Task Planning for Robotic Manipulation in Space Applications

A.C. SANDERSON, Senior Member, IEEE
M.A. PESHKIN, Member, IEEE
L.S. HOMEM DE MELLO
Carnegie-Mellon University

Space-based robotic systems will require new technologies of planning and manipulation to accomplish complex tasks such as diagnosis, repair, and assembly. A review is presented of results of work in assembly task representation, discrete task planning, and control synthesis which provide a design environment for assembly systems, and which extend to planning of manipulation operations in unstructured environments. In this approach, assembly planning is carried out using the AND/OR graph representation which encompasses all possible partial orders of operations and may be used to plan assembly sequences. A new algorithm for planning disassembly and repair using the AND/OR graph is introduced, and examples of repair sequences generated for a satellite electrical module are described. For discrete task planning, the configuration map facilitates search over discrete parameters in the space of bounded configuration sets.

Manuscript received March 31, 1987; revised December 8, 1987.

IEEE Log No. 22998

This paper was presented at the Workshop on Space Telerobotics, Jet Propulsion Laboratories, Pasadena, Calif., Jan. 1987.

Authors' current addresses: A.C. Sanderson, Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180; M.A. Peshkin, Department of Mechanical Engineering, The Technological Institute, Northwestern University, 2145 Sheridan Rd., Evanston, IL 60201; L.S. Homem de Mello, Robotics Institute, Department of Electrical and Computer Engineering, Carnegie-Mellon University, Pittsburgh, PA 15213.

0018-9251/88/0900-0619 $1.00 © 1988 IEEE

I. INTRODUCTION

Space-based robotic systems will perform tasks involving dexterity, perception, and planning. Telerobots integrate human perception and human planning capabilities in order to accomplish these tasks, while autonomous systems will incorporate imbedded task planning with accompanying sensory integration, control synthesis, and system architecture to support goal-directed activities in an uncertain environment. Diagnosis, repair, and assembly are tasks which will be essential to the maintenance of space-based systems and will involve both complex manipulation as well as reasoning about configuration and functionality. This paper reviews our recent work on task planning for assembly systems and discusses implications for the development of robotic systems for assembly, maintenance, and repair tasks.

Both manned and unmanned spacecraft present a variety of maintenance and repair problems including materials handling, diagnosis of faults, reasoning about the origin of faults, hypothesis formation and testing, planning and executing repair procedures, disassembly, assembly, and replacement of parts. Currently these tasks may be accomplished in a limited way by on-site manual and teleoperated systems. As the number, complexity, cost, and importance of these spacecraft increases, autonomous systems which can provide service and maintenance on a routine basis will become essential.

Diagnosis and repair are problems in reasoning as well as manipulation. A successful approach to these issues will utilize a representation of the task and an automated reasoning system which enables a decomposition of the problem into feasible sensing and manipulation procedures. Our work on assembly planning is based on several generations of assembly workcells which were built and demonstrated for manufacturing applications [1]. These flexible workcells incorporated multiple robot arms, vision, tactile, and force sensing to accomplish tsks in electronic assembly, wire harness assembly, and assembly of instrument products such as copiers and printers.

Our experience with implementing tasks on these prototype workcells is the basis for current research on the development of tools for efficient design, programming, and implementation of complex systems. Task representation, decomposition, and sequencing [2–4], discrete task planning [8], and adaptive control and learning techniques [9] are principal issues which are being addressed. Embedding such adaptation and learning procedures in the control and planning hierarchy is fundamental to successful implementation in uncertain environments.

Task planning uses geometric and physical constraints of the system to derive feasible operations sequences and precedence relations for accomplishment of specified goals. The AND/OR graph representation of assembly sequences, provides a tool for task planning and control design which incorporates all of the feasible plans in a form which may be conveniently accessed. In many assembly problems the selection of plans may take place off-line. In repair and maintenance problems the plan selection may take place in real-time depending on the conditions of test and observation. Criteria for plan selection are being explored. These include the use of entropy methods [10] to characterize the complexity of subassemblies.

Control synthesis maps the control hierarchy onto the set of feasible assembly plans in order to achieve desired performance. The assignment of system resources is adjusted iteratively subject to task precedence and configuration tolerance constraints. This procedure requires the definition of motion strategies and motion primitives which can be employed. A detailed understanding of sensorless manipulation strategies [5, 6, 7, 8] facilitate planning of sliding, pushing, and grasping operations, and an example of this type of planning is described in this paper. Control structures for vision [9], tactile, and force feedback have been explored, and feasibility of adaptive control strategies for visual servoing has been demonstrated using simulation techniques. This work on sensor-based control is currently being extended to employ learning algorithms at the level of the motion primitive in order to improve performance by local adaptation in the face of uncertainty in the task environment. We have also developed and demonstrated a new approach to arm signature analysis which improves the identification of kinematic models of manipulator structures and increases the resulting positioning accuracy [11].

Implementation of robotic systems in either a telerobotic or autonomous mode will incorporate many of these planning, control, and manipulation capabilities. Development of motion primitives and planning of fine-motion strategies are important topics for both types of systems. The addition of adaptive and learning strategies will permit extension to unstructured environments. The evolution of autonomous systems from telerobotic systems may employ models of human task planning strategies and task representation. The development of components and tools for the space environment will require a consistent task representation which evolves from telerobot to autonomous manipulation.

