some knowledge of the purpose that the actions serve - i.e. the goals that they are to accomplish. For these reasons, we have chosen to use a a domain independent planning approach at the top level, and add lower level components to deal with domain specific issues relevant to robotic assembly planning.

2. RELATED RESEARCH

Research applicable to robotic assembly planning can be broadly divided into two groups: research on problems that are specifically related to robotic assembly and research in domain independent planning. The former class includes planners that generate task sequences [10,11], planners that automatically derive geometric specifications of assemblies [1,14], motion planners [8,13,19], and error recovery planners [4,9,17]. The latter group contains the classical AI planners, where the emphasis is placed on the domain independent aspects of the reasoning process used to develop plans [5,18,22]. Neither of these approaches has yet produced a planner capable of creating complete assembly plans from high level specifications of assembly goals. The domain independent planners lack the ability to reason about geometric concerns and uncertainties in the work cell, while the task specific planners typically have a narrow focus, and deal only with limited aspects of assembly planning without any understanding of assembly goals or the effects of actions.

There are two systems, of which we are aware, with a somewhat larger scope than those listed above: TWAIN [15], and Handey [16]. Each of these planners begins with a high level task plan and then adds motion plans for the individual actions in that plan. TWAIN is a constraint posting planner which can also add sensory operations to the plan to reduce uncertainties. Handey is an integrated system which includes a sensory system to determine the initial world state. One of Handey's main strengths is its ability to plan grasping operations when the objects are in cluttered environments (this is discussed further in [21]). It should be noted, however, that neither TWAIN or Handey is capable of reasoning about the effects of actions and how they might be used to achieve goals (this not being necessary since both start with an initial task plan).

SPAR combines domain independent planning techniques with a number of modules containing task specific knowledge. SPAR uses a nonlinear constraint posting algorithm for its top level control structure, while domain specific knowledge is used by the constraint manipulation system (CMS) to evaluate lower level constraints during planning. Using a constraint posting approach, SPAR seeks to satisfy a goal by first examining all of the actions and constraints previously generated to see if the goal can be satisfied merely by adding a new constraint (where a constraint may be viewed as a specification or a restriction on an action). If this strategy fails, a new action is added to the plan. Nonlinear planning allows actions to be added to the plan without imposing a strict ordering on the set of actions.

One of the primary drawbacks of traditional domain independent planners (e.g. SIPE [22] and TWEAK [5]) is that the representations used must be domain independent. Because of this, these planners use high level symbolic constructs to represent the world, and the effects of actions. For example, both SIPE and TWEAK use well formed formulas (wff's) from predicate calculus to represent the effects of actions. In a complex domain, such as robotic assembly, this type of high level representation is not sufficient to represent all relevant aspects of the world. For example, while it is possible to represent a number of spatial relationships with wff's (e.g. on(block1,block2)), there would be no way to describe the reachable configurations of the robot arm with such a high level representation, since the reachable configurations are defined by a subset of an N-dimensional continuous space (for a robot with N joints). Therefore, while SIPE easily solves high level blocks world problems (such as, put some blue block on the top of some green block), it is not capable of solving the lower level details of these problems (e.g. determining the joint angles which must be used to position the robot arm to perform the stacking operation).

3. PLANNING IN SPAR

Spar extends traditional constraint posting planners, such as those described in [5,22], to include both geometric planning and uncertainty-reduction planning. By geometric planning, we mean the planning that determines the actual geometric configurations that will be used during the assembly process. These configurations include the configurations of the manipulator, the positions in which parts are placed, and the grasping configurations which are used to manipulate objects. Uncertainty-reduction planning consists of first determining whether or not the uncertainty in the planner's description of the world (e.g. the possible error in part locations) is sufficiently small to allow plan execution to succeed. If the uncertainties are too large, then either sensing operations or manipulations are added to the plan in an attempt to reduce the uncertainty to an acceptable level. If this fails, verification sensing and local recovery plans are added to the plan. These can be used during plan execution to monitor the robot's success and recover from possible run time errors.

When designing a nonlinear constraint posting planner, the degree to which constraint posting is used is an issue that must be considered. A pure constraint posting planner would make no unnecessary variable instantiations until all of its goals had been satisfied, at which time the CMS would determine the variable instantiations that simultaneously satisfied all of the constraints. The advantage to this approach is a decrease in the the amount of backtracking by avoiding arbitrary choices that could lead to failure. The disadvantage to a pure constraint posting approach is that maintaining the constraints can become more expensive than backtracking during planning. Therefore, in many cases a combination of constraint posting and backtracking is appropriate, the exact combination being determined by the complexities of the constraints and the cost of backtracking.

In SPAR, due to the complexities involved with the representation and evaluation of uncertainty-reduction goals, only the operational and geometric goals are satisfied using the constraint posting method. Therefore, SPAR performs its planning in two phases. In the first phase constraint posting is used to construct a family of plans that satisfy all operational and geometric goals. In the second phase, specific plan instances (generated by instantiating the plan variables so that the constraints are satisfied) are used as input for the uncertainty-reduction planning. We should note that the constraint posting paradigm is conceptually able to handle all three goal types, however, for the reasons of complexity that we have just mentioned, it is not expedient to force uncertainty-reduction planning into the constraint posting mold. Furthermore, it would not be advantageous to abandon constraint posting for the operational and geometric planning, since the cost of maintaining the constraints associated with these two types of goals is significantly less than the cost of a backtracking search algorithm.

