# COLLISION DETECTION AND AVOIDANCE IN COMPUTER CONTROLLED MANIPULATORS

Thesis by

Shriram Mahabal Udupa

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California
1977

(Submitted September 14, 1976)

*To My Parents*

## ABSTRACT

This dissertation tackles the problem of planning safe trajectories for computer controlled manipulators with two links and multiple degrees of freedom.

There are two ways to look at safe trajectory planning. The first concerns itself with planning trajectories in empty space; obstacles enter into consideration only indirectly in that they determine what part of the maneuverable space is free. The second considers obstacles alone; free space considerations are of secondary importance. We show how these complementary views can be used to advantage in the safe trajectory planning problem.

Obstacles are naturally described in cartesian space and trajectories in joint space. If obstacles and trajectories are both represented in one space, collision checks would not require the constant and expensive conversion between the two spaces. We show how it is possible to decompose the planning task so as to get the best of both cartesian space and joint space representations, and yet avoid the constant conversion overhead problem.

We show how the principles of hierarchial decomposition can be used to reduce the complexity of the manipulator trajectory planning problem. Different strategies are used for maneuvering far away from obstacles and for maneuvering close to obstacles. A characterization of large chunks of empty space makes maneuvering far away from obstacles very easy. A characterization of obstacle configuration types simplifies planning of maneuvers close to obstacles.

The key ideas in the representation that make it possible to realize the above claims are:

1) the identification of a hierarchy of abstraction spaces that permit simplified manipulator descriptions. These spaces make it possible to model the manipulator as two line segments, a single line segment, or incredibly as a point.

2) the identification of primitive trajectory types that make collision detection, trajectory hypothesis and modification numerically tractable.

3) the polyhedra-model of obstacles and the identification of one-time-only transformations on obstacles that significantly simplify trajectory planning.

4) a neat characterization of empty space. Empty space is approximated by easily describable entities called charts; the approximation is dynamic and under program control; the approximation can be selective, and thus it is

easy to make incremental modifications to the charts.

The thesis describes a model for collision detection and avoidance systems for computer controlled manipulators. The justification for the model lies in the computer implementations for 2D and 3D manipulator systems. These systems incorporate a significant portion of the model. The promising performance of the implementation makes fast collision avoiders a distinct possibility.

The solution presented treats manipulators with a sliding joint, and permits the manipulator to transport objects which can be enclosed within the minimum bounding cylinder of the manipulator link. Modifications of the solution that permit handling of large objects are indicated. An extension of the solution that solves the problem for manipulators with only rotary joints is described.

A consequence of the investigations into the collision detection and avoidance problem has been the identification of execution-time strategies for terminal phase motion. Guidelines have been presented for incorporating proximity sensors into the manipulator system.

## ACKNOWLEDGEMENTS

I would like to thank Professor Giorgio Ingargiola for getting me started and Professor G. D. McCann for helping me successfully complete my work. My thanks to my thesis supervisor, Professor Meir Weinstein, for his guidance during the intermediate years; his patience and understanding have been of invaluable help, and his attitudes and approaches to problems have considerably influenced my ways of looking at things.

I would like to thank Dr. William Whitney and Mr. Leonard Friedman for their active interest in the research reported here. Their critical evaluation of an earlier draft of this report considerably enhanced its readability and completeness.

I would like to thank Sankaran Srinivas and Scott Roth for the many long discussions that significantly enhanced both the technical content and the presentation of this thesis. I owe a lot to them and to Kent Stevens for making my stay at Cal Tech so much more pleasant and rewarding.

I would like to acknowledge the help of the DEC PDP-10 computer at Cal Tech with whom I have spent many pleasant evenings.

Finally I would like to thank Cal Tech and JPL for providing me with financial assistance throughout my graduate years.

## TABLE OF CONTENTS

## LIST OF ILLUSTRATIONS

# CHAPTER 1
# INTRODUCTION

This chapter introduces the problem of collision detection and avoidance in computer controlled manipulators. It discusses the relevance of the problem in the more general context of autonomous manipulation, and illustrates the problem by an example. The chapter concludes with a historical review of collision detection and avoidance, and an overview of this report.

## 1.1 AUTONOMOUS MANIPULATION

Autonomy in manipulation means that the operations of a manipulator are not controlled directly by a human operator; they are controlled by a programmed system [Bejczy (1972)]. There are two classes of computer controlled manipulator systems:

(1) Numerical Control Machines Manipulators in such systems are programmed for specific tasks. A human programs the complete motion of the manipulator on an analog or a digital computer, or he physically guides the manipulator through the desired motion and the motion is recorded in digital form on tape. The only function of the computer is to recall and execute these pre-programmed motions. A new task or a change in the environment requires complete

reprogramming. The system cannot cope with any unforseen contingencies during operation. Such manipulators are used in industrial robot applications [Olesten (1970)].

(2) Programmable Systems These are general purpose manipulator systems. Such a system is not programmed for a specific task but has knowledge about the manipulator's capabilities and the universe of discourse built into it. Using this knowledge the system can plan a specific course of actions in response to information received in the form of commands from humans or data from its sensors. We will be interested in programmable systems only.

To interact with such a general purpose manipulator system, the user is provided with a formal language for describing computational processes related to the application domain (industrial automation, planetary surface exploration etc.). Using this language the user can specify how objects are to be manipulated and how the manipulator should maneuver around objects. Since the system has extensive models of the universe of discourse, the user is relieved of the burden of specifying all his requirements explicitly. By making use of its internal models, the system analyzes the input requirements and determines how the manipulator will achieve its goals. Once the planning is completed the planned trajectory is executed. During the execution phase, the system can modify the planned trajectory based on real-time evaluations of any sensory data it may acquire.

A general purpose programmable system of the type described in the last paragraph does not exist. Various aspects of the complex problem have been and are being tackled. I will restrict myself to considering the problem of <u>planning</u> the motion of the manipulator from an initial configuration to a new configuration. Considerable progress has been made on the problem of <u>executing</u> such a planned trajectory on a real manipulator. Thus it should be possible to interface my planning system with any system running manipulator hardware. Very little, however, is known about modifying trajectories dynamically based on any sensory data that the system may acquire during execution. I will briefly touch upon this last problem in sections 9.3 and 11.3 but my central problem will be trajectory planning.

## 1.2. TRAJECTORY PLANNING

Trajectory planning deals with how to move the manipulator from a given initial configuration to some new configuration. The manipulator we are interested in is a hand/arm system that is capable of positioning the hand at any point within the maneuverable space and with any orientation. To do so requires three degrees of freedom in two dimensions and six degrees of freedom in three dimensions. Figures 3.1-3.2 show 2D manipulators and Figures 3.3-3.5 and 11.1 show 3D manipulators. For the 3D manipulator we need three degrees of freedom for the hand position and three for the hand orientation thus making a total of six degrees of freedom. There is one joint associated with each degree of freedom. Figures 3.1-3.5 show a two link manipulator with a sliding joint that is called a _mechanical manipulator_. Figure 11.1 shows a three link manipulator with a rotary joint replacing the sliding joint. Such a manipulator is called an _anthropomorphic_ or _humanoid_ or _elbowed_ manipulator.

The description of the manipulator's state can be provided either as a vector specifying the various joint angles or as a position and orientation of the hand. The former representation is called a _configuration_ and is said to be a representation in _joint variable space_ or _joint space_. The latter is a representation in _cartesian space_. The subspace of joint space generated by the three boom joints is called _boom space_. Given a position and orientation in cartesian space, determining the joint angles which will place the manipulator in the desired configuration is called the _position problem_. Solving the position problem for the initial and goal position and orientation will specify how much each joint is to be rotated to effect the transformation.

In addition to reaching the goal configuration, there may be other explicit statements about the intermediate states that the manipulator must go through. These statements are called constraints. An example of a constraint is : keep the forearm horizontal during motion. This is a useful constraint if the manipulator is carrying a cup of coffee and does not want to spill the coffee. In general the manipulator will have to move through space that may contain obstacles. These obstacles need to be avoided. Further the various manipulator joints have limits on the values they can take. In any movement of the manipulator these limits have to be respected. Obstacle avoidance and prevention of joint angle limit violations may be considered as additional constraints.

A trajectory specifies the manipulator configuration as a function of time. A trajectory locus is the curve a trajectory traces in joint variable space. Equivalently, the trajectory is a parameterized representation of the trajectory locus, the parameter being time. The trajectory planning problem is to find a trajectory locus that will take the manipulator from the start to the goal configuration subject to any given constraints. The boom planning problem is to find a trajectory locus for the three joints associated with the boom. Forearm planning refers to planning a trajectory locus for the three joints associated with the forearm. Trajectory planning includes boom planning and forearm planning, and any interactions between the two.

Trajectory calculation deals with computing a trajectory from a trajectory locus. The executive system responsible for servoing the movement of the physical manipulator uses the trajectory representation of the motion. Excellent work on trajectory calculation and servoing has been done by Paul(1972) and Lewis(1974). I will therefore restrict myself to the problem of determining the trajectory locus. I will be using the terms trajectory and trajectory locus interchangeably. Since I will be restricting myself solely to the planning problem, this should not cause any confusion.