II. ASSEMBLY TASK REPRESENTATION

In our approach to assembly system design [2, 3, 4], the planning of assembly of one product made up of several parts is viewed as a path search in the state space of all possible configurations of that set of parts. A syntax for the representation of assemblies has been developed based on contact and attachment relations. A decomposable production system implements the backward search for feasible assembly sequences based on a hierarchy of preconditions: 1)release of attachments, 2) stability of subassemblies, 3) separability of subassemblies, including a) local analysis of incremental motion, and b) global analysis of feasible trajectories. For most assembly problems, the branching factor from the initial state to the goal state is greater than the branching factor from the goal state to the initial state. The backward search is therefore more efficient and corresponds in this case to the problem of disassembling the product using reversible operations. The resulting set of feasible assembly sequences is represented as an AND/OR graph and used as the basis for enumeration of solution trees satisfying system and performance requirements.

Fig. 1 shows an example of an AND/OR graph representation of assembly sequences for a simple product with four parts. Each node in the graph corresponds to a subassembly and is described in the representation by a relational structure using the syntax of contacts and attachments. The hyperarcs correspond to the dissassembly operations, and the successor nodes to which each hyperarc points correspond to the resulting subassemblies produced by the disassembly operation. For most products, the assembly operations mate two subassemblies, and the resulting hyperarcs are typically 2-connectors as in this example. The AND/OR graph representation requires fewer nodes than a complete state transition graph for the same product, and therefore the search for solution sequences is computationally less complex.

A solution tree from a node *N* in an AND/OR graph is a subgraph that may be defined recursively as a subset of branching hyperarcs from the original graph. The AND/OR graph representation therefore encompasses all possible partial orderings of assembly operations. Moreover, each partial order corresponds to a solution tree from the node corresponding to the final (assembled) product. The AND/OR graph representation therefore permits one to explore the space of all possible plans for assembly or disassembly of the product. The problem of selecting the best assembly plan may therefore be viewed as a search problem in the AND/OR graph space, and for some given evaluation function on the graph, generic search algorithms such as AO* [12] may be used. In practice, the development of such an evaluation function is very difficult since it would often depend explicitly on implementation issues such as choice of devices and underlying control strategies. We have explored the assignment of weights to hyperarcs using criteria of a) operation complexity, and b) subassembly degrees of freedom, or parts entropy [10]. Such a preliminary search procedure narrows the search space for later examination using implementation details. In the simple examples studied, the resulting ranking of candidate assembly sequence was consistent with intuitive assessment of complexity.

The representation of assembly plans is particularly important for systems which do on-line planning or scheduling. Previous studies of on-line planning problems [13] have used discrete sequence representation or precedence diagrams of operations. In the precedence

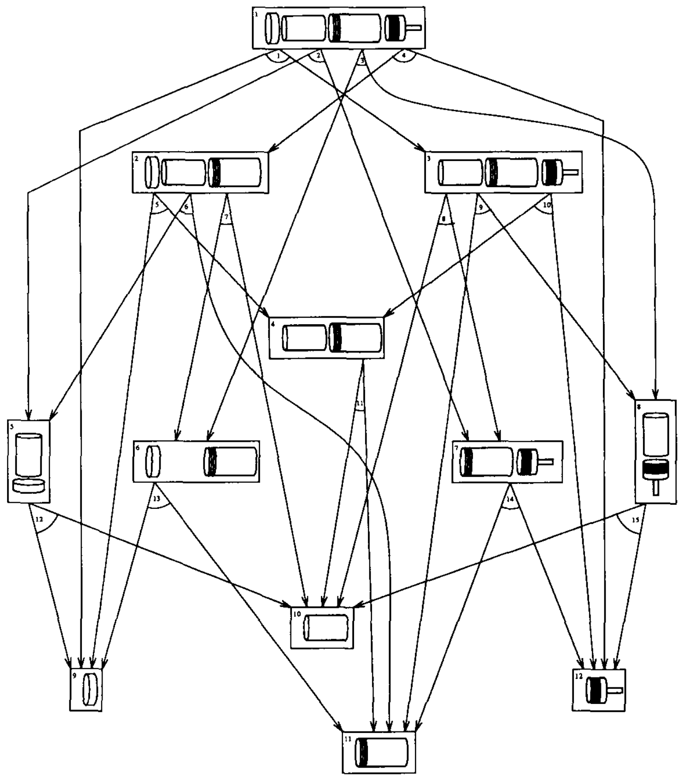


Fig. 1. AND/OR graph representation of assembly plans for simple product.

diagram formalism, typically no single partial order can encompass every possible assembly sequence. The AND/OR graph represents all possible partial orderings of operations, and each partial order corresponds to a solution tree from the node corresponding to the final product. We have illustrated the use of the AND/OR graph for on-line scheduling of a simple robotic workstation with random presentation of parts [2]. The resulting analysis showed a relative improvement in efficiency (number of operations required) from fixed sequence operation of 6 percent for precedence diagrams and 18 percent for the AND/OR graph. The principal advantage of this example was the reduced need for buffering and corresponding retrieval of parts.

The AND/OR graph representation provides a framework for the planning and scheduling of operations sequences. The problems of testing, disassembly, repair, and assembly all benefit from a unified representation which encompasses partial ordering of procedures. Preliminary search of the task space may reduce the candidate subtrees substantially, but the development of final plans typically involves directly the implementation and specification of the underlying devices and motions. In the next session, we illustrate the application of the AND/OR graph representation to repair sequence generation.

III. REPAIR SEQUENCE GENERATION ALGORITHM

Repair tasks require the replacement of identified parts or subassemblies in an assembly. Given a product description, such a replacement task includes both a disassembly and reassembly phase.

