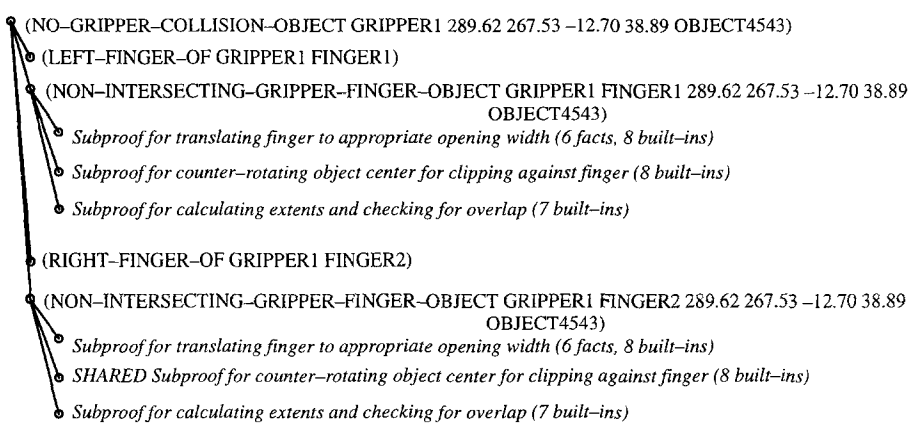


Figure 22. Explanation Specific to Failure

explanation is shown in Figure 22. This explanation for why no external force should have been sensed during the downward move of the gripper is the starting point for developing the qualitative tuning explanation. Data approximate quantities and tunable parameters employed in the plan support proof (explanation) are identified and asserted as such. A proof

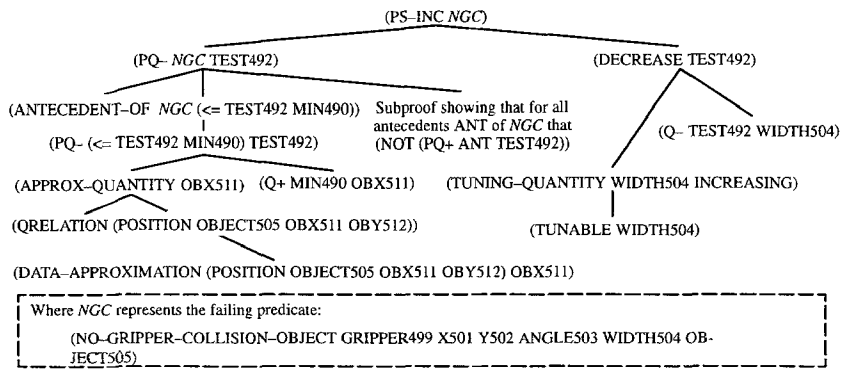


Figure 23. A Qualitative Tuning Explanation

is then constructed for increasing the probability of success of the *no-gripper-collision-object* goal. Figure 23 shows the qualitative explanation for how opening the gripper (increasing the opening-width tunable parameter) positively influences the probability that there will be no collision between the first gripper finger and the object. Table 1 gives the semantics for the predicates employed in the explanation. The topmost left-hand subtree establishes that decreasing the quantity TEST492 can positively influence the probability of success of the the probability of the *no-gripper-collision* predicate. This is because decreasing the quantity TEST492 can increase the probability of the predicate ($\leq$ TEST492 MIN490) which is an antecedent of a rule supporting the *no-gripper-collision* predicate and all of the other antecedents to that rule can be shown as non-decreasing with respect to decreasing TEST492.[4] The probability of success of the predicate ($\leq$ TEST492 MIN490) increases when TEST492 is decreased because MIN490 is influenced by a data approximate quantity. The right subtree of the proof establishes that the quantity TEST492 can be decreased because it is influenced inversely by a tunable parameter WIDTH504 which can be increased. The parameter WIDTH504 is the opening width parameter for the gripper.

The qualitative tuning explanation indicates that the chosen-opening-width parameter should be tuned. An increasing preference will be posted at the minimum opening width chosen in the failure. Figure 24 illustrates the shape of the chosen-opening-width parameter's quality function before (left) and after (right) tuning has occurred. After parameter tuning, the rules associated with parameter are updated Afterwards, the rule associated with this parameter reads a shown in Figure 25. This rule prefers selection of the peak of the newly re-calculated quality function which corresponds to opening as wide as the current situation permits. When the new more permissive plan is applied the resulting gripper finger positions are as illustrated in Figure 26.

(PS–INC ?pred)	*the magnitude of the probability of success of ?pred is increasing*
(ANTECEDENT–OF ?pred1 ?pred2)	*?pred2 is an antecedent of ?pred1 in the rule being analyzed*
(PQ– ?pred ?quant)	*the magnitude of the quantity ?quant negatively influences the magnitude of the probability of success of predicate ?pred*
(PQ+ ?pred ?quant)	*the magnitude of the quantity ?quant positively influences the magnitude of the probability of success of predicate ?pred*
(DECREASE ?quant)	*the magnitude of the quantity ?quant is decreasing*
(APPROX–QUANTITY ?quant)	*?qant is a data approximate quantity*
(Q+ ?q1 ?q2)	*the magnitude of quantity ?q2 positively influences the magnitude of quantity ?q1*
(Q– ?q1 ?q2)	*the magnitude of quantity ?q2 negatively influences the magnitude of quantity ?q1*
(TUNING–QUANTITY ?a ?b)	*plan parameter ?a is tuned according to the relation ?b*
(QRELATION ?pred)	*?pred is one of the predicates employed in the plan support structure (and employs continuous quantities which can be modelled qualitatively)*
(DATA–APPROXIMATION ?pred ?arg)	*the argument ?arg of the predicate ?pred is known to be a data approximate quantity*
(TUNABLE ?quant)	*the magnitude of quantity ?quant is a tunable plan parameter*