The trajectory planning problem is further restricted to that of collision detection and avoidance. Collision detection is concerned with testing for collision with obstacles and joint limit violations. Collision detection is performed by simulating the motion of the manipulator along the proposed trajectory. Collision avoidance is concerned with avoiding potential collisions and joint limit violations. The terms collision avoidance and safe trajectory planning will be used interchangeably.

To summarize, the problem that is of interest is collision detection and avoidance in mechanical manipulators. The solution that is presented permits the manipulator to transport objects which can be enclosed within the minimum bounding cylinder approximating the manipulator link. Extensions and modifications to the solution to permit handling of large objects and anthropomorphic manipulators are indicated.

I have presented the trajectory planning problem for computer controlled manipulators in an abstract manner. However the results of this study have an immediate application to the Jet Propulsion Laboratory (JPL) robot. A brief description of the JPL robot is presented in Appendix 1. Many engineering design decisions had to be made in implementing the solution to the collision detection and avoidance problem. The decisions were made in a manner to suit the JPL robot's manipulator [Dobrotin and Scheinman (1973), Lewis(1974)]. The solution to the safe trajectory planning problem presented here is also applicable to the anthropomorphic manipulator of Figure A3.1. Appendix 3 describes the details of the solution to the anthropomorphic manipulator.

## 1.3 AN EXAMPLE

We are attempting to solve the collision detection and avoidance problem for the three-dimensional manipulator. Since visualizing objects in three dimensions is difficult let us consider the problem in two dimensions. Figure 1.1 shows an example in 2D. In this report I have drawn extensively on examples from the 2D system. Once the 2D system is understood and the similarities between the 2D and 3D systems noted, it is quite easy to visualize the 3D solution.

### 1.3.1 The Problem

The manipulator has two links and three degrees of freedom. The larger link called the boom slides back and forth and can rotate about the origin. The smaller link called the forearm has a rotational degree of freedom about the tip of the boom. The tip of the forearm is called the hand. S and G are the initial and final configurations of the manipulator. Any real manipulator's links will have physical dimensions. The line segment representation of the links is an abstraction. The physical dimensions of the manipulator can be accounted for and how this is done is described later in the report. For now, let us remain with the simple two line segment model.

The closed polygons in the figure represent polygonal approximations to obstacles; these polygons may be concave or convex, and there is no limit to the number of sides.

The problem is to plan a collision free trajectory that will get the manipulator from S to G.

### 1.3.2 The Solution

Since the boom is much larger than the forearm the boom is the more constraining of the two links. To plan the safe trajectory let us, therefore, plan a safe boom trajectory first and then try to maneuver the forearm safely along the boom tip locus.

Boom Planning We can try to get the boom tip from S to G along the shortest path between the two boom tip locations - a straight line boom tip locus. In Figure 1.2 the shaded area represents the area that the boom will sweep when the boom tip traces a straight line from S to G. We notice that the shaded area intersects with the L-shaped object. To avoid collision with the L-shaped object an intermediate point P is chosen and the boom tip is required to go through P. We can then apply the above procedure recursively to the sections SP and PG. Figure 1.3 shows the final boom tip locus that guarantees boom safety.

Forearm Planning Suppose boom planning ends successfully and that it finds a sequence of straight line segments such that if the boom tip travels along these lines the boom will travel safely from S to G. Then if we can maneuver the forearm safely along the boom tip locus we will have found a safe trajectory for the entire manipulator. This is not easy. Furthermore, the maneuverability of the forearm near the goal configuration is very limited. This requires that the forearm be oriented "favorably" when the manipulator nears the goal.

Execution The above planning procedure results in a trajectory locus. Trajectory calculation routines use this trajectory locus to generate a trajectory. The executive system in charge of operating the hardware uses this trajectory to move the manipulator.

Embellishments Planning can be decomposed into two phases called mid-section phase and terminal phase. Mid-section phase has already been described. We could use it to plan safe trajectories relatively far away from obstacles. Terminal phase planning uses obstacle configuration dependent heuristics and we use it to plan motions near the start and goal configurations. Using mid-section and terminal phase planning results in a simpler trajectory. Figure 1.4 shows the boom tip locus for the simpler trajectory.

### 1.3.3 Discussion

The safe trajectory planning problem deals with finding one safe trajectory from an infinite set of safe trajectories. This latter set is a subset of the set of all trajectories the manipulator can execute. Computing a member of or determining membership in this set of safe trajectories is a computationally expensive affair. Thus anyone hoping to find a solution to the safe trajectory planning problem has to determine how to reduce the size of the search space and how to keep the computation tractable.

The following is a list of questions, the answers to which will provide a solution to the collision avoidance problem.

1) How can collision checking be done efficiently? If the physical dimensions of manipulator links are included, collision detection becomes even more expensive. Are simpler descriptions of the manipulator possible?

2) How should the initial trajectory hypothesis be done?

3) On detection of a collision how should trajectory modification be done?

4) How are obstacles, especially the ones with

irregular shapes, to be represented? Intuitively we feel that maneuvering far away from obstacles should be easy. Is it possible to realize this expectation in a computer program?

5) Is the problem of safe trajectory planning better dealt with as planning trajectories in free space rather than as a collision detection and avoidance problem, or is a judicious choice of both approaches possible? How then does one represent free space?

6) What are good "primitives" for trajectories? The primitives should simplify collision checking and make easy trajectory hypothesizing and modifying.

7) Obstacles are naturally described in cartesian space and trajectories in joint space. Since the two are represented in different spaces, collision checks require constant and expensive conversion between the two spaces. Should obstacles, therefore, be described in joint space or should trajectories be represented in cartesian space, or is it possible to use both spaces judiciously?

8) What are good planning heuristics? Should we use the same heuristics for maneuvering close to obstacles and far away from obstacles?

9) If safe trajectory planning is irreparably complex (computationally speaking) can some part of the planning be done at execution-time? Would sensors help in acquiring the

necessary data for execution-time planning? What should these sensors be?

This dissertation provides satisfactory answers to the questions raised here. These answers provide solutions to the collision detection and avoidance problem and make it numerically tractable. The solution is described for manipulators with a sliding joint (see Figures 3.1-3.5).

## 1.4 HISTORICAL PERSPECTIVE

Collision avoidance problems became manifest when computer controlled manipulators came into existence during the mid-sixties. Pieper(1968) was one of the first to investigate the problem. Paul(1972) did some excellent work on trajectory calculation and servoing. Lewis(1974) applied Paul's work to the JPL manipulator and also tackled the safe trajectory planning problem. Widdoes(1974) made by far the most serious attempt at the problem of collision avoidance. None of these earlier attempts could handle the complexities similar to the ones illustrated in the example of Figure 1.1. A comparison of my solution to the safe trajectory planning problem, with those of Pieper, Lewis and Widdoes is described in section 2.4.

## 1.5 OVERVIEW OF THE REPORT

This dissertation presents the solution to the safe trajectory planning problem at a number of different levels. The reader may choose to stop at any level and he should have a good understanding of the solution. If he is interested in knowing more he can go to the next level of detail. A necessary consequence of such an approach to describing anything is repetition. Definitions, descriptions of motivation, representations, algorithms etc. get repeated and are presented in greater detail and often in a slightly different context. The reader who plans to read the entire thesis should be prepared for this.

The problem I am interested in is the safe trajectory planning for the 3D manipulator. To simplify the problem I first solved the problem for a two-dimensional manipulator. Though a simple generalization to three dimensions is not possible, the solution to the two-dimensional problem was very useful in coming up with a solution to the three-dimensional problem.

Chapter 2 provides a theoretical framework for the solution to the collision detection and avoidance problem. For a casual reader interested in knowing the main features of the solution, chapters 2 and 11 should suffice. Chapter 2 restricts its discussions to the three-dimensional problem.

The presentations in chapters 3-9 follow a uniform pattern. The first few sections deal with the motivation and matters of general interest to both the two- and three-dimensional problems. This is followed by the solution to the 2D problem, and a discussion of the natural generalization of the 2D solution and the problems, if any, with such a generalization. The solution to the 3D problem concludes the chapter. Chapters 3-6 explore the models of different entities in the universe of discourse. They include the manipulator, the obstacles in the environment, the free space available for maneuvering, and trajectories. Chapter 7 discusses goal feasibility analysis, Chapter 8 trajectory planning in regions relatively far removed from obstacles and Chapter 9 discusses trajectory planning closer to obstacles.

Chapter 10 presents system details and a critical review of the 2D and 3D implementations. Chapter 11 concludes the report with a presentation of the key ideas of the solution and discusses some directions for future work.

Appendix 1 gives a brief description of the JPL robot. Appendix 2 describes an ordering relation. Appendix 3 indicates how to get further details on the implementations.