Task: Replace <part> in <product>
1) Disassemble <product> until <part> is independent substate
2) Replace <old part> with <new part>
3) Reassemble <product>

Both disassembly and reassembly sequences are generated by searching solution trees in the AND/OR graph. The set of terminal nodes for the disassembly phase of the replacement task are those that correspond to subassemblies that either do not include the part, or include only the part. Alternatively, this set of terminal nodes may also include the nodes which correspond to subassemblies made of parts for which there are replacements available; for example, a module with one defective component can be discarded if replacement for its other components are available. The set of terminal nodes for the reassembly sequence includes the tip nodes of the disassembly solution tree, except the node that corresponds to a subassembly that contains the target part, plus the nodes which correspond to assemblies that have one part only and a replacement is available.

Solution trees are represented by solution tree lists in the following form:

$$
\begin{aligned}
\langle\text{solution-tree}\rangle &= (\langle\text{node}\rangle\ [\langle\text{task}\rangle\ \langle\text{solution-tree-sequence}\rangle])\\
\langle\text{solution-tree-sequence}\rangle &= \langle\text{solution-tree}\rangle \mid \langle\text{solution-tree}\rangle\ \langle\text{solution-tree-sequence}\rangle\\
\langle\text{node}\rangle &= \langle\text{symbol}\rangle\\
\langle\text{task}\rangle &= \langle\text{symbol}\rangle
\end{aligned}
$$

Solution-tree lists, therefore, either have one element only, which is a terminal node, or they have two plus k elements including a nonterminal node, referred to as root, a task, and k trees, atached to the root *through* the task.

This representation of the solution tree is the basis for an algorithm Solve-Assembly which takes as input a node corresponding to a subassembly, checks whether that node is terminal, and, if not, spawns descendent tasks to be solved by an algorithm called Solve-Task. Solve-Task, in turn, takes as input one task and hands all the nodes which result from that task back to Solve-Assembly. The algorithms Solve-Assembly and Solve-Task are shown in Appendix I and are used to generate the feasible sequences for replacement tasks.

The algorithm Replace-Part shown in Fig. 2 is used to generate all complete sequences of operations required to disassemble, replace a part, and reassemble. It first generates all sequences of tasks for withdrawing the part, and for each of these sequences, generates all sequences for reassembly. The algorithm listed schematically in Fig. 2 uses Solve-Assembly to generate both the disassembly and the assembly sequences. These algorithms are implemented in Common Lisp.

```
procedure REPLACE-PART(part product)
begin
sequences ← NIL
initialize data structures used in testing whether a node is terminal for searching disassembly sequences
dseq ← set of sequences that can be used to withdraw part, obtained using SOLVE-ASSEMBLY.
while dseq is not empty do

    begin
    initialize data structures used in testing whether a node is terminal for searching reassembly sequences
        that follow the first sequence in dseq
    aseq ← set of sequences that can be used to reassemble the product after the first disassembly sequence
        in dseq, obtained using SOLVE-ASSEMBLY.
    while aseq is not empty do

        begin
        add to sequences the concatenatation of the first sequence in dseq and the first sequence in aseq
        withdraw the first sequence from aseq
        end

    withdraw first sequence from dseq
    end

return sequences
end
```

Fig. 2. Algorithm Replace-Part.

IV. SATELLITE REPAIR EXAMPLE

The repair sequence planning algorithms have been implemented and used for one subassembly of the Solar Max satellite shown in Fig. 3. The parts are abbreviated as F is frame, P is panel, M is main electrical box, A is connector A, and B is connector B.

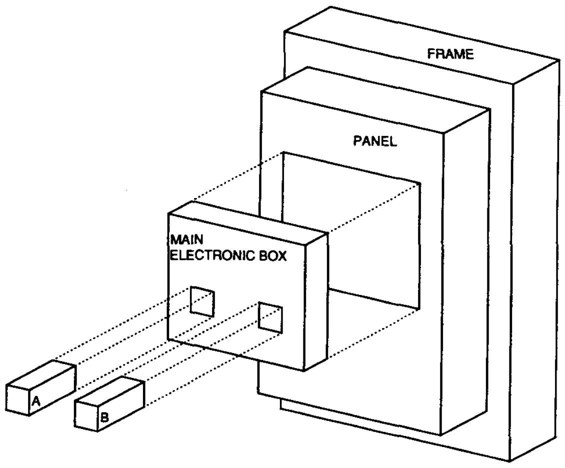


Fig. 3. Subassembly of Solar Max satellite.

The AND/OR graph of the satellite subassembly is shown in Fig. 4. We can illustrate the use of the repair planning algorithm for the task:

replace main electrical box in solar max satellite

Two scenarios are considered for this task. In Scenario A, the main electrical box, M, itself must be replaced alone. In this case the terminal nodes for searching disassembly sequences are the nodes corresponding to the subassemblies A, P, F, M, B, and FP. Fig. 5 shows one solution tree which could be used to disassemble the satellite. The terminal nodes for searching reassembly sequences are the nodes corresponding to A, M, B, and FP. Therefore, the same solution tree can be used to reassemble the satellite.

In Scenario B, there is a spare panel, in addition to a spare main electrical box. The terminal nodes for searching disassembly sequences are the nodes corresponding to subassemblies A, P, F, M, B, FP, and PM. Fig. 6 shows one solution tree which could be used to disassemble the satellite. The terminal nodes for searching reassembly sequences are the nodes corresponding to subassemblies A, P, F, M, and B. Fig. 7 shows one solution tree which could be used to reassemble the satellite after the disassembly sequence of Fig. 6.