In the first phase of planning, SPAR iteratively refines a partial plan to satisfy some pending goal. This is done by either constraining the execution of an action that is already in the plan, or by introducing a new action into the plan. In the latter case, SPAR adds the new action's preconditions to appropriate goal stacks, and also checks each currently satisfied goal, noting those which are possibly undone by the new action and placing them on the appropriate goal stack. The first phase of planning terminates when there are no more pending operational or geometric goals.

In the second phase of planning, the uncertainty-reduction preconditions are considered for specific plan instances. In order to create these plan instances, SPAR invokes its CMS to find consistent solutions for the plan's constraints. These solutions are then used to instantiate the variables in the plan actions. Specific plan instances are examined until one is found in which all uncertainty-reduction goals can be satisfied. If no such instance can be found, the instance that contained the fewest unsatisfied uncertainty-reduction goals is selected. This plan instance is augmented with sensing verification actions and potential recovery plans for anticipated possible errors.

Fig. 1 shows a block diagram of SPAR. To the left are the goal stacks and satisfied goals, used to keep track of planning progress. The dashed box at the top of the figure represents SPAR's knowledge about actions. This includes templates that represent actions, a set of rules for instantiating those templates, a set of actions that reduce uncertainty in the world, and a set of procedures used to construct the uncertainty-reduction preconditions for actions. To the right, enclosed by a dashed box, is the constraint system. This includes the CMS and a number of domain dependent modules used to evaluate constraints. These modules include routines to find upper and lower bounds on symbolic expressions (SUP/INF), an algebraic simplifier (SMP), and routines to solve the inverse kinematics of the robot (IKS). The constraint system also includes a constraint network, used to organize the plan's constraints. Finally, the bottom of the figure depicts the output of the planner: the verification sensory operations and local recovery plans (used when uncertainty-reduction goals cannot be satisfied), and the set of actions in the plan.

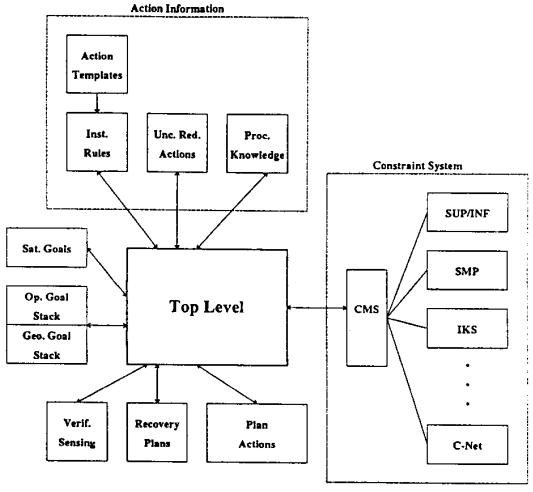


Fig. 1: Architecture of SPAR.

4. REPRESENTATIONAL ISSUES IN SPAR

One of the issues that must be addressed when designing a planning system is the choice of representation schemes. These representations determine the power that the planner will have, in terms of its ability to adequately model the world and the possible actions that can be performed to alter the world. In this section, we will describe how SPAR represents actions, uncertainty, and goals.

4.1. Representation of Actions

Currently, SPAR plans with three actions: pickup, putdown and assemble. These actions are represented by action templates, each of which has the following components.

Action id: An identifier to reference a particular instance of the action.

Action: The name of the action, and its arguments.

Preconditions: The operational geometric, and uncertainty-reduction preconditions of the action.

Add list: A list of conditions true in the world after the execution of the action.

Delete list: A list of conditions no longer true in the world after the execution of the action.

action-id: ActionId,

action: pickup(Object,Grasp),

preconditions:

operational:

op(G1,ActionId, gripper(open))
op(G2,ActionId, part_location(Object,Pos))

geometric:

geo(G3,ActionId,
reachable(Grasp,Pos),
part_location(Object,Pos))

uncertainty-reduction:

$0 < [0,1,0,0]\, P_1^{-1}\, C_1$
$0 > [0,1,0,0]\, P_1^{-1}\, C_2$
$0 < [0,1,0,0]\, P_2^{-1}\, C_1$
$0 > [0,1,0,0]\, P_2^{-1}\, C_2$

add-list:

holding(Object,Grasp)
part_location_unc(Object,NewUnc)
gripper(closed)

delete-list:

part_location(Object,Pos)
part_location_unc(Object,OldUnc)
gripper(open)

Fig. 2: Action template for the pickup action.

As an example, Fig. 2 shows the action template for the pickup action.

When SPAR adds an action to the plan, it instantiates the template for that action to accomplish the particular goal that caused the action's addition. This consists of first instantiating the various identifiers in the action to unique labels (e.g. the ActionId the Gi's), and then either instantiating or constraining the plan variables in the action so that it achieves the goal. SPAR uses a set of rules to determine the proper variable instantiations for an action template, given the goal which the action is to achieve. By using this approach to instantiating action templates, SPAR is able to use a small set of generic robot operations and instantiate these to specific actions based on the objects to be manipulated by those actions.

4.2. The Representation of Uncertainty

In order to create assembly plans to be executed in an uncertain environment, SPAR requires a representation for the uncertainty in its world description, an understanding of how much uncertainty in that description can be tolerated before an action can no longer be guaranteed to succeed, and a knowledge of how the various assembly actions affect the uncertainty in the world description. In this section, we will address each of these three issues.