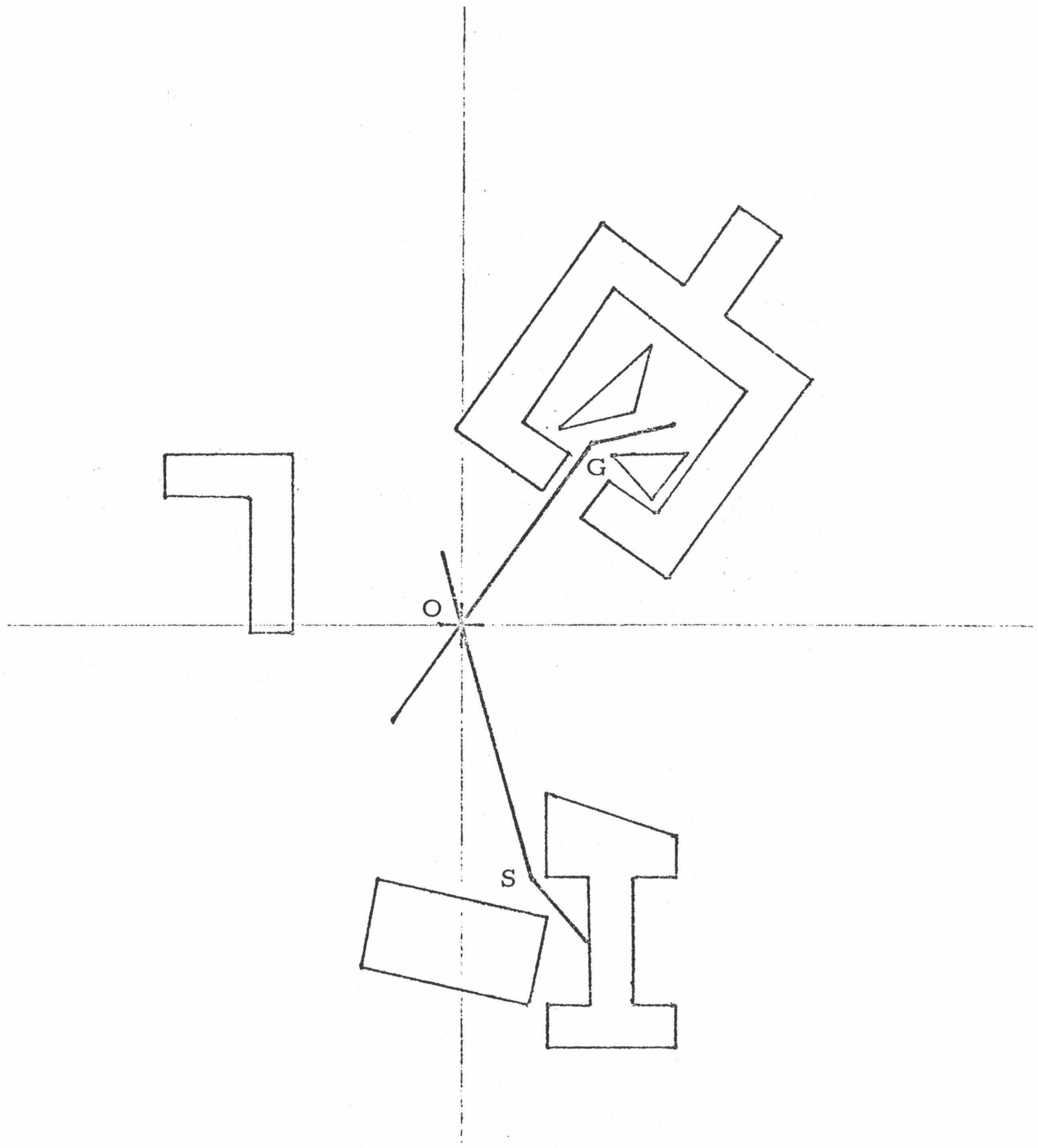


Figure 1.1 An Example

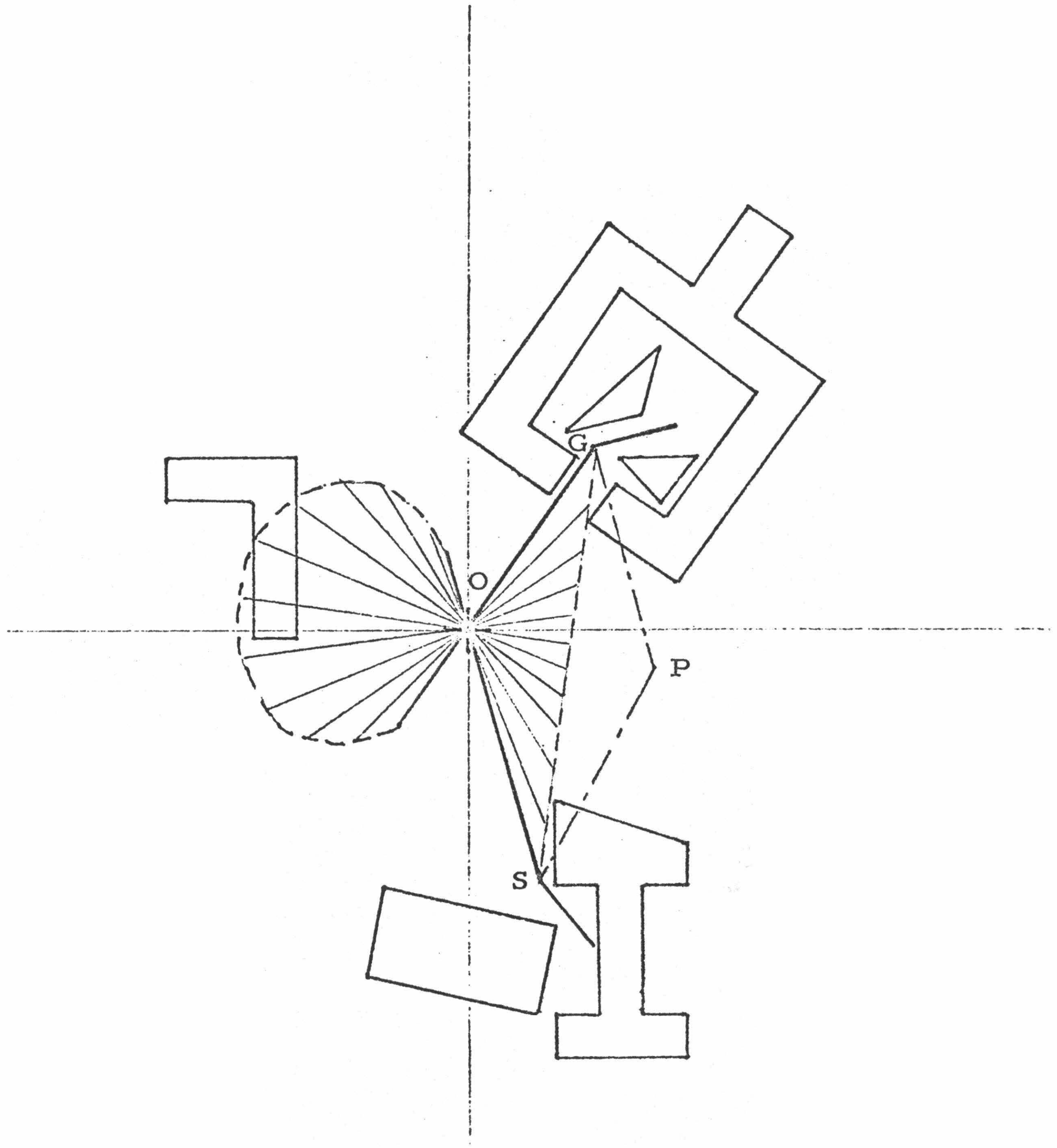


Figure 1. 2 Basic Ideas

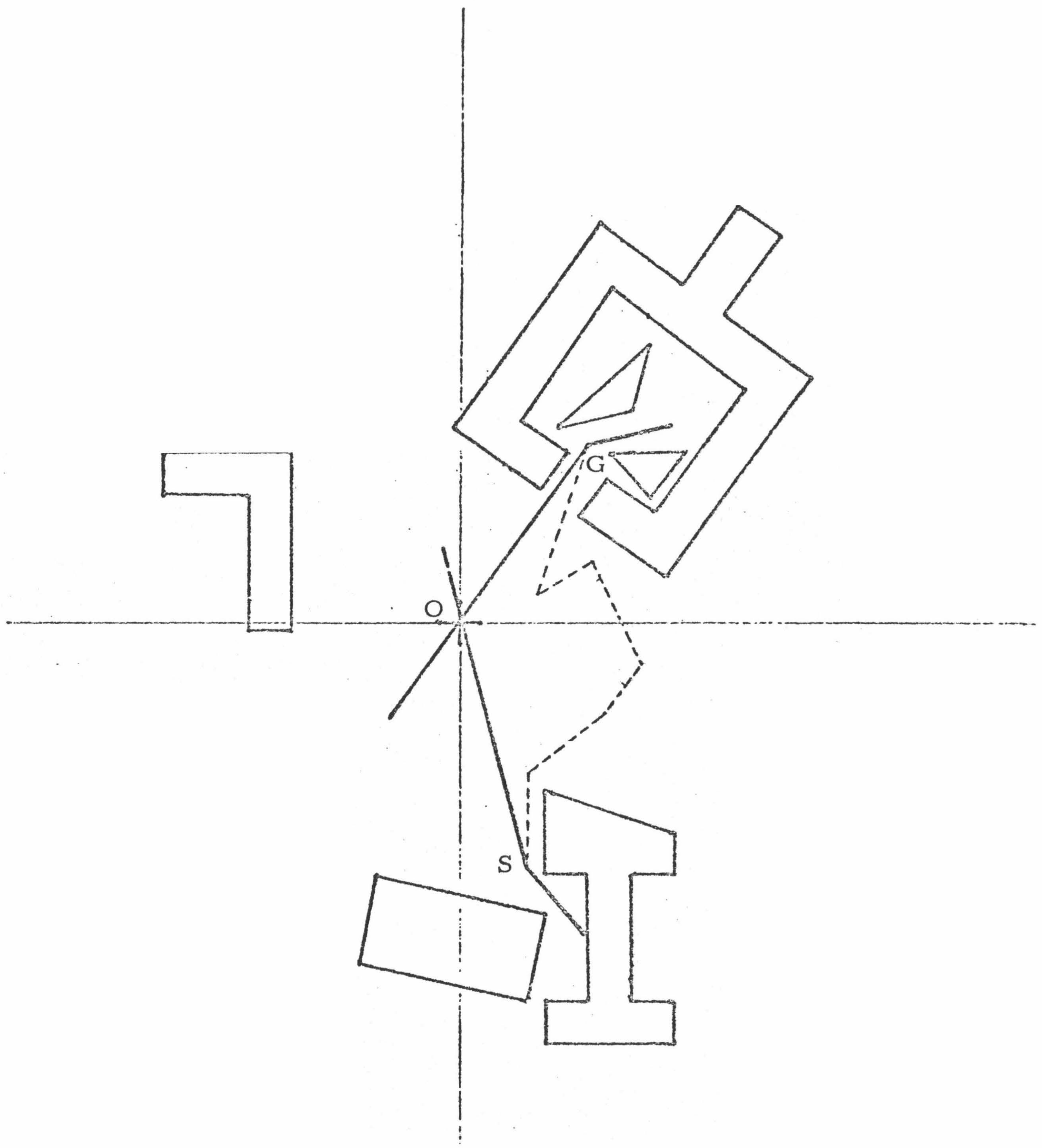


Figure 1.3 Boom Tip Locus

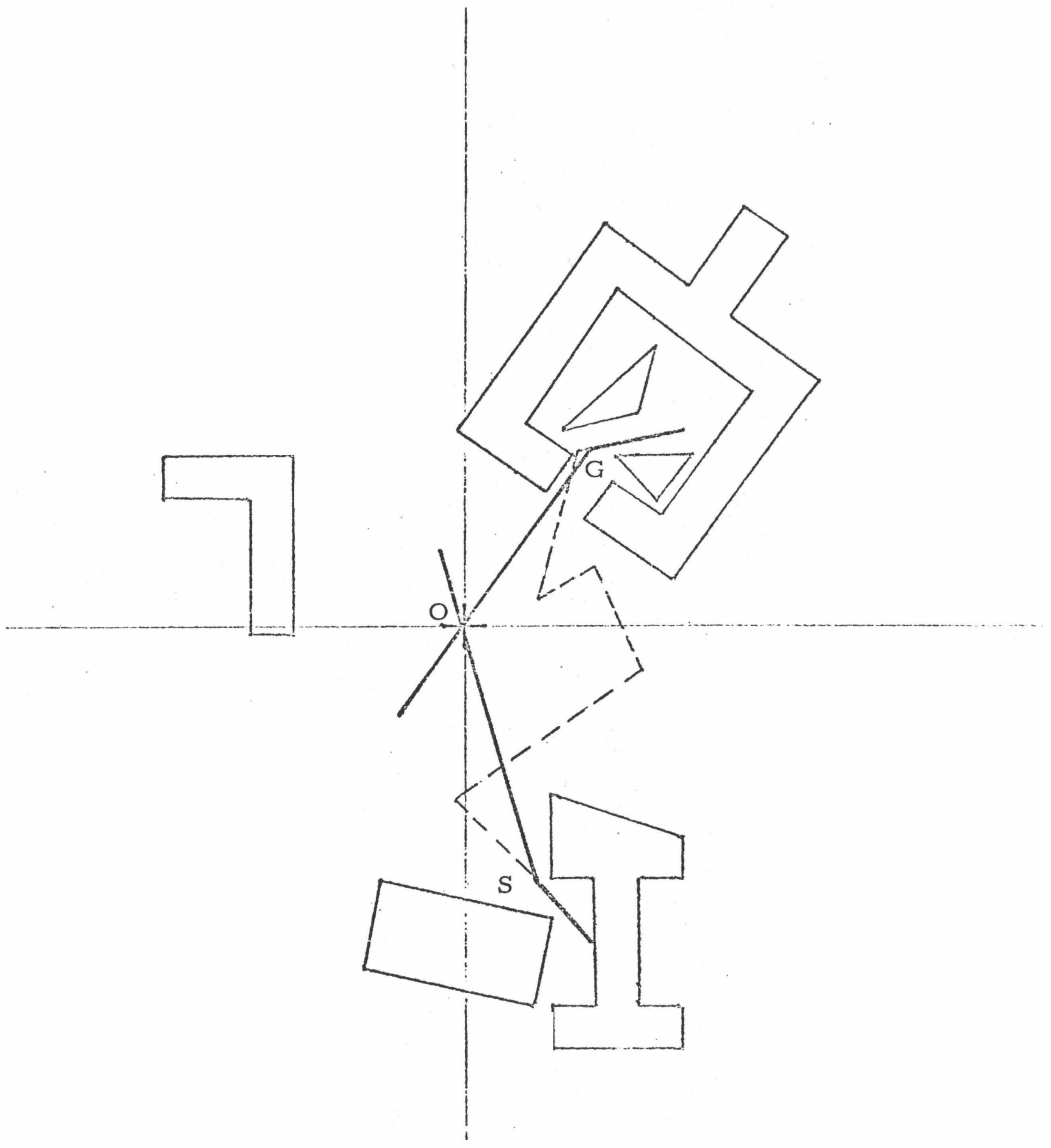


Figure 1.4 Boom Tip-Locus with Terminal Phase Planning

# CHAPTER 2
# A THEORETICAL FRAMEWORK

The trajectory planning task may be pictured as shown in the flowchart of Figure 2.1. The manipulator system on initialization is given a description of the environment. The environment will change due to the manipulator picking up, transporting and putting down objects elsewhere. The environment may also be reinitialized to something completely new. It is assumed that such reinitializations are infrequent compared to the total number of trajectories planned. This assumption is referred to as the _infrequent environment initialization hypothesis_.

The input consists of the position and orientation of the manipulator for the goal configuration. The output is a list of typed intermediate configurations, the type indicating the nature of the subsequent section of the trajectory. The trajectory calculation program uses these type specifications when generating trajectories to run the hardware. The first step in the planning process is to hypothesize a trajectory. Following this is an iterative step which checks for collisions. If there is potential danger, the proposed trajectory is modified and the iteration continued. If the trajectory is safe the planning is over. Let us consider the steps in trajectory planning -

hypothesize and modify trajectories, and check collisions in greater detail.