V. DISCRETE TASK PLANNING

A sequence of assembly or disassembly subtasks is implemented by performing operations on the parts using system resources such as robot hands, fixtures, or

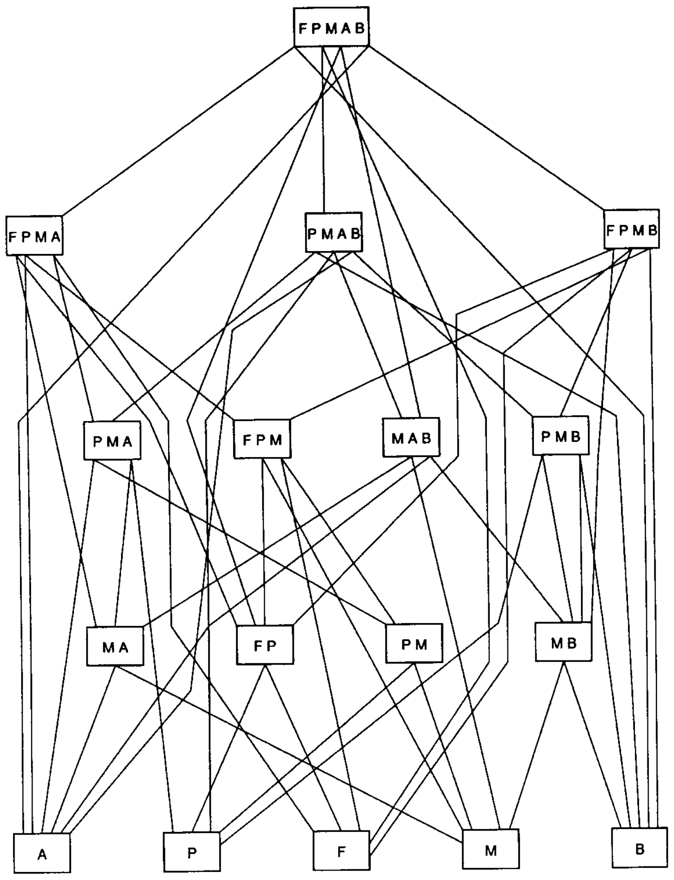


Fig. 4. AND/OR Graph of satellite subassembly.

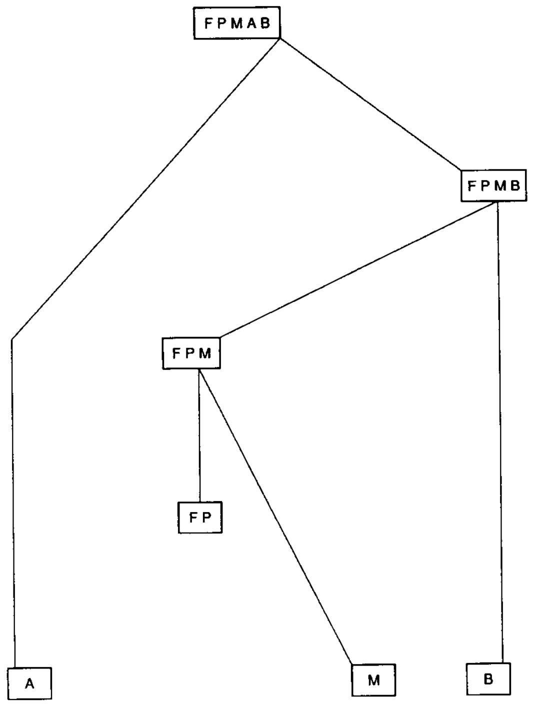


Fig. 5. Solution tree to disassemble and reassemble satellite in scenario A.

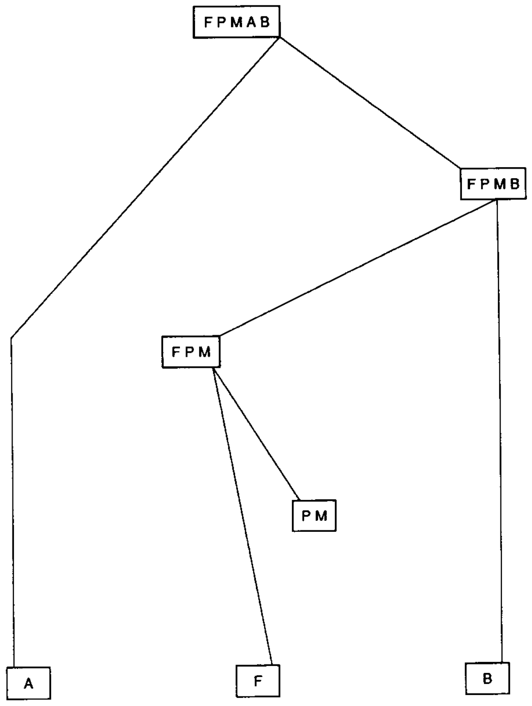


Fig. 6. Solution tree to disassemble satellite in scenario B.