4.2.1. Representing Uncertain Quantities

In our current implementation of SPAR, we have chosen to limit the number of quantities considered uncertain. For an object resting on the work table, the X,Y,Z location of the object (i.e. the object's displacement) and the rotation about an axis through the origin of the object's local frame and perpendicular to the table are considered uncertain. This choice reflects our assumption that objects resting on the work table will be in a stable pose, which fixes two rotational degrees of freedom of the object. For the manipulator, we consider the X,Y,Z location of the tool center, and the rotation about the Z

*There is only a limited repertoire of actions that can be carried out by a single robot arm and the three listed here represent those that are used most often. Actions like threading and fixturing could be considered to be more specialized forms of the assemble action presented here, the specialized forms being obtained by the addition of more geometric and uncertainty-reduction constraints.

axis of the manipulator's local frame (i.e. the approach axis of the manipulator) to be uncertain.

All uncertainties in SPAR are expressed in terms of uncertainty variables. The possible values for an uncertainty variable are defined using bounded sets. We represent the uncertainty in the position of an object by a homogeneous transformation matrix whose entries are expressed in terms of uncertainty variables. By combining the ideal position of an object (i.e. the position of the object if all uncertainty is eliminated) with the uncertainty in that position, we obtain the possible position of an object. This possible position will be a homogeneous transformation matrix, with some or all of its entries expressed in terms of uncertainty variables. A matrix obtained by substituting valid values for the uncertainty variables will represent one possible position of the object.

As an example, we define the transformation which represents the uncertainty in the position of the manipulator relative to the manipulator's local frame to be:

$$T_{\Delta M} = \begin{bmatrix} \cos(\Delta\theta_g) & -\sin(\Delta\theta_g) & 0 & \Delta X_g \\ \sin(\Delta\theta_g) & \cos(\Delta\theta_g) & 0 & \Delta Y_g \\ 0 & 0 & 1 & \Delta Z_g \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Again, note that the values ΔX_g, ΔY_g, ΔZ_g, and $\Delta\theta_g$ are bounded symbolic variables. Therefore, the matrix $T_{\Delta M}$ represents all of the transformations that could be obtained by substituting valid numerical values into the matrix in place of the symbolic variables. The bounds on these variables are stored in SPAR's database, and retrieved as needed.

4.2.2. Derivation of Uncertainty-Reduction Goals

The uncertainty-reduction preconditions of an action are symbolic expressions that define the maximum uncertainty in the world description under which the action can still be reliably executed. As an example, consider the uncertainty-reduction preconditions for the pickup action. In order for the manipulator to grasp an object, the two contact points on the object must lie between the fingers of the manipulator, even when worst case uncertainties occur. This is illustrated in Fig. 3. The possible locations of the manipulator fingers are given by

$$P_{1,2} = T_{M+\Delta}\,\mathrm{trans}(0, \pm\tfrac{1}{2}W_m, 0) \quad (2)$$

where $T_{M+\Delta}$ represents the combination of the ideal position of the manipulator with the error transformation (shown in EQ. 1), trans(x,y,z) represents a pure translation, and W_m is the width of the manipulator opening. The possible positions of the contact points of the object are given by:

$$C_{1,2} = Tr_O\, T_{\Delta O}\, R_O\, T_G\, \mathrm{trans}(0, \pm\tfrac{1}{2}W_g, 0)[0,0,0,1]^t \quad (3)$$

where Tr_O is the x,y,z position of the origin of the block's local frame, $T_{\Delta O}$ is represents the uncertainty in the position of that frame, R_O is the orientation of the block's frame, T_G represents the position of the manipulator relative to the position of the block's frame (i.e. T_G is the grasping transformation), and W_g is the width of the block at the grasp points.

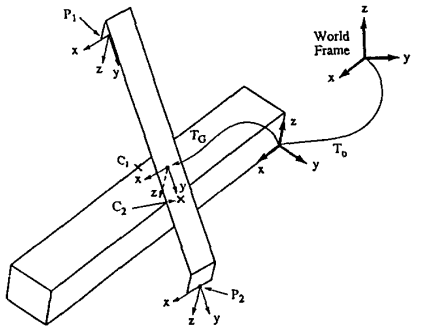


Fig. 3: Possible locations of fingers and contact points.

To test whether the contact points lie between the fingers, we transform the locations of C_1 and C_2 to be defined in terms of the coordinate frames P_1 and P_2, and check to see that the Y-components of these locations are on the positive Y axis for P_1 and on the negative Y-axis for P_2, for all possible values of the uncertainty variables. Therefore, the four uncertainty-reduction preconditions for the pickup action are:

$$0 < [0,1,0,0]\,P_1^{-1}\,C_{1,2}\,, \quad 0 > [0,1,0,0]\,P_2^{-1}\,C_{1,2} \quad (4)$$

Again, note that all of the matrix multiplications shown above must be performed symbolically, since many of the entries in the matrices will be expressed in terms of uncertainty variables. More detailed descriptions of SPAR's uncertainty-reduction preconditions can be found in [12].