Table 1. Predicates Employed in the Tuning Explanation

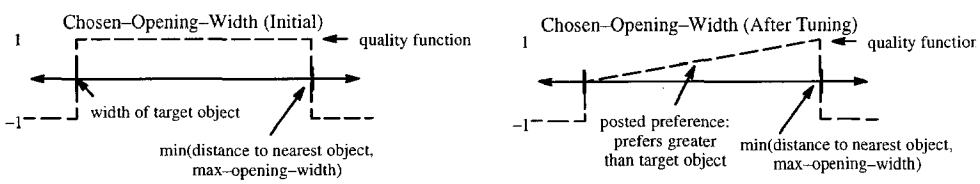


Figure 24. The Chosen-Opening-Width Parameter Quality Function After Learning New Preference

```
INTRA-RULE: R4611
   FORM:
      (CHOSEN-OPENING-WIDTH ?GRIPPER ?X ?Y ?ANGLE ?OBJECT ?RETURN)
   ANTS:
      (GRIPPER-PERP-WIDTH ?GRIPPER ?SPAN)
      (DISTANCE-TO-CLOSEST-OBJECT ?OBJECT ?X ?Y ?ANGLE ?SPAN ?RADIUS)
      (GRIPPER-FINGER-PARALLEL-WIDTH ?GRIPPER ?PSPAN)
      (DIF ?RADIUS ?PSPAN ?NRADIUS)
      (MAX-GRIPPER-OPENING ?GRIPPER ?MAX-OPEN)
      (MIN ?NRADIUS ?MAX-OPEN ?RETURN) <— peak chosen here
      (MIN-SPAN-FOR-OBJECT ?OBJECT ?X ?Y ?ANGLE ?SPAN ?LEFT ?RIGHT)
      (SUM ?LEFT ?RIGHT ?MIN)
      (<= ?MIN ?RETURN) <— must be wider than object
   CONS:
   PARAMETER: CHOSEN-OPENING-WIDTH
```