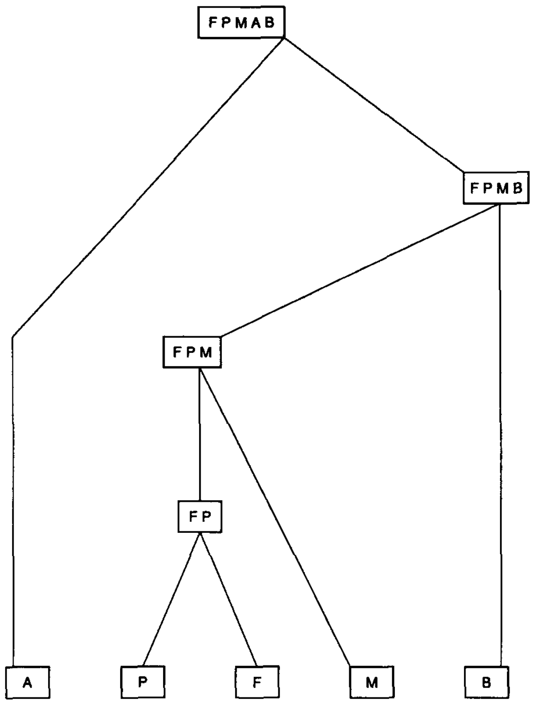


Fig. 7. Solution tree to reassemble satellite in scenario B.

sensors. The allocation of these resources and the synthesis of control programs to coordinate them must be developed in a second level of planning. In general, such operations require detailed motion planning of individual devices. In this section, we describe a definition of discrete operations which lend themselves to planning through manipulation of the configuration map relating input and output configuration states.

Any subtree of the AND/OR graph may be thought of as a subtask precedence graph, and each branch of the subtask precedence graph defines a process in the configuration space C of the parts, where θ^0 indicates the initial configuration state, θ^f indicates a final configuration state, and (θ_i, θ_j) are the states of parts i, j. An assembly operation can then be defined by the following

Assembly Operation:

Given $\theta^0 = (\theta_i^0, \theta_j^0) \in C$
control manipulation, sensing, and computation
to achieve $\theta^f = (\theta_i^f, \theta_j^f) \in T$, then
execute operation

where T = tolerance set,

$T \subseteq C = \theta_i \times \theta_j$ for entities i, j

is the set of configurations (region of configuration space [14]) for which an operation on i, j can be successfully performed.

This definition emphasizes the basic problem in assembly as the control over configuration uncertainty in order to meet tolerance requirements of successive operations. While it is possible to define probability distributions over configurations of parts, in practice, it is very difficult to accurately estimate such distributions, and it is cumbersome to propagate the effect of such distributions through successive operations in a sequence. The configuration map used here provides a tool to compute the effect of operations on bounding sets of configuration points.

A bounding set $B(\nu)$ is defined as

$B(\nu) = \{\text{possible outcomes of } \nu\}$

where ν is a bounded variable. We can define in turn:

joint bounding set: $B(\nu_1, \nu_2, \cdots, \nu_n)$

conditional bounding set: $B(\nu_1 \mid \nu_2 = \eta) = \{\nu_1 \mid (\nu_1, \eta) \in B(\nu_1, \nu_2)\}$

sum of bounding sets: $A + B = \{x \mid x = a + b \text{ for } a \in A, b \in B\}$

scalar multiplication: $cA = \{x \mid x = ca \text{ for } a \in A\}$.

An operation which alters the configuration of a part may be described by a mapping between the initial configuration θ^0, and the final configuration θ^f. An operation with a unique mapping occupies a single point in C-space $\times$ C-space and completely defines the change in configuration state of the system. In this case, planning of operations reduces to planning of unique trajectories in configuration space. As discussed above, such unique mappings are often of limited use due to the uncertainty in configurations and the finite tolerance of operations. Then, states of the objects may be described by bounding sets of points in the configuration space.

The configuration map $M(A_i, B_j)$ describes a single operation which maps a bounded set A_i of input configuration points to a bounded set B_j of output configuration points:

$$M(S_1, S_2) : \{S_1\} \rightarrow \{S_2\}.$$

The configuration map takes on logical values in C-space $\times$ C-space where each logical '1' defines a feasible mapping. The configuration map for a rigid part is a function of twelve dimensions, although in many cases these degrees of freedom are not of equal interest.

The usefulness of the configuration map representation of operations lies in the ease of combining sequential operations. An operation $M_1(\theta^0, \alpha)$ followed by an operation $M_2(\alpha, \theta^f)$ is defined as

$$M_{1\,2}(\theta^0, \theta^f) = M_1M_2 = \cup_\alpha \{M_2(\alpha, \theta^f) \cap M_1(\theta^0, \alpha)\}.$$

Sequences of alternative operations may therefore be compared using simple relations.

The configuration map is particularly useful in cases where inputs and outputs may be partitioned into bounded sets. If we identify N subintervals B of the output space and N subintervals of A of the input space, then a symbolic mapping:

$$M' = \cup_j \{A_j \times B_j \mid M(\theta_i, \alpha) > 0\}$$

defines bounded regions of the configuration map associated with transformations of bounded sets due to a given operation. A useful instance of the bounded set map occurs when we let

$$A_i = \cup_{\alpha \in B_j} \{\theta_i \mid M(\theta_i, \alpha) > 0\}.$$

Then the configuration map

$$M' = \cup_j A_i \times B_j$$

is rectangular and the operation is completely defined by the symbolic map and the definition of the underlying sets.

The product of rectangular configuration maps is completely defined by bounding set operations:

$$M_1 M_2 = \cup_j {}^2B_j \times \{\cup_{k \in {}^21_{C_j}} {}^1A_k\}$$

where

$${}^{2\,1}C_j = \{k \mid {}^2A_j \cap {}^1B_k \neq 0\}$$

is the resulting configuration map product.

Fig. 8 shows the configuration space diagram for an example of a peg insertion operation in two dimensions.