4.2.3. The Propagation of Uncertainty by Actions

Actions propagate uncertainties in distinct ways. For example, the pickup action has the effect of transforming an object's displacement uncertainty into the manipulator coordinate frame, and then reducing the Y component of this uncertainty to the uncertainty in the Y component of the manipulator. The pickup action also reduces the uncertainty in the object's orientation to be equal to the uncertainty in the orientation of the manipulator (this assumes that the uncertainty in the position of the object was initially greater than the uncertainty in the position of the manipulator). The uncertainty in the position of the block while it is in the grasp of the manipulator is represented by:

$$T_{\Delta O} = \begin{bmatrix} \cos(\Delta\theta_g) & \sin(\Delta\theta_g) & 0 & Dx + \Delta X_g \\ \sin(\Delta\theta_g) & \cos(\Delta\theta_g) & 0 & \Delta Y_g \\ 0 & 0 & 1 & Dz + \Delta Z_g \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

where Dx and Dz represent the initial uncertainty in the x and z positions of the block, ΔX_g, ΔY_g and ΔZ_g represent uncertainty in the position of the manipulator and $\Delta\theta_g$ represents the uncertainty in the orientation of the manipulator. Note that the uncertainty in the Y component of the displacement uncertainty has been limited to the uncertainty in the Y component of the location of the manipulator's tool center. Further, note that the rotational uncertainty is the same as the rotational uncertainty in the orientation of the manipulator.

A more detailed description of how SPAR's actions propagate uncertainty can be found in [12]

4.3. Operational and Geometric Goals

Goals in SPAR have three relevant attributes: a type (either operational, geometric or uncertainty-reduction) a condition that must be satisfied (i.e. the actual goal) and an action identifier. The action identifier is used to indicate when the goal must be satisfied, in particular, that it must be satisfied prior to the execution of the action specified by the action identifier. We will use the terms goal and precondition to refer to either the condition part of the goal or to the entire structure. Which of these is meant should be evident from the context.

SPAR's operational goals are similar to the high level goals used in traditional domain independent planners (e.g. STRIPS or TWEAK). One difference is our inclusion of plan variables that can be used to link operational and geometric goals. For example, one operational precondition of the assemble action is:

op(G1, ActionId, holding(Obj1, Grasp))

The plan variable Grasp is not used in the operational planning, but serves the purpose of linking operational and geometric planning. The variable ActionId is used to indicate the time at which the goal must be satisfied. In particular, it must be satisfied prior to the execution of the action whose action identifier is ActionId.

Geometric goals are slightly more complex, with two main components. The first is a geometric constraint and the second

is a set of operational goals. The meaning of this pair is that the planner is to establish the operational goals in such a way that the geometric constraint is satisfied. For example, one geometric precondition of the putdown action is:

```
geo(G2, ActionId,
    reachable(Grasp, Pos),
    holding(Obj, Grasp))
```

where holding(Obj,Grasp) is the single operational goal, and reachable(Grasp,Pos) is the geometric constraint. This goal indicates that the grasp used to achieve the holding goal must also permit the manipulator to reach the destination position.

5. GOAL SATISFACTION

In this section, we will individually discuss the methods used to satisfy operational, geometric and uncertainty-reduction goals. In the course of this discussion, we will frequently allude to the CMS's role in the process of goal satisfaction, however, we will leave a discussion of the CMS for Section 6. For the purposes of this section, it is sufficient to assume that the CMS is capable of determining if a new constraint is consistent with the current constraint set.

5.1. Satisfying Operational Goals

Ensuring the satisfaction of an operational goal proceeds in two steps, as described in [5]: finding an action that establishes the goal and then dealing with actions that could violate (or undo) the goal.

In order to find an action that establishes an operational goal, SPAR first looks at the add lists of the actions already in the partially developed plan. If any element of the add list of such an action can be unified with the operational goal, then that unification is performed, and the action is declared to have established the goal. If such an action is found, it is constrained to take place prior to the time at which the goal must be satisfied. If the CMS determines that this new ordering constraint is not consistent with the current ordering constraints, the constraint addition fails and SPAR backtracks in an attempt to find another action in the plan to establish the goal.

If no action in the plan is found to establish the goal, SPAR adds a new action. This consists of instantiating an action template, adding the action to the plan, and constraining the new action to occur prior to the time at which the goal must be satisfied. Any time SPAR adds an action to the plan, it is possible that the new action may violate goals that have already been satisfied. For this reason, when a new action is added to the plan, SPAR examines the list of satisfied goals and transfers any of these that could be violated by the new action to the appropriate pending goal stack.

Once an operational goal has been established, SPAR examines each action in the current partial plan to see if it could possibly violate the goal. An action can violate an operational goal if any element in the action's delete list can be unified with the goal. There are two ways SPAR deals with a potential goal violation: the violating action can be constrained to occur after the time at which the goal must be satisfied (promotion of the goal); an action can be used to re-establish the goal. A re-establishing action can either be an action already in the plan, or it can be a new action which is added specifically for the purpose of re-establishing the violated goal.

5.2. Satisfying Geometric Goals

Geometric goals are satisfied by constraining the way plan actions are performed. For example, if a geometric goal specifies that the manipulator should be holding an object in a particular grasping configuration, the way to satisfy that goal is to place a constraint on how the manipulator performs the grasping action. In order to do this, SPAR needs to link together the operational and geometric levels of planning. For this purpose, when planning to satisfy operational goals, plan variables are introduced that can be constrained by the geometric level of planning to determine how an action is executed. The geometric preconditions are expressed in terms of those variables. For example, a traditional STRIPS type action is pickup(Object). SPAR's equivalent action is pickup(Object,Grasp). The variable Grasp is used to define the geometric configuration that will be used by the manipulator to grasp the object. At the operational level, the variable Grasp is primarily ignored, but its presence gives SPAR a method of constraining how the pickup operation is actually performed, thus linking distinct levels of planning.