Figure 25. Rule Supporting Opening-Width Parameter After Tuning

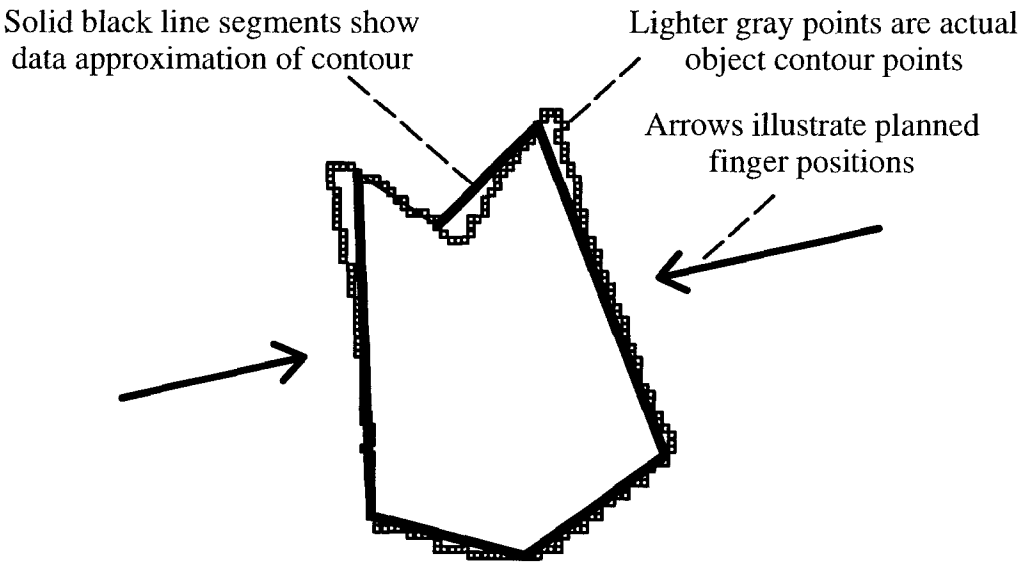


Figure 26. A Successful Wide Grasp

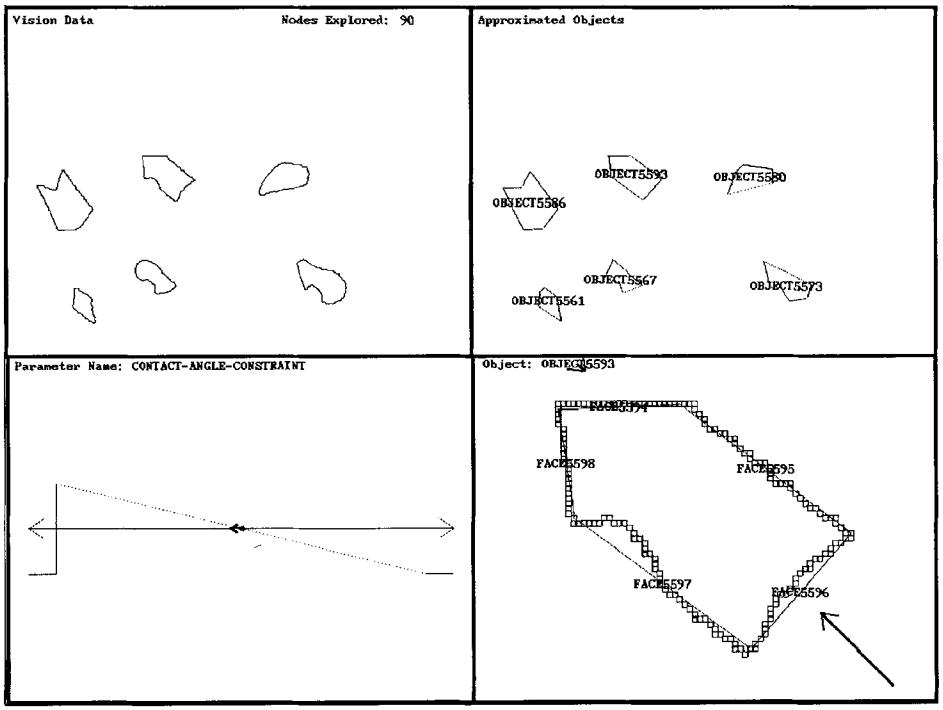


Figure 27. System Status Display During Grasp of Object5593

3.3.2. *Example 2*

Next, after the system has learned to open widely when grasping objects, a second object is presented for grasping. Figure 27 shows the system status display while initially planning the grasp for this object. Figure 28 highlights the selected target object. The dark line indicates the polygonal object approximation and the light colored pixels show the sensed object contour points. The arrows illustrate the planned positions for the fingers in the generated grasping plan. Notice, that the fingers are well clear of the object due to the tuned opening-width parameter learned in the first example. Generally, the opening-width parameter is the first one to be tuned because striking the objects while attempting to surround them is a fairly common error. However, the parameter regarding acceptable angles between contact faces still has only the initial default preference. The angle between chosen faces must be greater than 0 (parallel) and less than the angle at which slipping occurs according to the friction coefficient between the gripper and faces being grasped. All angles fitting these criteria are treated as equally good. Consequently, the two faces chosen for this grasp should be acceptable given the specified friction coefficient. The rule on which the contact-angle-constraint parameter is based is:

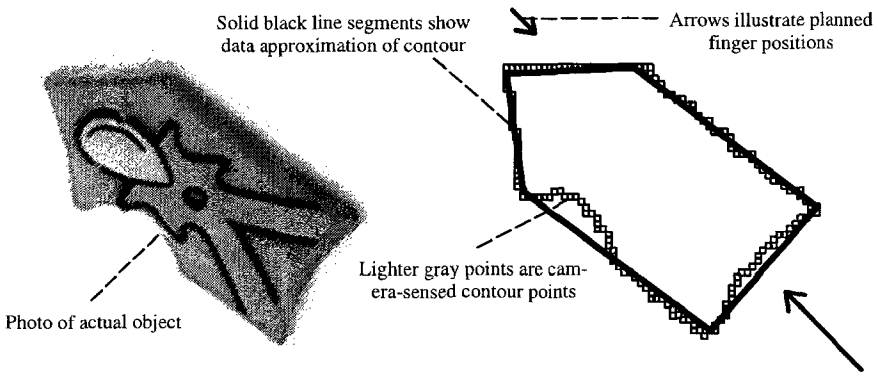


Figure 28. Grasp Target and Planned Finger Positions

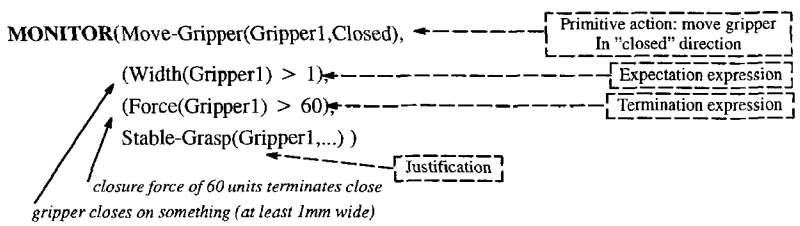
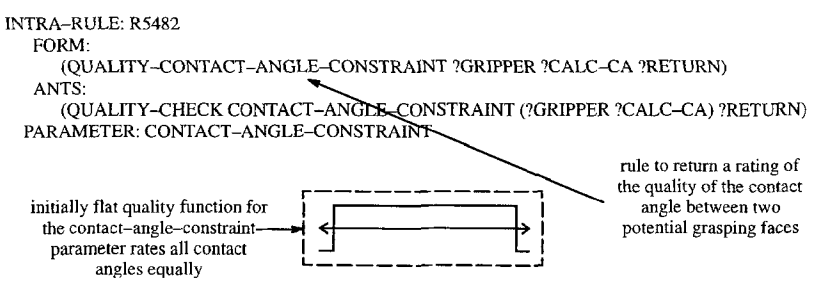


Figure 29. The Failing Monitored Action



This rule references a built-in predicate QUALITY-CHECK which returns a rating of the specified contact angle based on the current quality function for that parameter. The rating is used in choosing from among several candidate contact angles and is therefore only good in relation to other contact angles with respect to the same quality function. After the explanation is generated, and its associated operator sequence executed, the monitored action shown in Figure 29 encounters a violation of the expected sensor readings. The traces in Figure 30 and Figure 31 show, respectively, plots of gripper force and gripper position with time (in seconds) into the overall execution sequence.