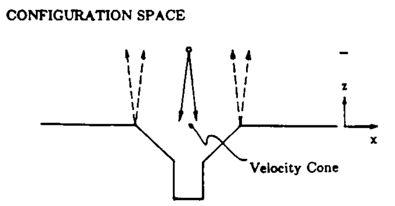


Fig. 8. Configuration space diagram for peg insertion task in two dimensions.

This type of problem has been studied from the point of view of trajectory planning in configuration space [15]. The configuration map shown in Fig. 9 is derived from such a trajectory analysis and summarizes the input-output relations in a manner which permits the resulting discrete operation to be integrated into a task plan. A different configuration map is developed for each set of discrete operations parameters, and the ability to form configuration map products permits search over the space of operations sequences. In Fig. 8, the x position of the peg is regarded as the independent variable of the map, and the initial z position of the peg is fixed for a given configuration map. The operation moves the peg in an a–z direction using a compliant move and directional uncertainty represented by the velocity cone [16].

The resulting configuration map in Fig. 9 has three output bands corresponding to successful insertion, miss-to-the-left, and miss-to-the-right. These three bands occur consistently for different parameter values. Five input bands may then be reconstructed and labelled defining a partitioning of the input configuration space. The resulting map may be rectangularized as shown by the dotted areas, and in that form the symbolic mapping provides a complete description of the operation and a basis for search procedures.

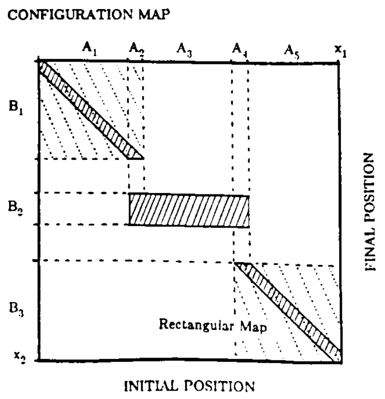


Fig. 9. Configuration map for peg insertion task.

An example of a product of configuration maps is shown for a different set of operations in Fig. 10. Each of these maps is derived from analysis of sliding objects

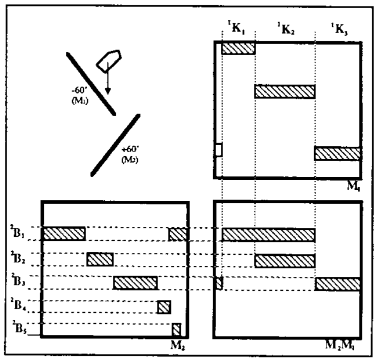


Fig. 10. Product of configuration maps for fence pushing problem.

[5–7] and corresponds to the orientations of a polygonal object being pushed by a two-dimensional fence of finite length. Equivalently, the object may be moving on a conveyor belt past a fixed fence. The independent variable in each map is the object orientation while the operation parameter is the fence angle. The uncertainty represented by the finite width bands in the maps is a result of the unknown support distributions of the objects. In [5–7] we derived bounds on the rates of rotation of such objects and have used these to compute the configuration maps for this example. The product of configuration maps therefore defines the bounds on the sets of orientations resulting from successive fence pushing operations, and can be used as a planning tool for designing sequences of fence push operations to achieve required goals.

For discrete tasks, the space of all operations sequences may be represented by a tree. Arcs correspond to operations, and each node represents a set of possible configuration states after execution of all the operations on the path from the root to that node. Fig. 11 illustrates one such tree which corresponds to a sequence of fence pushing operations for fences of different angles operating on the object shown in Fig. 3. The possible configurations of a part at a given node are obtained by multiplying the configuration maps for the operations on

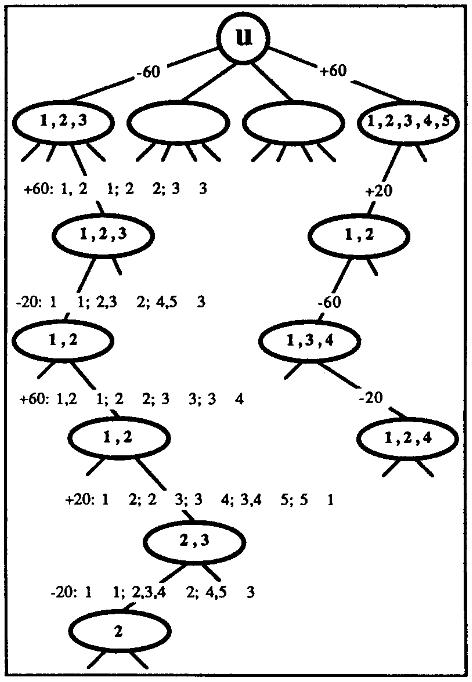


Fig. 11. Search tree for sequences of fence pushing operations.

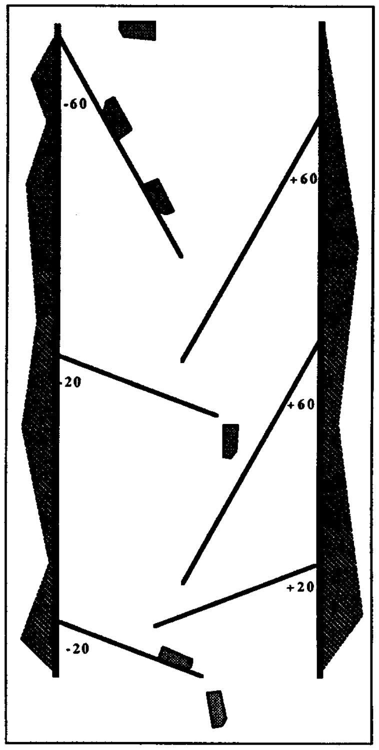


Fig. 12. Resulting parts-feeder design for part geometry shown in Fig. 10.

the path from the root to the node. Traversing the tree in order to search it is facilitated by the ease with which products of multiple configuration maps can be computed using the code sets. Each node is labelled with the subset of the indices *j* of *B* for the bands *B* for the fence angle of the preceding arc. The goal of this task was to reduce the set of possible configurations to a narrow range of orientation, and a search strategy was implemented to reduce the number of output bands to one using the minimum number of operations.