The routines that hypothesize and modify trajectories will find it very convenient to have a good characterization of 1) Large empty spaces, because a trajectory designed to pass through large empty spaces is very likely to be safe. 2) Terminal obstacle configurations, since special heuristics can be associated with different obstacle configurations, thereby increasing the chances of proposing a collision free trajectory at the first try. What are good representations for empty spaces and obstacles? What are good heuristics for hypothesizing trajectories? How and where should trajectory modification be effected so that the same problem does not recur, and that new problems do not arise? This dissertation provides answers to these questions.

Collision detection has its own problems, making it computationally expensive. Since it is a computation which is repeated many times it is essential to make this step efficient. Trajectories are most conveniently described in joint variable space while obstacles are described naturally in cartesian space. When the manipulator moves, its links trace a volume in cartesian space called the <u>trajectory envelope</u>. Collision detection involves checking intersections of the trajectory envelope (represented in joint variable space) and obstacles (represented in cartesian space). Since the two are represented in different spaces, intersection checks require constant conversion between the two spaces. This makes the checks expensive and is referred to as the <u>conversion overhead problem</u>. Should obstacles, therefore, be described in joint variable space or should trajectories be represented in cartesian space or is it possible to use both spaces judiciously? This thesis shows how it is possible to use the best of both the joint variable and cartesian space representations. Again, safe trajectory planning can be viewed as a) maneuvering in free space, and b) avoiding obstacles. This thesis shows these complementary views can be used to advantage in solving the planning problem.