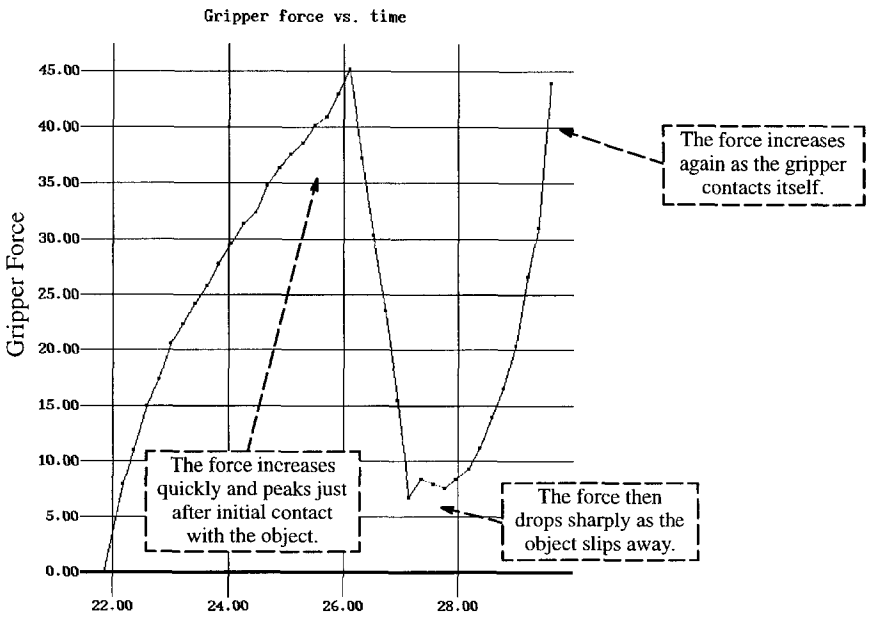


Figure 30. Gripper Force vs. Time into Action Sequence

The original explanation for the *stable-grasp* goal indicated in the above monitored action is now suspect due to the violated expectations. A sketch of the specific explanation is shown in Figure 32. This explanation for why a stable grasp should have been achieved is the starting point for developing the qualitative tuning explanation. Data-approximate quantities and tunable parameters employed in the plan support proof are identified and asserted as such. A proof is then constructed for increasing the probability of success of the *stable-grasp* goal. Figure 33 shows the qualitative explanation for how preferring a smaller contact angle positively influences the probability that a stable grasp will be achieved. Table 1 gives the semantics for the predicates employed in the explanation. The topmost left-hand subtree establishes that as the quantity A747 is decreased the probability of success of a stable grasp is increased. This is true because the predicate ($\leq$ A747 FANGLE635) is an antecedent of the rule supporting the stable grasp, the probability of success of ($\leq$ A747 FANGLE635) is influenced negatively by the magnitude of A747, and none of the other antecedents to the rule are influenced positively by the same quantity. The probability of success of ($\leq$ A747 FANGLE635) is influenced negatively by A747 because FANGLE636 is influenced by a data-approximate quantity FRIC634, the friction coefficient between the object faces and gripper. The topmost right-hand subtree shows that the quantity A747 is a tunable parameter which can be decreased.

The qualitative tuning explanation indicates that smaller contact angles should be preferred in choosing grasping faces. Figure 34 illustrates the shape of the contact-angle-constraint