Searching this tree of discrete operations exhaustively is computationally difficult due to the high branching factor which results from the available set of fence angles at each step. Two techniques have been developed to make this search feasible. First, there are systematic relations among bands for different operations parameters. Since there are only a few distinct code sets for the output arcs, it is often possible to systematically choose the subset of arcs which needs to be followed among these outputs. Second, branches of the tree which develop code sets which have occurred previously in a shorter route may be pruned during search.

Implementation of these search techniques permits solution to the fence sequence design problem with the resulting design shown in Fig. 12. This parts feeder design aligns parts with the geometry shown in Fig. 10 independent of the input orientation. Bounds on the orientation of the resulting single band are also derived from the procedure. The output part is then aligned for acquisition or handling by a robot. This search problem requires a few seconds of computation.

VI. CONCLUSIONS

In this paper, we have reviewed several results in assembly representation, discrete task planning, and their relation to underlying control strategies. These methods of planning and manipulation are important for applications which require autonomous systems to carry out complex tasks in diagnosis, repair, and assembly. The development of such analytical tools and their demonstration in prototype systems is an important part of the evolution of telerobotic and autonomous systems for space applications.

APPENDIX I. ALGORITHM SOLVE-ASSEMBLY

```
procedure SOLVE-ASSEMBLY (``s'' ``mode'')
begin
if ``s'' is terminal in type ``mode'' of search return ( (``s'') )
``task-list'' <--- list of tasks that are descendants of ``s''
``node-solutions'' <--- nil
while ``task-list'' is not empty do
    begin
    ``task'' <--- FIRST(``task-list'')
    ``task-list'' <--- TAIL(``task-list'')
    if ``task'' is feasible do
        begin
        ``task-solutions'' <--- SOLVE-TASK(``task'' ``mode'')
        while ``task-solutions'' is not empty do
            begin
            ``node-solutions'' <--- UNION(``node-solutions''
                                          ( (``s'' ``task''
                                               FIRST(``task-solutions'')
                                            )
                                          )
                                        )
            ``task-solutions'' <--- TAIL(``task-solutions'')
            end
        end
    end
return ``node-solutions''
end
```

APPENDIX II. ALGORITHM SOLVE-TASK

```
procedure SOLVE-TASK(``a'' ``mode'')
begin
``node-list'' <--- list of nodes that are descendants of ``a''
``list-of-node-solutions'' <--- nil
while ``node-list'' is not empty do
    begin
    ``node'' <--- FIRST(``node-list'')
    ``node-list'' <--- TAIL(``node-list'')
    ``node-solutions'' <--- SOLVE-ASSEMBLY(``node'' ``mode'')
    if ``node-solutions'' is nil return nil
    ``list-of-node-solutions'' <--- UNION(``list-of-node-solutions''
                                            ``node-solutions'')
    end
return MULTIPLY(``list-of-node-solutions'')
end
```

REFERENCES

[1] Sanderson, A.C., and Perry, G. (1983)
Sensor-based robotic assembly systems: research and applications in electronics manufacturing.
Proceedings of the IEEE (Special Issue on Robotics), *71*, 7 (July, 1983).

[2] Homem-de-Mello, L.S., and Sanderson, A.C. (1986)
AND/OR graph representation of assembly plans.
In *Proceedings of the 1986 AAAI Conference on Artificial Intelligence*, Aug. 1986, pp. 1113–1119.

[3] Sanderson, A.C., and Homen-de-Mello, L.S. (1987)
Task planning and control synthesis for flexible assembly systems.
Machine Intelligence and Knowledge Engineering for Robotic Applications (NATO Advanced Science Institute).
New York: Springer-Verlag, Vol.F33, 1987, pp. 331–353.

[4] Krogh, B.H., and Sanderson, A.C. (1986)
Modeling and control of assembly tasks and systems.
Technical report, *CMU Robotics Institute*, CMU-RI-TR-86-1, 1986.

[5] Peshkin, M.A., and Sanderson A.C. (1985)
The motion of a pushed, sliding object, Part 1: sliding friction.
Technical report, *CMU Robotics Institute*, CMU-RI-TR-85-18, 1985.

[6] Peshkin M.A., and Sanderson A.C. (1986)
The motion of a pushed, sliding object, Part 2: contact friction.
Technical report, *CMU Robotics Institute*, CMU-RI-TR-86-7, 1986.

[7] Peshkin, M.A., and Sanderson, A.C. (1986)
Robotic manipulation of a sliding object.
In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, April 1986, pp. 233–239.

[8] Peshkin, M.A., and Sanderson, A.C. (1986)
Planning sensorless robot manipulation of sliding objects.
In *Proceedings of the 1986 AAAI Conference on Artificial Intelligence*, Aug. 1986, pp. 1107–1112.