As was discussed earlier, geometric goals consists of a set of operational goals and a geometric constraint to be applied to the actions that achieve the operational goals. Each operational goal that is associated with a geometric precondition of an action is also listed separately as an operational precondition of the action. Therefore, since SPAR only considers geometric goals when the operational goal stack is empty, the operational goals associated with a geometric goal are guaranteed to be satisfied by the current partial plan. Therefore, in order to satisfy a geometric goal, SPAR first finds the actions that establish its associated operational goals, and attempts to constrain the execution of those actions so that the geometric constraint is satisfied. This is done by instructing the CMS to add the geometric constraint to the plan. If this succeeds, the goal is satisfied and moved to the list of satisfied goals.

If the CMS determines that the geometric constraint is not consistent with the current constraints, then one or more new actions must be added to the plan. These new actions are chosen based on the operational goals associated with the geometric goal. Once the actions have been added, the appropriate geometric constraint is also added. This constraint will be consistent with the constraints currently in the plan, since the new action will contain new plan variables that have not yet been constrained. Note that the addition of actions to the plan will introduce new operational goals, and therefore effectively transfer control back to operational planning.

There is no need for SPAR to check for actions that might violate geometric constraints. The reason for this is that the set of constraints has no sense of temporal ordering. All constraints must be consistent at all times. Therefore, if any constraint had the effect of violating the new geometric constraint, this would have been detected by the CMS when attempting the constraint addition.

5.3. Satisfying Uncertainty-Reduction Goals

When there are no remaining operational or geometric goals, SPAR begins planning to satisfy uncertainty-reduction goals. As mentioned earlier, uncertainty-reduction planning does not use a constraint posting approach. Instead, specific plan instances are generated and tested for satisfaction of the uncertainty-reduction goals.

To test a particular plan instance, SPAR creates an augmented plan instance consisting of four components: the instantiated list of plan actions (obtained by instantiating the actions from the partial plan that was developed in the first phase of planning so that all constraints are satisfied), an error count (initially set to zero), a list of sensory verification actions (initially set to the empty list), and a list of local error recovery plans (also initially set to the empty list). SPAR then sequentially examines each individual action in the instantiated action list and attempts to satisfy its uncertainty-reduction preconditions After an action has been considered, its add and delete lists are used to update the world state to reflect the effects of the action. The uncertainty in the world description is also propagated forward, thereby defining the uncertainty in the world when the next action in the sequence will be executed.

There are three ways to satisfy an uncertainty-reduction goal for an individual action: the goal may be satisfied by the current set of constraints on the uncertainty in the world description; sensing actions can be added to the plan to reduce uncertainties; manipulations can be added to the plan to reduce uncertainties. If SPAR fails to satisfy the goal, it prepares for possible execution errors. The error count for the augmented

plan instance is incremented and sensing verification actions and local recovery plans are added to their respective lists in the augmented plan instance.

6. CONSTRAINT MANIPULATION

In SPAR, the bulk of the domain knowledge resides in the constraint manipulation system. This allows the planning algorithm to proceed without any need to "understand" the domain of robotic assembly. The action descriptions include preconditions on geometric configurations and the tolerable uncertainties in the world description, but in order to satisfy these preconditions, the top level planner merely requests that the CMS add constraints to the constraint database. It is the task of the CMS to determine whether or not these new constraints are consistent with the current constraints in the plan, which in turn, requires a certain amount of domain specific knowledge.

SPAR currently uses three constraint types. In operational planning, ordering constraints are used to ensure that actions are performed in the proper sequence (and that goals are satisfied at the appropriate times). In geometric planning, binary constraints between object positions and manipulator configurations are used to ensure that the robot will be able to perform the required manipulations. Finally, at the uncertainty-reduction level, symbolic algebraic inequalities are used to express the maximum uncertainty that can exist in the world description prior to the execution of an action. A directed graph is used to represent ordering constraints, a binary constraint network for the geometric constraints, and algebraic inequalities (expressed in terms of bounded symbolic variables) for the uncertainty-reduction constraints.

Using a directed graph to represent ordering constraints, an arc directed from a node, A, to another node, B, implies that action A must precede action B in the plan. Ordering constraints are added to the plan by adding appropriate arcs to the ordering graph. A cycle in the ordering graph indicates an inconsistent set of ordering constraints.

All of the geometric constraints in SPAR are either binary constraints between plan variables representing object positions and manipulator positions, or unary constraints on plan variables. Furthermore, both object poses (i.e. possible orientations of objects, not including displacement information) and grasping configurations have been quantized, and assigned labels, so that each of these can be represented by a single, symbolic variable rather than a continuous variable in six dimensional space. Because of these qualities, it is straightforward to represent the geometric constraints using a binary constraint network. By using a binary constraint network, when the CMS is instructed to add a new constraint, the consistency of that constraint with the current set of constraints can be determined by adding an arc to the constraint network and then checking the new network for consistency. Because of space limitations, we will not give an introduction to constraint networks here. A thorough introduction can be found in either of [6,7].

In order to represent SPAR's geometric constraints using a binary constraint network, each geometric plan variable (e.g. grasp configurations, positions) is represented by a node in the network. When a new variable is introduced into the plan, a node is added to the network and assigned an initial label set. This label set is merely the set of values which may be assigned to that variable. For example, if the variable represents a grasping configuration for a particular object, then the initial label set for its node in the constraint network will contain the labels of all of the grasping configurations for that object.

Unary constraints on plan variables are achieved by limiting the label set for the corresponding node in the constraint network. In this way, the set of values that a particular geometric parameter may be given can be restricted (e.g. certain grasping configurations can be excluded). Binary constraints between plan variables are represented by arcs between the corresponding nodes in the network (these arcs are not directed). Each arc in the network contains a set of pairs of values which indicate the valid pairs of labels for the connected nodes. Determining the valid pairs of labels requires a semantic understanding of the domain, but once the pairs have been assigned, no domain knowledge is required to check for network consistency.