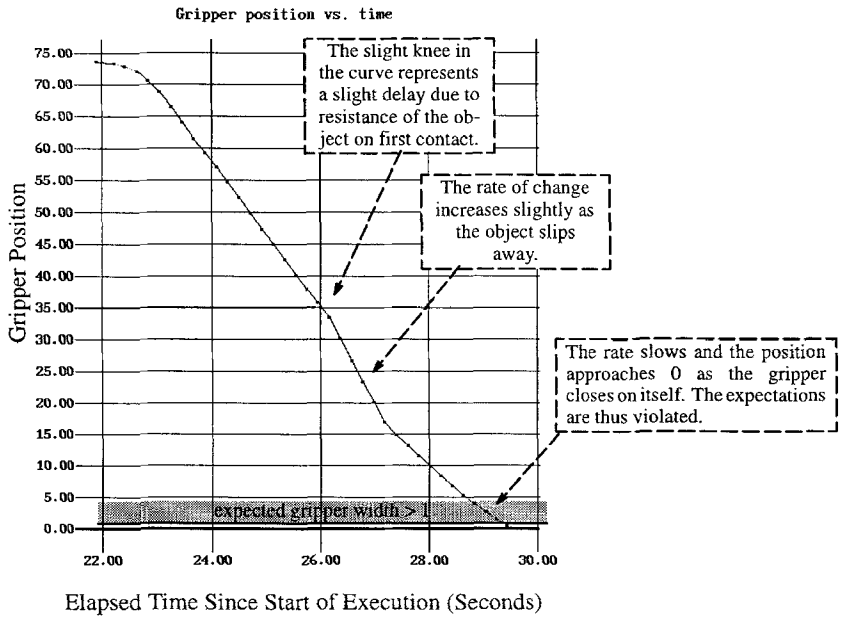


Figure 31. Gripper Position (Width) vs. Time into Action Sequence

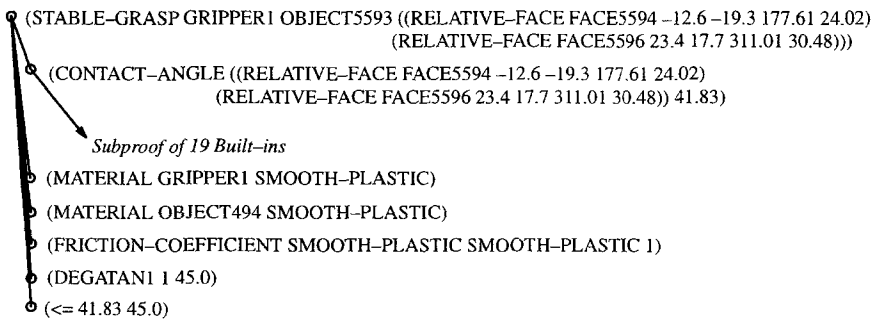


Figure 32. Explanation Specific to Failure

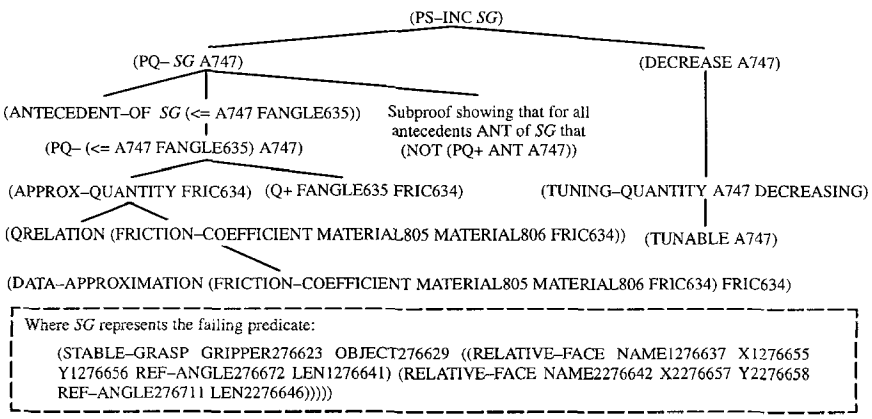


Figure 33. A Qualitative Tuning Explanation

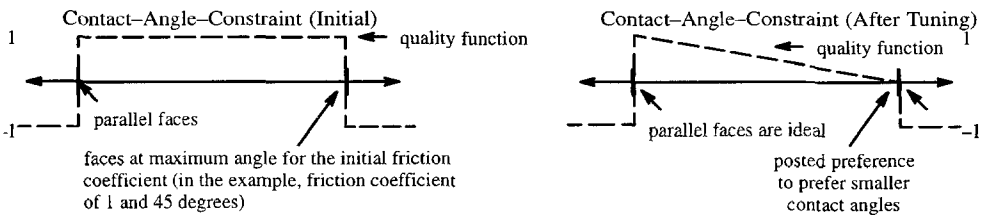


Figure 34. The Contact-Angle-Constraint Parameter Before and After Tuning

parameter's quality function before (left) and after (right) tuning has occurred. After parameter tuning, the rule which uses the parameter to rate contact faces (shown earlier) will take advantage of the new contact-angle-constraint parameter's quality function and choose a grasp position such that the two faces to be contacted are closest to parallel. Figure 35 shows the new more permissive plan as applied to the same object. In this case, the most parallel faces were preferred over those picked in the earlier application of the plan.

4. Empirical Results

The GRASPER system was given the task of achieving equilibrium grasps on the 12 smooth plastic pieces of a children's puzzle. Figure 36 shows the gripper and several of the pieces employed in these experiments. A random ordering and set of orientations was selected for presentation of the pieces. Target pieces were also placed in isolation from other objects. That is, the workspace never had pieces near enough to the grasp target to impinge on the decision made for grasping the target. The first run was performed with parameter