[9] Weiss, L.E., Sanderson, A.C., and Neuman, C.P. (1988)
Dynamic sensor-based control of robots with visual feedback.
IEEE Journal of Robotics and Automation (Jan. 1988), pp. 404–417.

[10] Sanderson, A.C. (1984)
Parts entropy methods for robotic assembly system design.
In *Proceedings of the IEEE International Conference on Robotics and Automation*, Mar. 1984, pp. 600–608.

[11] Stone, H.W., Sanderson, A.C., and Neuman, C.P. (1986)
Arm signature identification.
In *Proceedings of the IEEE International Conference on Robotics and Automation*, Apr. 1986, pp. 41–48.

[12] Nilsson, N.J. (1980)
Principles of Artificial Intelligence.
New York: Springer-Verlag, 1980.

[13] Fox, B.R., and Kempf, K.G. (1985)
Opportunistic scheduling for robotics assembly.
In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, 1985, pp. 880–889.

[14] Lozano-Perez, T. (1983)
Spatial planning: a configuration space approach.
IEEE Transactions on Computers, C-32, 2 (Feb. 1983), 108–120.

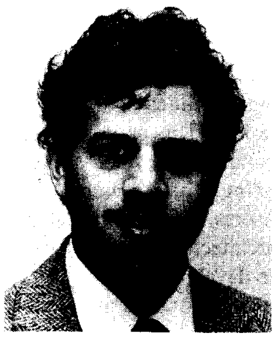
[15] Lozano-Perez, T., Mason M.T., and Taylor, R.H. (1984)
Automatic synthesis of fine-motion strategies for robots.
International Journal of Robotics Research, 3, 1 (Spring, 1984), 3–24.

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

Arthur C. Sanderson (S'66—M'74—SM'86) received his B.S. degree from Brown University, Providence, R.I., in 1968, and his M.S. and Ph.D. degrees from Carnegie-Mellon University, Pittsburgh, Pa., in 1970 and 1972, respectively.

From 1968 to 1970, he was a Research Engineer at Westinghouse Research Laboratories and worked on design and simulation of solid-state electronic devices. From 1972 to 1973 he was a Visiting Research Fellow at Delft University of Technology, The Netherlands, and conducted research in the areas of signal processing and pattern recognition. From 1973 to 1987, Dr. Sanderson was a faculty member at Carnegie-Mellon Univesity in the Department of Electrical and Computer Engineering. From 1980 to 1987 he was Professor of Electrical and Computer Engineering. Dr. Sanderson participated in the founding and development of the Robotics Institute at CMU, and was Associate Director of the Robotics Institute from 1980 to 1987. As Associate Director, he coordinated many of the research initiatives in the Institute. From 1985 to 1987, Dr. Sanderson was on leave from CMU and held the position of Director of Information Sciences Research at Philips Laboratories, Briarcliff Manor, New York. As Research Director, he developed new technical programs in artificial intelligence, computer architecture, computer-aided design, and robotics and flexible automation. In 1987, Dr. Sanderson joined Rensselaer Polytechnic Institute as Professor and Department Head of the Electrical, Computer, and Systems Engineering Department.

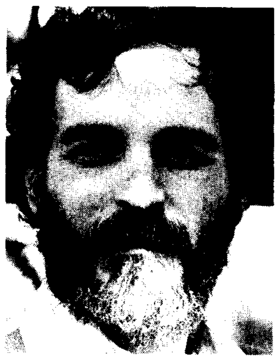
Dr. Sanderson is the author of over 120 technical publications and proceedings. His current research interests include planning systems for robots and automation systems, sensor-based control, computer vision, and applications of knowledge-based systems. He is a Senior Member of the IEEE, and Associate Editor of the IEEE Journal of Robotics and Automation. He was the Program Chairman for the 1987 IEEE International Conference on Robotics and Automation. He is a member of AAAI, SME, and AAAS.



Michael A. Peshkin (S'86—M'86) received the B.A. degree from the University of Chicago, Ill., the M.A. degree from Cornell University, Ithaca, N.Y., and the Ph.D. degree from Carnegie-Mellon University, Pittsburgh, Pa., all in physics.

At Carnegie-Mellon he worked in the Robotics Institute, under the direction of Arthur Sanderson. He is presently an Assistant Professor of Mechanical Engineering at Northwestern University. Current research interests include task planning and force control.

Dr. Peshkin is a 1988 NSF Presidential Young Investigator. He is the author of fourteen technical publications and a forthcoming book, *Planning Robotic Manipulation Strategies*. He belongs to IEEE, ASME, and the American Physical Society.



Luiz S. Homem de Mello has an M.S. degree in electrical engineering and computer science from the University of California, Berkeley, and an Electrical Engineer degree from the University of Sao Paulo, Brazil. He is presently working toward a Ph.D. degree in electrical engineering at Carnegie-Mellon University, Pittsburgh, Pa. His thesis topic is automatic planning for robotic assembly.

Mr. Homem de Mello spent the summer of 1987 at the Jet Propulsion Laboratory, Pasadena, Calif., working in the Machine Intelligence Group of the Electronics and Control Division. From 1980 to 1984 he was a Research Scientist in the Systems Engineering Division of the Instituto de Pesquisas Tecnologicas, in Sao Paulo, Brazil, and from 1982 to 1984 he was an Assistant Professor of Electrical Engineering at the University of Sao Paulo. Prior to that, he had appointments as a Systems Engineer, first at the Sao Paulo Subway Company, and later at Consorcio Nacional de Engenheiros Consultores, both in Sao Paulo, Brazil. His research interests include robotics, intelligent control, and artificial intelligence.