When the CMS is instructed to add a unary constraint to the network, it first updates the label set of the appropriate node, and then updates each arc connected to that node by deleting pairs that are no longer valid given the node's new label set. Finally, the new network is checked for consistency. When the CMS is instructed to add a new binary constraint to the network, it adds an arc between the appropriate nodes (creating the nodes if they do not already exist in the network), and then checks for network consistency.

When the planner inserts a manipulation action into the plan, it must ensure that all of the configurations required to perform that manipulation will be physically realizable. In order to do this, SPAR uses reachability constraints. The two conditions that must be met to satisfy reachability are that the manipulation be within the physical capabilities of the robot (this is verified using the inverse kinematic solutions for the particular robot), and that grasps used in the course of the manipulation do not obscure mating features (those features that must come into contact with other objects during the course of the operation). A description of these how these constraints are derived can be found in [12]. When a reachability constraint is added to the geometric binary constraint network, all pairs of grasps/positions that satisfy the constraint are placed on an arc between nodes representing the planning variables for the grasp configuration and the position. A network consistency check is then performed.

When the planner considers the uncertainty-reduction goals, it does so for a particular instantiated plan instance. As a consequence of this, at the time of their evaluation, the uncertainty-reduction goals (expressed as symbolic algebraic inequalities) will be expressed in terms of specific bounded symbolic variables. Therefore, determining if an uncertainty-reduction goal is satisfied consists of a single evaluation (rather than a series of evaluations as is required for geometric constraints). In particular, since the uncertainty-reduction goals are expressed as inequalities of the form $\mathbf{expr}_1 < \mathbf{expr}_2$, and since at least one of these expressions is always a single constant, if we find the maximum value for $\mathbf{expr}_1$ and the minimum value for $\mathbf{expr}_2$ (under the constraints contained in the world description), we can determine whether the uncertainty-reduction goals are met simply be checking to see if $\max(\mathbf{expr}_1) < \min(\mathbf{expr}_2)$.

In order to find upper and lower bounds on symbolic expressions, we have implemented a system similar to the SUP/INF system which was introduced by Bledsoe [2], and then refined by Shostak [20], and later Brooks for his ACRONYM system [3]. The functions SUP and INF each take two arguments, a symbolic expression and a set of variables, and return upper/lower bounds on the expression in terms of the variables in the variable set. The method SUP/INF employs is to recursively break down expressions into subexpressions, find bounds on these subexpressions, and then combine the bounds using rules from interval arithmetic. Obviously this works for linear expressions where superposition holds. When expressions are nonlinear, however, it is quite possible that the bounds on the individual subexpressions will be looser than the bounds on the subexpressions when considered in the context of the whole expression. Because of this, it is possible that SUP/INF will sometimes find inexact bounds.

In spite of this, the policy of recursively finding bounds on subexpressions and then combining those bounds guarantees that the algorithms will terminate. This has been shown by Shostak for his version of SUP/INF, and later by Brooks for his modified versions. Furthermore, even though it is possible that SUP/INF will not return exact bounds, it has been shown (again, by Shostak and later by Brooks) that they are conservative, in that SUP always returns a value greater than or equal to the maximum, and INF always returns a value less than or equal to the minimum. The fact that SUP/INF sometime only approximates solutions is not a severe problem for SPAR, since

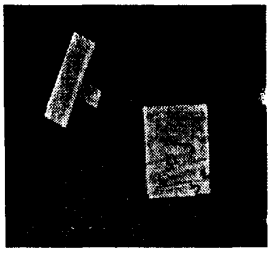
failure to satisfy uncertainty constraints has a worst case result of the addition of sensing actions to the plan. That is, if the CMS determines that the uncertainty constraints cannot be satisfied, it does not backtrack. It merely prepares for the possibility of failure.

7. EXPERIMENTAL RESULTS

We have used SPAR to plan a number of assembly tasks. The resulting assembly plans were then executed in a robot work cell vequipped with a PUMA 762 robot. The assembly goal in each experiment was to mate a block with a peg to a block containing a hole. By varying the initial positions of the objects, SPAR was forced to develop distinct plans for the individual tests, even though the assembly goals were the same.

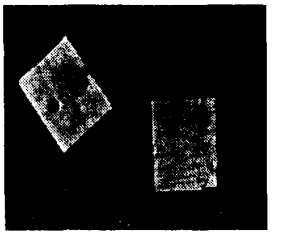
Figs. 4 and 5 illustrate experiments that have been performed. Each figure includes a photograph of the initial world situation and a listing of the plan that is output by SPAR. In Fig. 4, the block is initially face down and the peg is resting on its side. In order to perform the assembly, the robot must first reposition the block so that the face containing the hole is accessible. This requires two manipulations: one to place the block on its side (an intermediate position) and one to then place the block on its back. Once the block is repositioned, the peg is picked up and the assemble action is performed. Fig. 6 shows the robot executing the steps in the plan shown in Fig. 4.