Now, the complexity of planning is a function of how the manipulator is modelled. This thesis shows that the manipulator can be modelled in a number of problem spaces of increasing abstraction. Starting with a simple and direct model of two connected cylinders, we go to show how the manipulator can be modelled as two connected line segments, a single line segment, and incredibly as a point! If we model the manipulator as two connected cylinders we will be operating at the most complex level; with a point model of the manipulator the planning problem will be the simplest.

This thesis describes the different problem spaces, their properties, how they are generated and the relationship between them.

The solution will be presented in two parts - representation and planning. Section 2.1 outlines the criteria for a satisfactory solution. The last section of this chapter is a survey of the past work as seen in the framework of my solution.

## 2.1 SOLUTION CRITERIA

We are looking for a system that solves the trajectory planning problem in a variety of obstacle configurations. We want a system that plans safe trajectories in a time comparable to the execution time of the trajectories, which for the JPL robot's arm is between 5 and 10 seconds. The system need not produce a truly optimal plan. In fact an optimal plan is often not worth the extra computation required to produce it. At the same time the system should not produce blatantly stupid plans. We do not want the manipulator to do any unnecessary acrobatics. The system should perform well in simple and commonly occurring situations and it may take more time on difficult problems. It should be able to recognize when things go out of hand and ask for assistance from a human when that happens.

## 2.2 REPRESENTATION

The representation aims at simplifying the tasks of hypothesizing and modifying trajectories, and checking the safety of proposed trajectories. The entities in the universe of discourse that need to be represented are the manipulator, the obstacles in the environment, the maneuverable free space and trajectory envelopes.

The infrequent initialization hypothesis, in effect, says that a large number of trajectories are planned for any given environment. In view of this it is worthwhile looking for alternative problem spaces where the computational complexity of trajectory planning might be less.

Multiple problem spaces are extensively used in engineering and science. The time-domain and frequency-domain analysis of dynamic systems is a classic example. Whenever multiple representations are used equivalence of the representations is of great importance. Intuitively, equivalence of representations guarantees the existence of solutions in the alternative space when a solution exists in the first and vice versa. The second important aspect about multiple representations is concerned with transformations between spaces.

The representation hierarchy of Table 2.1 shows three problem spaces. The table also describes the representation of every entity in the world in each of the three problem spaces. The relationship between the three problem spaces is described first and the individual spaces are described next. The description of a problem space includes the representations for the four components of the world, their inter-relationships and how they are generated. Their use in trajectory planning is described in section 2.3.

The first space is called the _real problem space_ and is closest to the real world. A solution to the trajectory planning problem in this space is a solution to the trajectory planning problem in the real world. The converse is true if one ignores the fact that obstacle shapes are approximated by bounding polyhedra. When a solution to the trajectory planning problem in one space implies a solution in the other and vice versa, the two spaces are said to be _equivalent_(*).

--------------

* The concept of equivalence used here is in the sense described in Chapter 4 of Shoenfield(1967).

Solving the trajectory planning problem in the real problem space is easier than doing so for the real world. The task, however, is still quite difficult. We therefore identify a new space called the _primary problem space_, that is equivalent to the real problem space and wherein the complexity of the task is greatly reduced. The reasons for the simplicity of trajectory in the primary problem space will be presented later. The process of generation of the primary problem space ensures that the trajectory in the primary problem space is identical to a trajectory in the real world i.e. there is an identity transformation relating the solution in the two spaces.

The primary problem space simplifies the trajectory planning problem considerably as compared to the real problem space. However, forearm planning is still quite expensive and so a third space called the _secondary problem space_, is introduced to simplify forearm planning. The secondary problem space admits a simple description of the manipulator; the manipulator consists of just the boom. As a consequence of this simplicity the primary and secondary problem spaces are _not_ equivalent. For, it may happen that there is a solution to the trajectory planning problem in the primary problem space but not in the secondary space. However, a solution in the secondary problem space always implies a solution in the primary problem space. This is discussed in detail in section 2.2.3. Trajectory planning in the secondary problem space is so simple that it more than justifies planning in a problem space that is not equivalent to the real world. Again, as in the primary problem space, the process of generation of the secondary problem space ensures that there is an identity transformation relating trajectories in secondary space to trajectories in the real world. The primary problem space is an <u>extension</u>(*) of the secondary problem space.

The relationship between the three spaces is summarized below:

$$\begin{tikzcd} \text{Real Problem Space} \arrow[d, Leftrightarrow] \\ \text{Primary Problem Space} \\ \text{Secondary Problem Space} \arrow[u, Rightarrow] \end{tikzcd}$$

where A => B means a solution in space A implies a solution in space B and A <=> B means A => B and B => A.

--------------

* The concept of extension used here is in the sense described in Chapter 4 of Shoenfield(1967).

### 2.2.1 Real Problem Space

Any object the manipulator is likely to collide with is termed an obstacle. On the JPL robot (see Figure A1.1) obstacles would include the platform, the interface electronics rack, the TV and laser rack, the wheels and their motors etc. If the robot is operating in an outdoor environment a boulder within the manipulator's reach would be considered an obstacle. Some of the obstacles have well defined geometric shapes such as parallelopipeds, cylinders, toroids etc. Others, especially natural obstacles are very irregular in shape. Since collision detection involves determining intersection of shapes, the more complex the shape, the more the computational effort. Also, the less succinct the description, the more will the storage requirements be. To reduce the storage requirements and the computational time for intersection checking, the obstacles are replaced by their enclosing polyhedra. These polyhedra may be concave or convex. There is also no limit on the number of faces and thus the approximation by polyhedra can be accurate to any arbitrary degree. The set of polyhedra, each approximating a real obstacle, is called the map.

The maneuverable space is the complement of the volume occupied by elements of the map, with respect to the manipulator's workspace.

The collision detection and avoidance system handles computer controlled manipulators which can be abstractly described as having two links and multi-degree of freedom. An example of this class of structures is the Scheinman arm shown in Figure 3.4 [Dobrotin and Scheinman (1973), Scheinman (1969)]. Abstractly, the arm consists of a <u>manipulator post</u> and two links, one called the <u>boom</u> and the other called the <u>forearm</u>. When looking along the boom at the forearm, the boom is either on the right or the left side of the manipulator post. This gives rise to the notion of a right-handed and left-handed manipulator, and is called the <u>lateral property</u>. Since the manipulator post is fixed, it can be considered to be an obstacle. The boom and the forearm have physical dimensions, length, breadth and width. When these links move they trace a volume in space and the trajectory envelope, therefore, is a two-element three dimensional <u>solid</u>.

Figure A3.1 shows a humanoid "two" link and six-degree freedom manipulator. [Winston (1974), page 221]. It has a rotary joint in the center of the arm (an "elbow"). It differs from the manipulator of Figures 3.4-3.5 in that all its joints are rotary; the mechanical manipulator of Figures 3.4-3.5 have one sliding joint. The mechanical and humanoid manipulators are similar in all other respects.

### 2.2.2 Primary Problem Space

The primary problem space admits simplified manipulator descriptions which simplify trajectory planning while still maintaining equivalence with the real problem space. Instead of considering the manipulator as consisting of two solid links, the manipulator is viewed as consisting of a single line segment and having no lateral property. In order to preserve the equivalence with the real problem space, appropriate transformations are made on the obstacle and maneuverable space descriptions. It is very essential that these transformations have the following minimality property: the transformations need to be computed only once or if this is not possible then the number of times the transformation is computed should be far less than the number of trajectory computations. Otherwise the advantage gained by using the simplified representation would be lost in the generation of the representation.

Consider the minimum bounding cylinders for the boom and the forearm. The finite axis of the cylinder bounding the forearm is the single-line segment model of the manipulator. I will now describe how such a simple view of the manipulator is possible while still preserving equivalence with the real problem space.

First consider a two-line segment model of the manipulator. The finite axes of the cylinders bounding the boom and forearm are used for this model. In order to preserve equivalence we enlarge the obstacles. Let k be the radius of the cylinders. Each polyhedron in the map is subject to the _enlarge_ transformation. The transformation generates a new polyhedron such that every point on the surface of the new polyhedron is at least a distance k away from the nearest point on the surface of the old polyhedron. The enlarged polyhedron is called a _primary obstacle_. The set of primary obstacles is called the _primary map_. With line-segment models of the manipulator links, the trajectory envelope is now two connected surfaces, one called the _boom surface_ and the other the _forearm surface_. The maneuverable space is called _primary free space_ and is the complement of the volume occupied by primary obstacles with respect to the manipulator's workspace. The original collision detection and avoidance problem is equivalent to the simplified collision detection and avoidance problem for the line links and the enlarged obstacles of the primary problem space. The enlargement transformation needs to be done just once.

Next, in order to ignore the lateral property of the manipulator and still maintain equivalence between the real and primary problem spaces, appropriate one-time-only transformations are used to generate a _left primary map_ and a right primary map. The polyhedra descriptions in these maps reflect the manipulator characteristics. This finer classification of maps was left out of Table 2.1 so as to keep the table simple.