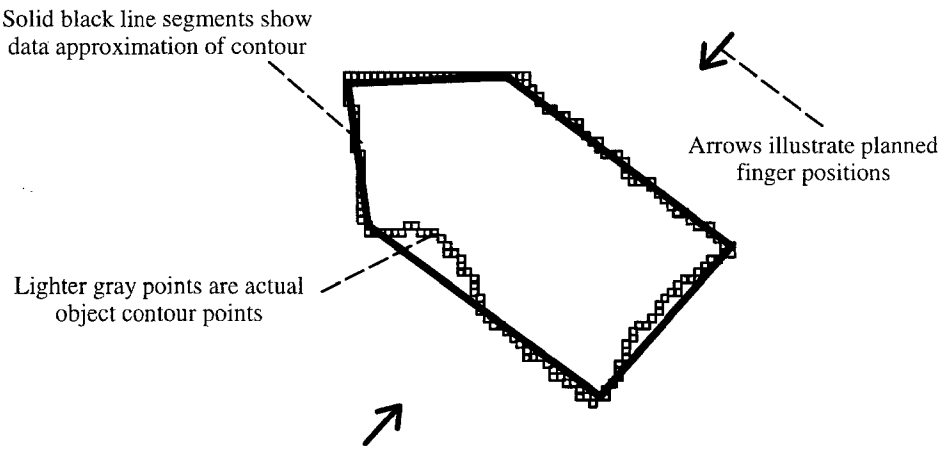


Figure 35. Successful Grasp Employing Tuned Parameter Constraining Contact Angle

Figure 36. Gripper and Pieces

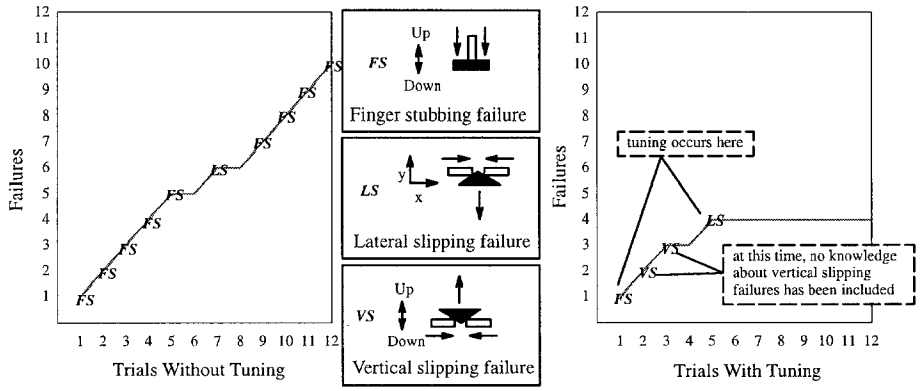


Figure 37. Comparison of Tuning to Non-tuning in Grasping the Pieces of a Puzzle

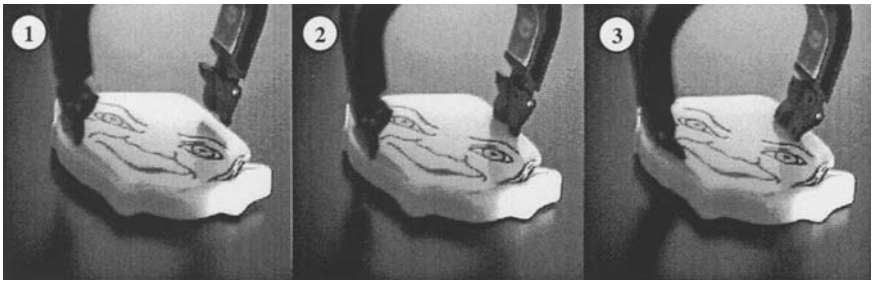


Figure 38. An Instance of a Finger Stubbing Failure

tuning turned off. The results are illustrated in Figure 37. Failures observed during this run included *finger stubbing failures* (see Figure 38) where a gripper finger struck the top of the object while moving down to surround it and *lateral slipping failures* (see Figure 39) where, as the grippers were closed, the object slipped out of grasp, sliding along the table surface. In the system's initial approximate representation for the world, the choice of grasping faces is constrained only by the gripper being able to open wide enough to surround them and that an equilibrium grasp is realizable with the current gripper-object friction coefficient (initially 1 here). Since a friction coefficient of 1 is likely to be high for these materials, the choice of contact faces is likely to be under-constrained initially, resulting in slipping failures. The choice of opening width is the minimum deviation from the current opening width (initially 0 here) which satisfies the approximate model of the grasp. Due to uncertainties in the world, this approximate opening width may often result in stubbing failures. Therefore, the error rate of our initial approximate plan was high resulting in 9 finger stubbing failures and 1 lateral slipping failure in 12 trials.

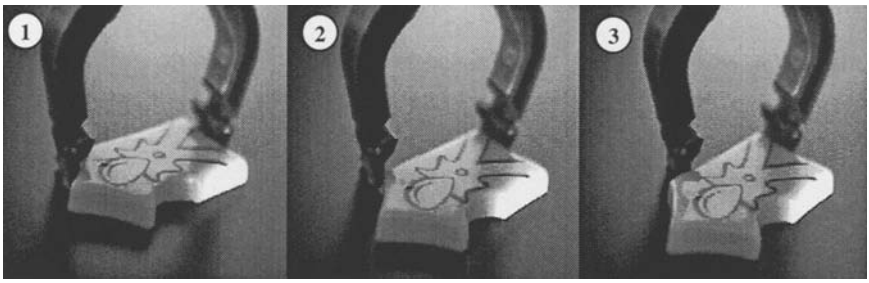


Figure 39. An Instance of a Horizontal Slipping Failure

In our second run, parameter tuning was turned on. An initial stubbing failure on trial 1 led to a tuning of the chosen-opening-width parameter which determines how far to open for the selected grasping faces. Since the generated qualitative tuning explanation illustrated that opening wider would decrease the chance of this type of a failure, the system tuned the parameter to choose the largest opening width possible (constrained only by the maximum gripper opening and possible collisions with nearby objects). In the case of isolated grasp targets, opening to the maximum gripper width is preferred. In trials 2 and 3, finger stubbing failures didn't occur as they had previously because the opening width was greater than the object width for that orientation. Vertical slipping failures, which the current implementation does not currently have knowledge about, did occur. The system had to be told that a vertical slipping failure had occurred instead of the lateral slipping failure it thought had occurred. This is because, without further knowledge about vertical slipping failures and a means for detecting them, they look in other ways (the force vs. position profile of the gripper closing) like a lateral slipping failure. Preventing vertical slipping failures involves knowing shape information along the height dimension of the object, which we plan to give in the future using a model-based vision approach. In trial 5, a lateral slipping failure is seen and the qualitative tuning explanation suggests decreasing the contact angle between selected grasping surface through tuning the contact-angle-constraint parameter. This is tuned to prefer smaller contact angles. A single tuning for the finger stubbing and lateral slipping failures was sufficient to eliminate those failures with isolated grasp targets.