We should note that, although the uncertainty-reduction planner is fully implemented, it is not yet integrated with the execution system. There are two reasons for this. First, we have not yet fully and accurately characterized the uncertainties in the sensors and robotic manipulator. Second, the current set of recovery plans lacks robustness and generality. These two areas are subject of ongoing research.



```
((1 (pickup-object block
        ((0.707107 0.0 -0.707107)
        (-0.0 1.0 0.0)
        (0.707107 0.0 0.707107)
        (1.0 1.375 1.625)))
 (2 (putdown-object block
        ((-1 0 0)
        (0 1 0)
        (0 0 -1)
        (-10 30 -18.05))))
 (3 (pickup-object block
        ((-0.707107 8.6593e-17 -0.707107)
        (-1.2246e-16 -1.0 0.0)
        (-0.707107 8.6593e-17 0.707107)
        (1.0 1.375 1.625))))
 (4 (putdown-object block
        ((0 0 1)
        (0 1 0)
        (-1 0 0)
        (-10 20 -21.3)))))
 (5 (pickup-object peg
        ((-0.707107 -0.707107 -6.123e-17)
        (8.6593e-17 6.163e-33 -1.0)
        (0.707107 -0.707107 6.123e-17)
        (0.5 1.375 1.625))))
 (6 (assemble-objects peg block
        ((-1 0 0)
        (0 1 0)
        (0 0 -1)
        (3 0 3.25))
        (-4 0 0))))
```

Fig. 4: SPAR generated plan to assemble the two objects shown. In the initial state, the block faces down and the peg points sideways.



```
((3 (pickup-object block
        ((0.707107 0.0 -0.707107)
        (-0.0 1.0 0.0)
        (0.707107 0.0 0.707107)
        (1.0 1.375 1.625))))
 (4 (putdown-object block
        ((-1 0 0)
        (0 1 0)
        (0 0 -1)
        (-10 30 -18.05))))
 (5 (pickup-object block
        ((-0.707107 8.6593e-17 -0.707107)
        (-1.2246e-16 -1.0 0.0)
        (-0.707107 8.6593e-17 0.707107)
        (1.0 1.375 1.625))))
 (6 (putdown-object block
        ((0 0 1)
        (0 1 0)
        (-1 0 0)
        (-10 20 -21.3))))
 (1 (pickup-object peg
        ((-0.707107 8.6593e-17 -0.707107)
        (-1.2246e-16 -1.0 0.0)
        (-0.707107 8.6593e-17 0.707107)
        (1.0 1.375 1.625))))
 (2 (putdown-object peg
        ((-1 0 0)
        (0 1 0)
        (0 0 -1)
        (-10 25 -21.3))))
 (7 (pickup-object peg
        ((0.707107 0.0 -0.707107)
        (-0.0 1.0 0.0)
        (0.707107 0.0 0.707107)
        (0.5 1.375 1.625))))
 (8 (assemble-objects peg block
        ((-1 0 0)
        (0 1 0)
        (0 0 -1)
        (3 0 3.25))
        (-4 0 0))))
```

Fig. 5: Same as Fig. 4, except now the peg initially points up.

8. CONCLUSIONS

This paper represents a step toward a planning system which can create assembly plans given as input a high level description of assembly goals, geometric models of the components of the assembly, and a description of the capabilities of the work cell (including the robot and the sensory system). The resulting planner, SPAR, reasons at three levels of abstraction: the operational level (where high level operations are planned), the geometric level (where geometric configurations of the actions are planned) and the uncertainty-reduction level (where world uncertainties are taken into account).

At the first two levels of planning, we have extended the constraint posting approach to domain independent planning by adding geometric preconditions to the actions, linking these to operational goals via plan variables, and expanding the CMS to be able to deal with geometric constraints. At the uncertainty-reduction level of planning, we have expressed uncertainties in the world in terms of homogeneous transformations whose elements are defined in terms of symbolic uncertainty variables. We then expressed limits on tolerable uncertainties in terms of operations on transformations. When the uncertainty-reduction goals cannot be satisfied, rather than abandon the plan, our system augments the plan with sensing operations for verification, and when possible, with local error recovery plans.