Finally, the single element description of the manipulator is made possible by a transformation called survey which permits the boom to be viewed as a single point instead of a finite line segment. The trajectory envelope will then be the forearm surface generated by the motion of the forearm line segment. Survey when applied to free space results in a chart. The nomenclature stems from the use of charts for navigation. A chart generated to represent primary free space is called a primary chart. To see what survey does we start with primary free space. Consider the set of all points in the primary free space such that the entire boom is safe from collision if the boom tip were positioned there. This subset of free space is called navspace (for navigational space). The survey transformation approximates navspace by boxes in r-theta-phi space called regions and the set of regions is called a chart. Corresponding to the left and right primary maps we have the left primary chart and the right primary chart. Again, to keep matters simple, the finer classification of charts was left out of Table 2.1.

Regions are structured entities (see Figure 2.2). They are made up of <u>sectoroids</u> and sectoroids are composed of <u>pascs</u>. The pasc (<u>pa</u>rallelepiped in <u>s</u>pherical <u>c</u>oordinates) is the smallest unit. The choice of the parallelopiped in spherical coordinates as the unit of shape is based on how the planning routines will use them. Pascs, sectoroids and regions are bounded by constant phi and constant theta surfaces. All pascs in a sectoroid have the same phi limits. All sectoroids in a region have the same theta limits. Pascs have associated with them a maximum and minimum r value, called <u>rmax</u> and <u>rmin</u> respectively, indicating the safe limits of the boom extension. The difference between the maximum and minimum r value is called the <u>safe limit interval</u>. Similar to pascs, sectoroids and regions have associated with them maximum and minimum r values indicating the best possible safe limits of the boom extension. A region, sectoroid or pasc is considered <u>impassable</u> if the safe limit interval is less than some prespecified value.

Regions essentially are an approximation to the points in navspace. This approximation is dynamic and can be changed by higher level programs. The approximating procedure is called <u>refinement</u>, and the refinement level is called <u>resolution</u>. The system can refine areas where the manipulator needs to maneuver in to a greater resolution, while elsewhere the resolution may be quite crude. This flexibility is very useful because refining every part of free space to the finest level possible is expensive and often quite unnecessary. This flexibility permits the system to decide where refinement is essential and what the resolution should be. If the resolution of a particular part of the environment is not adequate, the system can refine that portion of the maneuverable space. This is termed the <u>selective</u> <u>refinement</u> capability. As a result of this capability, the survey transformation is not a one-time operation. This is the price that has to be paid for the flexibility. Since there is a limit to the precision of placement of the hardware the process of refinement will not continue indefinitely. The data structures generated during the refinement process are saved for reuse. Selective refinement makes <u>incremental</u> <u>modifications</u> to the chart very inexpensive. Incremental modifications are necessitated by minor changes in the environment that might result from the transporting of objects from one place to another.

The concept of navspace permits considering the boom as a single point. Navspace and its approximation by charts is thus crucial to safe trajectory planning. The reason for imposing a structure on charts is to have some selectivity in terms of what parts of navspace should be refined and to what level. It is important to note that the exact nature of a region and its components is irrelevant to the concept of navspace and collision checking. The choice of boxes in r-theta-phi space as the unit is dictated by the choice of a particular planning strategy described in section 2.3.3. The concepts of navspace and charts, however, are independent of planning strategies.

## 2.2.3 Secondary Problem Space

In the primary problem space the manipulator was viewed as a single line segment with no lateral property. The secondary problem space admits a still simpler description of the manipulator - a single point. Unfortunately, as mentioned earlier, the secondary problem space representation is not equivalent to the primary problem space. However the primary problem space can be made an extension of the secondary space and to do so appropriate transformations are made on primary obstacles. As before we require that these transformations satisfy the minimality property (see section 2.2.2).

First consider the two line segment model of the manipulator. The finite axes of the cylinders bounding the boom and forearm are used for this model. Suppose we ignore the forearm. The trajectory envelope will be the boom surface generated by the motion of the boom line segment.

The polyhedra in the primary map are enlarged by the length of the forearm. This enlargement results in <u>secondary obstacles</u> and a <u>secondary map</u>. The maneuverable space is called <u>secondary free space</u> and is the complement of the volume occupied by the secondary obstacles with respect to the manipulator's work space.

The single point description of the manipulator is made possible by applying the survey transformation to secondary free space resulting in a <u>secondary chart</u>. Secondary charts are composed of <u>secondary regions</u>. Whenever the boom tip is in a secondary region the following are true : 1) by definition of the region the entire boom is free of collisions, and 2) since secondary regions are generated using secondary obstacles, the forearm is free from collision irrespective of its orientation. The trajectory envelope at this level then is the line generated by the motion of the boom tip. A complex two-element trajectory solid has thus been reduced to a line. The <u>refinement</u> process for secondary charts is similar to primary charts and so are all the attributes and transformations discussed in the context of primary charts. In secondary problem space too, there are <u>left</u> and <u>right secondary maps</u> and <u>left</u> and <u>right secondary charts</u>. Again, to keep matters simple, the finer classification of charts was left out of Table 2.1.

If the manipulator needs to maneuver close to obstacles, secondary problem space is of no use. The "gross" representation of the forearm results in the system complaining that trajectories close to obstacles are not feasible. Of course this does not mean that a trajectory necessarily does not exist. The finer model of the forearm as a line segment (as in primary problem space) should result in better performance. This is what I meant when I said that if a solution to the trajectory exists in secondary problem space then there is a solution in primary problem space, while if there is no solution in secondary problem space it does not mean there is no solution in primary problem space. Equivalently, the above remark is same as saying that every safe trajectory in primary problem space need not be a safe trajectory in secondary problem space.

Looked at slightly differently, the ideas of secondary problem space representations (the secondary charts in particular), are a formal characterization of the intuitive ideas of ease of maneuvering in large chunks of empty space far away from obstacles. The reduction of the trajectory solid to a line makes the expectation come true. Since, close to obstacles, secondary problem space representations are not fine enough, primary problem space representations will have to be used. With primary problem space representations the trajectory envelope is a surface and this is in accordance with our intuitive feeling that maneuvering close to obstacles is not as easy as maneuvering far away from them. A judicious use of secondary and primary problem space representations will significantly reduce the search space for good candidate trajectories, and considerably simplify the collision detection and avoidance task.

Left and right primary maps, and left and right secondary maps were described as four different entities, and so were the charts. For efficiency considerations, in the implementation, left and right primary obstacles are grouped together, and so are left and right secondary obstacles. With charts, the primary and secondary regions are grouped together while the left and right regions remain distinct.

## 2.2.4 Trajectory Envelopes

The discussion of the three problem spaces showed how simpler and simpler manipulator descriptions reduced the complexity of the trajectory envelope from the two-element solid to a single surface, the trajectory surface, or even a single line, the trajectory trace. Collision detection involves the determination of the intersection of the trajectory envelope and the obstacle faces. The complexity of this task depends on the nature of the trajectory envelope. It is therefore imperative that we look for additional constraints to further reduce the complexity of collision checking. Since obstacle faces are planes in cartesian space, if the trajectory surface (trace) were a plane (line) in cartesian space, collision checking would be simple.

Since the manipulator hardware permits each of the joints to be operated independently it should be possible to get the boom tip to trace cartesian space straight lines. However, planning cartesian space straight line loci for the boom tip is beset with computational problems. We choose therefore to settle for a boom space straight line locus for the boom tip. Boom space is the subspace of joint variable space generated by the three boom joint variables. This straight line in boom space can then be approximated by a sequence of straight lines in cartesian space. Safety of the boom tip locus guarantees the safety of the entire manipulator only when the locus passes through a secondary chart. Elsewhere the trajectory envelope is still a surface and, to make collision checking tractable, constraints on forearm motion have to be introduced. We choose the following trajectory primitives for the forearm. When the boom is moving, the forearm tip shall trace a straight line in cartesian space parallel to the approximated boom tip locus, and when the boom tip is stationary the forearm shall move in a single plane. These constraints on the boom and forearm result in the decomposition of the trajectory surface into a sequence of parallelograms and sectors of a circle, enormously simplifying the collision detection task.

### 2.2.5 <u>Concluding remarks</u>

The primary and secondary problem space representations along with the restrictions on the nature of the trajectory make the trajectory planning problem numerically tractable. Precisely how these representations are used in planning is described in the next section.

## 2.3 PLANNING

The first step in the planning process is to hypothesize a trajectory. Following this is an iterative step which checks for collisions. If there is potential danger, the proposed trajectory is modified and the iteration continued. If the trajectory is safe the planning is over (see Figure 2.1). The central aim is to reduce the planning time. It is therefore essential that very few errors be made during trajectory hypothesizing and suggesting of modifications since errors will need costly fix-ups that the system can ill afford.