5. Conclusions & Future Work

In the work discussed here, an assumption is made about limited interactions between plane parameters. We plan to eliminate this assumption by extending the method to learn about interactions between parameters. The illustration shows a view from above of two objects in the robot's workspace. The black rectangles are the gripper fingers as seen from above. The goal is to grasp object 1. The fingers are positioned along the pair of faces chosen for this grasp. Factors like the angle between the fingers and faces and friction coefficient will be the same no matter where the gripper contacts the object along that particular pair of faces.

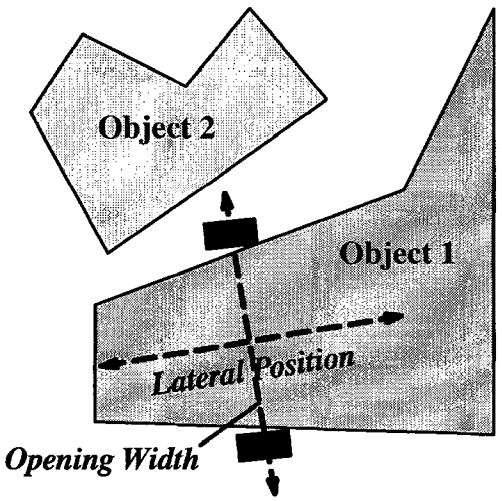


Figure 40. An Example of a Parameter Interaction: Lateral Position and Opening Width in the Robotic Grasping Domain

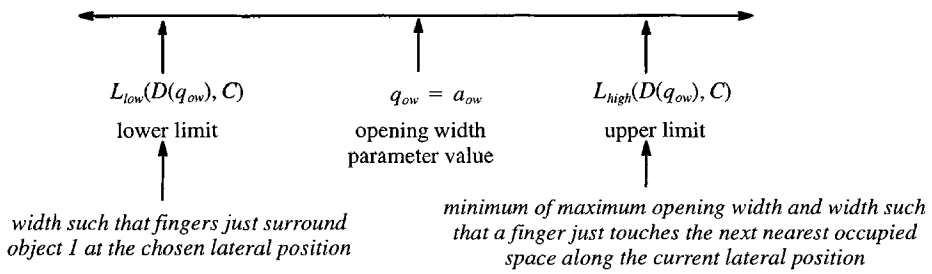


Figure 41. A Graphic Representation of Limits on Opening Width

However, the lateral position along the pair of faces affects the choice of opening width both because of the changing width of object 1 as one moves laterally and because of the nearby object 2. The upper and lower bounds on the opening width parameter depend on the lateral position parameter. These are illustrated in Figure 41. Suppose that the plan constructed by the system chooses a value for the lateral position parameter prior to choosing a value for opening width. Under the limited interaction assumption made currently this assumes that a lateral position is chosen such that $L_{low}(D(q_{ow}), C) \leq L_{high}(D(q_{ow}), C)$ and therefore a possible opening width is guaranteed. For a planner to guarantee this assumption requires exhaustively reasoning about all interacting parameters, an expensive process. It makes sense to learn incrementally about interactions which lead to failures. Instead of spending the time to guarantee no interactions, our initial plan could choose a lateral position without regard to opening width. Now suppose a lateral position was chosen such that the width of

object 1 at that point was wider than the maximum opening width of the gripper. In order to prevent trouble with the interaction, the constraint on opening width would be imposed on the choice of lateral position. The savings of an incremental approach to management of interactions is based on the assumption that many interactions, although theoretically possible are not likely to arise in practice. For instance, suppose all the objects in the workspace tend to have a maximum dimension well within the maximum opening width of the gripper and are well separated from nearby objects. In this case, the lateral position vs. opening width interaction is not likely to be a factor.

Another interesting area for study is the possibility of relaxing the conditions for generation of a qualitative tuning explanation. Guaranteeing soundness of such explanations can be costly. The requirements for generating qualitative proportionalities (Q_+ and Q_-) could be relaxed to permit transitivity of influences. For example, from $Q_+(a,b)$ and $Q_+(b,c)$ we could conclude $Q_+(a,c)$. Although such rules can make qualitative proportionalities easier to derive, they can introduce ambiguities. That is, it may be possible to show both $Q_+(a,b)$ and $Q_-(a,b)$. Qualitative tuning explanations (QTEs) would then be more heuristic than guaranteed methods of demonstrating a way to increase the probability of success of the failed predicates. However, such heuristic QTEs could be produced very quickly. The value of these heuristic QTEs would be tested empirically. Further heuristic QTEs could be produced if initial ones fail in practice. This approach is very much in the flavor of plausible EBL (DeJong, 1989).