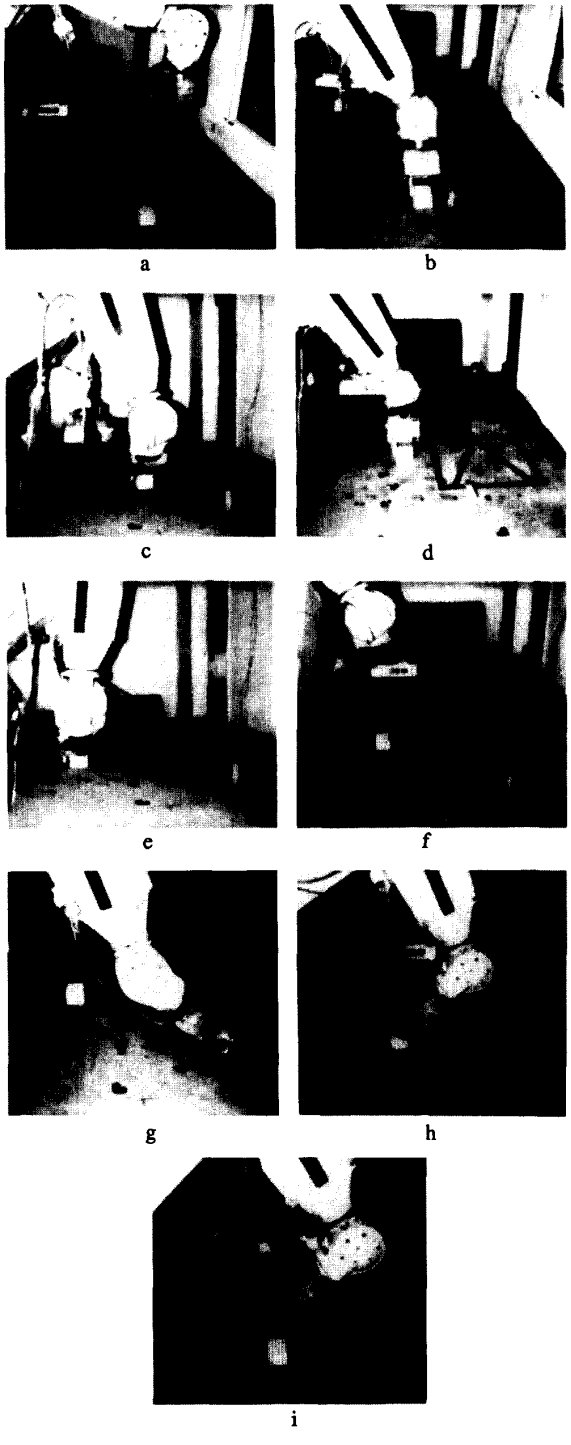


Fig. 6: Robot executing the plan shown in Fig. 4.

At this point, there are a number of areas in our system which are either ad hoc, or require input from the user. For example, the local error recovery plans must be entered by the user, and associated with the uncertainty-reduction goals a priori. One goal of our future work will be to automate this process by employing geometric reasoning about possible errors and error recovery. A further shortcoming of SPAR is the lack of a motion planning module. Incorporating a motion planner with the current system is another goal of our future work.

REFERENCES

[1] A. P. Ambler and R. J. Popplestone, "Inferring the Positions of Bodies from Specified Spatial Relationships," *Artificial Intelligence*, Vol. 6, 1975, pp. 157-174.

[2] W. W. Bledsoe, "The SUP-INF method in Presburger Arithmetic," U. of Texas at Austin Math. Dept. Memo ATP-18, Dec. 1974.

[3] R. A. Brooks, "Symbolic Reasoning Among 3D Models and 2D Images," *Artificial Intelligence*, Vol. 17, 1981, pp. 285-348.

[4] R. A. Brooks, "Symbolic Error Analysis and Robot Planning," *The Int'l Journal of Robotics Research*, Vol. 1, No. 4, Winter 1982.

[5] D. Chapman, "Planning for Conjunctive Goals," *Artificial Intelligence*, Vol. 32, No. 3, July 1987, pp. 333-378.

[6] E. Davis, "Constraint Propagation with Interval Labels," *Artificial Intelligence*, Vol. 32, No. 3, July 1987, pp. 281-331.

[7] R. Dechter and J. Pearl, "Network-Based Heuristics for Constraint-Satisfaction Problems," *Artificial Intelligence*, Vol. 34, No. 1, Dec. 1987, pp. 1-38.

[8] B. R. Donald, "A Search Algorithm for Motion Planning with Six Degrees of Freedom," *Artificial Intelligence*, Vol. 31, No. 3, March 1987, pp. 295-353.

[9] S. N. Gottschlich and A. C. Kak, "A Dynamic Approach to High Precision Parts Mating," *IEEE Trans. on Systems, Man and Cybernetics* Vol. 19, No. 4, July/Aug. 1989, pp. 797-810.

[10] L S. Homem de Mello and A. C. Sanderson, "A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences," *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 1989, pp. 56-61.

[11] Y. F. Huang and C. S. G. Lee, "Precedence Knowledge in Feature Mating Operation Assembly Planning," *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 1989, pp. 216-221.

[12] S. A. Hutchinson and A. C. Kak, "A Task Planner for Simultaneous Fulfillment of Operational, Geometric and Uncertainty-Reduction Goals," Purdue University Technical Report, TR-EE 88-46, Sept. 1988.

[13] Y. K. Hwang and N. Ahuja, "Path Planning Using a Potential Field Representation," University of Illinois at Urbana-Champaign Tech. Report, UICU-ENG-88-2251, 1988.

[14] Y. Liu and R. J. Popplestone, "Planning for Assembly from Solid Models," *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 1989, pp. 222-227.

[15] T. Lozano-Perez and R. A. Brooks, "An Approach to Automatic Robot Programming," MIT AI Lab, AIM 842, 1985.

[16] T. Lozano-Perez, J. L. Jones, E. Mazer, P. A. O'Donnell, W. E. L. Grimson, P. Tournassound and A. Lanusse, "Handey: A Robot System that Recognizes, Plans and Manipulates," *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 1987.

[17] J. Pertin-Troccaz and P. Puget, "Dealing with Uncertainties in Robot Planning Using Program Proving Techniques," *Proc. of the Fourth Int'l Symposium of Robotic Research*, Aug. 1987.

[18] E. D. Sacerdoti, *A Structure for Plans and Behavior*, Elsevier North-Holland, Inc., New York, 1977.

[19] J. T. Schwartz and M. Sharir, "A Survey of Motion Planning and Related Geometric Algorithms," *Artificial Intelligence*, Vol. 37, 1988, pp. 157-169.

[20] R. E. Shostak, "On the SUP-INF Method for Proving Presburger Formulas", *Journal of the ACM*, Vol. 24, No. 4, Oct. 1977, pp. 529-543.

[21] P. Tournassound and T. Lozano-Perez, "Regrasping," *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, 1987, pp. 1924-1928.

[22] D. E. Wilkins, "Representation in a Domain-Independent Planner," *Proc. Eighth Int'l Joint Conf. Artificial Intelligence*, 1983, pp. 733-740.