Hierarchy, separability and reversibility are the key concepts in planning. The principle of reversibility states that if a trajectory from S to G is collision free then the same trajectory backwards from G to S is also collision free. Hence for collision detection and avoidance it does not matter whether a trajectory from S to G or G to S is planned. Separability means the decomposition of the goal into disjoint, reasonably independent parts. Hierarchy is used in the usual sense. For each part of the goal the most important aspects are tackled first and the lesser ones next. This is applied to every stage of the process. If some decisions made at a higher level do not pan out, local corrections are made. If the local fix-ups do not solve the problem the system returns to the next higher level for replanning. Some indication as to what went wrong is preserved and is used during subsequent attempts at planning. At each stage it is ensured that the system will terminate its activities in a finite amount of time. If the system is not successful in solving the problem it gives up and asks for human help.

The goal is specified as a 3-space position and orientation of the forearm. The position problem for the goal is solved. In other words, the joint angles which will place the manipulator in the goal configuration are determined and the ones corresponding to the starting manipulator configuration (same lateral configuration) are chosen.

## 2.3.1 Two Approaches

Conceptually, since the trajectory envelope has the simplest description in the secondary problem space, planning should start in that space. Having planned as much of the trajectory as possible in the secondary problem space, the system should attempt to plan the rest of the trajectory in the primary problem space. In both spaces the system should use the principles of hierarchy and separability. The main drawback of this approach is that the problem of interfacing between the two spaces has no easy solutions.

Another approach to planning is the following : Plan the boom trajectory in primary problem space. For parts of the boom tip locus that lie within the secondary chart (of secondary problem space) no forearm planning needs to be done. For the remaining sections, forearm planning is carried out. Instead of starting with secondary problem space and then going over to primary problem space, the second scheme starts with primary problem space. The partial solution in primary problem space is "refined" using the secondary problem space. This alternative way of looking at the trajectory planning problem solves the interface problems that plagued the first scheme. The details of the second approach will now be presented.

## 2.3.2 Overview of Planning

The trajectory planning problem is separated into three phases. The first is a goal feasibility analysis phase, the second is the mid-section planning phase and the last is the terminal planning phase. At the feasibility analysis stage, the goal feasibility is checked and any necessary refinements of the charts are carried out. The terminal phase activities use the reversibility principle and plan trajectories near the initial and final configurations. The mid-section phase deals with midway trajectory planning. For the terminal phase, forearm and boom planning iterate until a satisfactory boom tip location for starting the mid-section trajectory is found. For the mid-section, planning proceeds hierarchially. Boom trajectory is first planned using the primary charts alone. For portions of the boom tip locus that do not lie in the secondary chart, forearm planning is done. The separability principle is used in boom planning; the trajectory for the theta-phi joints is planned first and the r-joint is fixed next.

If a safe forearm trajectory cannot be found, the nature of the problem is identified and is used to revise the boom trajectory and another attempt at forearm planning is made. If the system is unable to come up with a safe trajectory even after a prespecified number of attempts, it resorts to a configuration switch. The same techniques are used to plan a trajectory to get the manipulator to the goal, this time however, in a different lateral configuration. If this also fails, the system gives up. Initialization of the environment and each of the three phases in planning is discussed in the next few paragraphs.

Note that planning incorporates simple strategies. It may so happen that the system fails to find a solution when there exists one in the real world. It is unlikely that such situations will be encountered except in some pathological obstacle configurations.

### 2.3.3 Initialization

The system is initialized with a description of the environment. The system uses the input polyhedra and generates primary and secondary obstacles for the left and right, secondary and primary maps. All the charts are generated for a default resolution. The regions of the charts will be further refined as and when necessary. The initialization needs to be done once for every new environment.

### 2.3.4 Goal Feasibility and Impossible Situations

Goal feasibility is done before planning begins. It includes boom placement and forearm placement safety checks. It determines whether the boom tip lies within a pasc of a primary region. If not the appropriate region is repeatedly refined until either the goal boom tip position is within a pasc or the resolution limit is reached and the system returns complaining that the goal is not feasible. The forearm feasibility study involves checking whether in the final configuration the forearm is safe from collision. If the forearm is not safe the goal is deemed not feasible.

During mid-section phase boom planning, the system keeps a watch for situations which would get the boom stuck. If the boom cannot be maneuvered out of an area, the system complains. Again, during forearm planning along a proposed boom tip locus, the system looks out for situations which would get the forearm stuck. The system requests a boom trajectory refinement if this happens. Such situations are called impossible situations.

### 2.3.5 Mid-section Planning

Boom Planning : Boom planning is separated into two phases. The first deals with a theta-phi space trajectory and the second fixes the trajectory in the r-dimension.

The system hypothesizes a boom tip locus that is linear in the theta-phi joints. A list of primary regions through which this trajectory passes is computed and certain minimal checks on the safe limit intervals of the regions in this list are made. If for example a region is impassable a fixed number of attempts are made to further refine the region. If the region is still impassable, subgoals are introduced to avoid this region. The heuristics minimize the number of subgoals and aim for subgoals in regions with large safe limit intervals. If the start or goal boom tip position is completely boxed in by impassable regions the system complains that the goal is not feasible.

The system next plans the r joint. Piecewise linear trajectories in the three boom joint angles are what is being attempted. Failure at any level results in a return back to the next higher level for replanning. Back at the topmost level the system tries a configuration switch. The initial choice is the same lateral configuration at the goal as at the start. The last choice is the alternate lateral configuration. Planning starts using the same strategies.

If this second attempt is unsuccessful, the system gives up.

Planning the r joint trajectory is done at three levels - region, sectoroid and the pasc level. At each level two steps are taken. Consider, for example, the two steps at the region level. We are interested only in those regions the boom tip locus passes through. The first step handles the problem of two adjacent regions having a disjoint (rmax, rmin) interval. The second step handles the problem of the locus lying outside the (rmax, rmin) interval of a region it passes through.

Having planned the boom trajectory, the forearm joint angle trajectory is planned. By definition of the charts and the restrictions imposed on boom trajectories, it is clear that the planned boom tip locus is a curve which is linear in r-theta-phi and which always lies within the regions of the primary chart. The first step in the forearm planning is to identify sections of this curve that do not lie within some secondary chart region. Only for these sections does forearm planning have to be done. The forearm planning along one such section is described next.

Forearm Planning : The primitives for forearm trajectory are two types of motions called the sphere and pgram motions. During sphere motion the boom is stationary and the forearm moves in the plane formed by the lines passing through the initial and final forearm configurations. It is obvious that during sphere motion the hand traces part of a circle.

The boom tip locus is approximated by a sequence of straight line segments in cartesian space. During pgram motion the boom is moving and the forearm moves along a straight line parallel to the boom tip locus. Pgram motion generates a parallelogram for a trajectory surface and hence the name. The trajectory surface is a plane for both types of motions. The circular boundary of the surface generated by sphere motion is replaced by straight line segments. These simplifications make collision detection and avoidance numerically tractable. The basic heuristic used for the forearm planning is to get the forearm in the wake of the boom and thereby decrease the chances of a forearm collision.

## 2.3.6 Terminal Phase Planning

Terminal phase planning uses obstacle configuration dependent heuristics and the nomenclature arises from the observation that near the start and the goal, obstacle configuration specific heuristics are most likely to be useful. As a consequence of the reversibility principle we need not distinguish between departure from start and approach to the goal. The strategies and heuristics for terminal phase planning are described next.

The terminal phase strategy consists of planning pairs of <u>adjust</u> and <u>move motions</u>. The adjust motion orients the forearm to reduce the chances of a collision during the subsequent move motion. A sequence of such pairs of motions puts the boom tip at a safe point, from which the mid-section strategies take over. A <u>safe point</u> is a point in a secondary pasc, or if there is no secondary pasc with a reasonable safe limit interval then it is a point in a primary pasc whose safe limit interval exceeds a prespecified value. Figure 2.3 shows one adjust-move pair motion for an example in two-dimensions. The adjust move (A) aligns the forearm with the dotted line. The subsequent move motion (B) retracts the boom tip to P2 from P1.

During <u>move motion</u> the boom tip moves along a line collinear with the forearm and away from the hand, and the forearm maintains its orientation in cartesian space (see Figure 2.3). This motion continues until either the boom tip reaches a safe point (and terminal phase planning is over) or a potential collision is recognized. In the latter case, the system proceeds with another adjust and move motion pair. At the end of every such pair of motions a check is made to see that progress is being made. If the manipulator joint angles remain unchanged, the system returns a failure.

The adjust motion orients the forearm to reduce the chances of a collision during the subsequent move motion. For this motion, the nature of obstacle configurations is more important than the nature of the obstacle itself. Thus the fact that the obstacles form a cave-like structure is more important for orienting the forearm than the fact that one of them is a prism. Obstacle configurations have been classified. For any boom tip location and forearm orientation the heuristics, associated with the configuration types, give by how much and in what direction, the forearm should move. The obstacle configuration of Figure 2.3 is called a 2D channel. The heuristics associated with a 2D channel suggest that the forearm should be aligned with the dotted line.

Terminal phase planning is one of the most expensive components of the safe trajectory planning problem. This component can be factored out and done at execution time using hardware proximity sensors. Section 9.3 describes the input/output charateristics of these sensors, and the logic that is required to analyze the sensory data.