The current approach assumes that every failure is worth preventing. This is not necessarily true. If we had reason to believe that a failure encountered was very unlikely, it may not be in our interest to attempt to tune parameters to prevent it. First, tuning parameters does require some effort which may outweigh the possible gain in tuning the unlikely failure. Second, tuning parameters to prevent the failure may involve some cost in terms of the modified plan. For instance, the parameter tuning may incur extra motion of a manipulator which results in slower execution of the overall plan. One possible strategy for deciding what failures are worth tuning is to empirically perform initial testing in the domain to discover what failures occur most frequently. Another approach is to develop a weak theory of errors which can be used to justify which failures are worth tuning.

When a newly learned preference is inconsistent with the existing preferences for some plan parameter, it becomes necessary to distinguish the context in which the conflicting preference occurred from the current context. We call this *context splitting*. A new set of preferences can then be created with the new preference added and the conflicting preference removed which is applicable to the current context (as distinguished from the context attached to the original set). One method of distinguishing contexts is to ask the user (Rosenbloom, Laird & Newell, 1987). We would like to have the system make the distinction using its domain knowledge. The knowledge is available all along but, without experience in the world, the system has insufficient experience to conclude which distinctions are important. In our framework, inconsistent preferences trigger the process of context splitting.

Notes

1. While experiments have not been performed with incorrect qualitative theories it seems that the technique would work in spite of some types of erroneous qualitative theories while it would result in poor performance with others.
2. For a model of the different aspects of utility for a plan to be executed in uncertain, complex domains see (Bennett, 1989).
3. Q_+, Q_-, and the associated inference rules about increasing and decreasing quantities are used as in Qualitative Process Theory (Forbus, 1984).
4. The subproof supporting the expectations is packaged into a single general rule prior to generation of the qualitative tuning explanation.

References

Agre, P. & Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proceedings of the National Conference on Artificial Intelligence*, pages 268–272.

Bennett, S.W. (1989). Learning uncertainty tolerant plans through approximation in complex domains. Master's thesis, University of Illinois at Urbana-Champaign.

Brooks, R.A. (1982). Symbolic error analysis and robot planning. Technical Report 685, MIT AI Lab.

Brooks, R.A. (1987). Planning is just a way of avoiding figuring out what to do next. Technical Report 303, MIT AI Lab.

Chapman, D. (1987). Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–378.

Davis, E. (1986). *Representing and Acquiring Geographic Knowledge*. Morgan Kaufmann Publishers, Inc.

DeJong, G.F. (1989). Explanation-based learning with plausible inferencing. In *Proceedings of The Fourth European Working Session on Learning*, pages 1–10.

DeJong, G.F. & Mooney, R.J. (1986). Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176.

Erdmann, M. (1986). Using backprojections for fine motion planning with uncertainty. *International Journal of Robotics Research*, 5(1):19–45.

Firby, R.J. (1987).. An investigation into reactive planning in complex domains. In *Proceedings of the National Conference on Artificial Intelligence*, pages 202–206.

Forbus, K.D. (1984). Qualitative process theory. *Artificial Intelligence*, 24:85–168.

Hammond, K., Converse, T. & Marks, M. (1990). Towards a theory of agency. In *Proceedings of the Workshop on Innovative Approaches to Planning Scheduling and Control*, pages 354–365.

Hutchinson, S.A. & Kak, A.C. (1990). Spar: A planner that satisfies operational and geometric goals in uncertain environments. *Artificial Intelligence*, 11(1):30–61.

Hwang, Y.K. (1988). *Robot Path Planning using Potential Field Representations*. PhD thesis, University of Illinois at Urbana-Champaign.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98.

Lozano-Perez, T., Mason, M.T. & Taylor, R.H. (1984). Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24.

Malkin, P.K. & Addanki, S. (1990). Lognets: A hybrid graph spatial representation for robot navigation. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 1045–1050.

Mitchell, T.M., Keller, R. & Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, January 1986.

Mitchell, T.M., Mahadevan, S. & Steinberg, L.I. (1985). Leap: A learning apprentice for vlsi design. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 573–580.

Mooney, R.J. & Bennett, S.W. (1986). A domain independent explanation-based generalizer. In *Proceedings of the National Conference on Artificial Intelligence*, pages 551–555.

Rosenbloom, P.S., Laird, J.E. & Newell, A. (1987). Knowledge level learning in soar. In *Proceedings of the National Conference on Artificial Intelligence*, pages 499–504.

Schoppers, M.J. (1987). Universal plans for reactive robots in unpredictable environments. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1039–1046.

Segre, A.M. (1988). *Machine Learning of Robot Assembly Plans*. Kluwer Academic Publishers.

Suchman, L.A. (1987). *Plans and Situated Actions*. Cambridge University Press.

Wong, E.K. & Fu, K.S. (1985). A hierarchical orthoganal space approach to collision-free path planning. In *Proceedings of the 1985 Institute of Electrical and Electronics Engineers International Conference on Robotics and Automation*, pages 506–511.

Zadeh, L.A. (1965). Fuzzy sets. *Information and Control*, 8(3):338–353.

Zhu, D.J. & Latombe, J.C. (1990). Constraint reformulation in a hierarchical path planner. In *Proceedings of the 1990 Institute of Electrical and Electronics Engineers International Conference of Robotics and Automation*, pages 1918–1923.