## 2.3.7 Discussion

Boom planning deals with finding a path through the primary chart. Obstacles influence the trajectory only indirectly. Since charts represent free maneuverable space as such, I call this planning as <u>finding a safe trajectory through empty space</u>. This is to be contrasted with forearm planning where it is the obstacles that directly influence the determination of the trajectory locus. I refer to this sort of planning as <u>determining safe trajectories by avoiding collisions</u>. Note how these complementary views of safe trajectory planning have been used to advantage.

The introduction to this chapter mentioned cartesian space and joint space representations of obstacles and trajectories, and the conversion overhead problem. The main issue was to determine in what space should obstacles and trajectories be represented to make safe trajectory planning efficient. The representation and planning described in the last two sections provide the answer. Boom trajectory loci and charts (empty space) are represented in joint space, while obstacles and forearm trajectory loci are represented in cartesian space. This choice is very convenient. The best part of this solution is that the conversion overhead problem is also solved; only one conversion of the boom space straight line locus to cartesian space straight line locus is required.

## 2.4 COMPARISON WITH PREVIOUS WORK

Work on restricted versions of the collision detection and avoidance problem have been attempted by Pieper(1968), Lewis(1974) and Widdoes(1974). None of these earlier systems came anywhere close to handling the complexities illustrated in the example of Figure 1.1. Furthermore, the solutions were often plagued by computational inefficiencies. I will discuss the work of Pieper, Lewis and Widdoes under the topics of representation and planning.

### 2.4.1 Manipulator and Obstacle Models

The systems of Pieper, Lewis and Widdoes model the manipulator links as cylinders. In my solution the physical dimensions of the boom and the forearm can be accounted for by extending the sizes of the obstacles appropriately. For any given environment, the obstacles need to be enlarged only once while collision detection has to be done much more often. Since checking for collision of line segments with obstacles is computationally less expensive than detecting collision of cylinders with obstacles, my treatment of the manipulator as two line segments proves to be computationally better.

Widdoes and Pieper model obstacles as infinite planes, cylinders and spheres. Lewis models obstacles as prisms. These representations greatly simplify collision detection but often result in loss of valuable maneuverable space. Pieper and Widdoes would approximate a wedge by a sphere with diameter equal to the diagonal. Lewis would approximate the wedge by an enclosing prism. Both approximations can be quite crude depending on the nature of the wedge. My solution models obstacles by their enclosing polyhedra. A representation as polyhedra, with no limits to the number of faces will reduce the loss of maneuverable space. When the loss of maneuverable space is not critical, the obstacles can be represented as prisms.

### 2.4.2 Free Space Models

Lewis has no explicit representation of free space. Pieper defines the notion of a region. The workspace of the manipulator is divided into 64 equal parts called regions. Each region is a cube in cartesian space. Each region is associated with a list of objects that intersect with the region. The computation of the properties (the list of objects associated with the the region) needs to be done just once for any given environment.

Widdoes has a more elaborate representation of free space. He has four boom strategy world models. The first one is for the back of the boom. The next two are for the front of the boom - one for the initial forearm orientation and another for the final forearm orientation. The fourth one is a model for forearm transitions from the initial to the final orientation. Each model is a uniform two-dimensional grid in the first two joint angles. Each element of the grid is called a region and has associated with it an r-value. For the first model, the r-value denotes the minimum boom extension for which the region is guaranteed to be collision free. For the remaining models, the r-value denotes the maximum boom extension for which the manipulator is safe from collisions or for which the forearm transitions from the initial to final orientations is guaranteed to be collision free.

The models are generated by covering each surface of the obstacles with a mesh of points. Consider the generation of the back-boom model. The maximum extent of the boom for which the back of the boom will collide with the point is computed. Since the boom is modelled as a cylinder, the range of the first two joint angles for which the back of the boom will collide with the point is calculated. The computed distance is the r-value resulting from this object for all regions in the model which intersect the range of the first two joint angles. In the case of the back-boom model, for each region the minimum of the r-value contributions is chosen as the desired r-value. Similar computations are made for the other models.

The most severe limitation of this representation is that these models have to be recomputed every time the forearm configuration is changed. Further, the forearm transition model requires the initial and final orientations of the forearm to be known. The orientations may not always be known as is the case when more than a single maneuver is needed to get the forearm out of a complex obstacle configuration (see Figure 1.1, for example). In such cases the intermediate orientations need to be determined rather than be considered as given.

The idea of having an explicit internal representation of free space is Widdoes' best contribution. However, since he did not decouple the boom and the forearm, and because of his manipulator model the generation of the internal representations became computationally expensive.

In my solution free space models are a crucial ingredient of the primary and secondary problem spaces. It is these free space models which permit greatly simplified manipulator and trajectory envelope descriptions.

### 2.4.3 Trajectory Models

Lewis and Widdoes specified the time histories of the joint variables as polynomial sequences. What complicated their solution was that they placed no constraints whatsoever on the relationship between the different joint angles. Independent joint angle movement of cylindrical links resulted in complex 3-space trajectory envelopes to be generated when the manipulator moved. These volumes made trajectory planning very expensive. The independence of joint angle movement also forced Pieper, Lewis and Widdoes to end up checking the safety of the trajectory by checking the safety of the manipulator at a finite number of points along the trajectory. This scheme though safe enough in practice does not always guarantee a collision free trajectory. Furthermore, there is no way of knowing whether the trajectory is in a relatively obstacle free environment or a cluttered environment. Thus there is even no hope of achieving any saving by adjusting the placement of points at which collision checks are made.

Lewis suggests the storing of precomputed safe trajectories between commonly accessed arm positions and orientations. He recognizes that such a scheme will be a valuable addition to any collision detection and avoidance scheme but that it does not solve the problem.

In my solution the simplified manipulator descriptions, the concept of a trajectory locus and the primitive trajectory types that constitute a trajectory locus, all result in simple and numerically tractable trajectory envelopes.

### 2.4.4 Planning

Pieper and Lewis have an environment independent trajectory hypothesizing scheme. There is no notion of a trajectory locus. Planning results in a trajectory. Each joint of the manipulator is planned independent of the others. The safety of the manipulator is checked at a finite number of points along the trajectory and intermediate points are introduced to avoid any detected collision. Pieper uses clever strategies for selecting intermediate points but these strategies are not complete and they often introduce new problems causing the system to flounder.

Widdoes decomposes planning into boom and forearm planning. His free space models enable him to restrict his space of potential candidate trajectories to start with. Thus his trajectory hypothesis stage in planning is more sophisticated than those of Pieper and Lewis. His free space model computations and the use of optimization algorithms for picking out a good candidate from his starting set of candidates, however, are expensive affairs. Collision detection and avoidance by introducing intermediate points is very similar to those of Pieper and Lewis.

The planning in my system is more sophisticated in terms of the use of the general principles of hierarchy and separability. The planning heuristics, however, are quite simple and there is almost no searching in the traditional sense. Boom planning is treated as path planning through empty space, and forearm planning as that of collision avoidance. The goal is to plan safe trajectories. The underlying representations provide for the best possible choice between the two ways of looking at the problem : maneuvering in free space, and avoiding collisions.

## 2.5 SUMMARY AND CONCLUSIONS

The chapter began with a discussion of the trajectory planning problem and criteria the solution must satisfy. This was followed by a discussion of the computer representations of the entities in the universe of discourse for the three problem spaces and the use of these spaces in trajectory planning. Finally there was a brief description of the previous work on collision detection and avoidance.

To summarize, my solution, in contrast with the earlier attempts, has a large amount of knowledge - about trajectories, obstacles, manipulators and free space - built into the programs. The planning system makes good use of the powerful principles of hierarchy and separability. The planning heuristics, however, are simple but yet effective in practice and their efficacy and ease of use is due to the underlying representations.

As mentioned in section 1.2.2 a slight modification to the solution for the mechanical manipulator provides a solution to the humanoid manipulator problem. The solution for the humanoid arm is the same as the general solution with the following differences

1) For a given position and orientation of the hand there are essentially four solutions to the ˆ0

Table 2.1 The Representation Hierarchy

| ELEMENTS IN WORLD / PROBLEM SPACE | MANIPULATOR | OBSTACLES/ ENVIRONMENT | EMPTY SPACE | TRAJECTORY ENVELOPE |
|---|---|---|---|---|
| SECONDARY | Single Point (Boom Tip) | | Secondary Region/ Secondary Chart | Line |
| | Single Line Segment (Boom) | Secondary Obstacle/ Secondary Map | Secondary Free Space | Surface (Boom) |
| PRIMARY | Single Line Segment (Forearm) | | Primary Region/ Primary Chart | Surface (Forearm) |
| | Two Line Segments (Boom and Forearm) | Primary Obstacle/ Primary Map | Primary Free Space | Two Connected Surfaces (Boom and Forearm) |
| REAL | Two Solid Segments (Boom and Forearm) | Polyhedra/ Map | Maneuverable Space | Two Connected Solids (Boom and Forearm) |

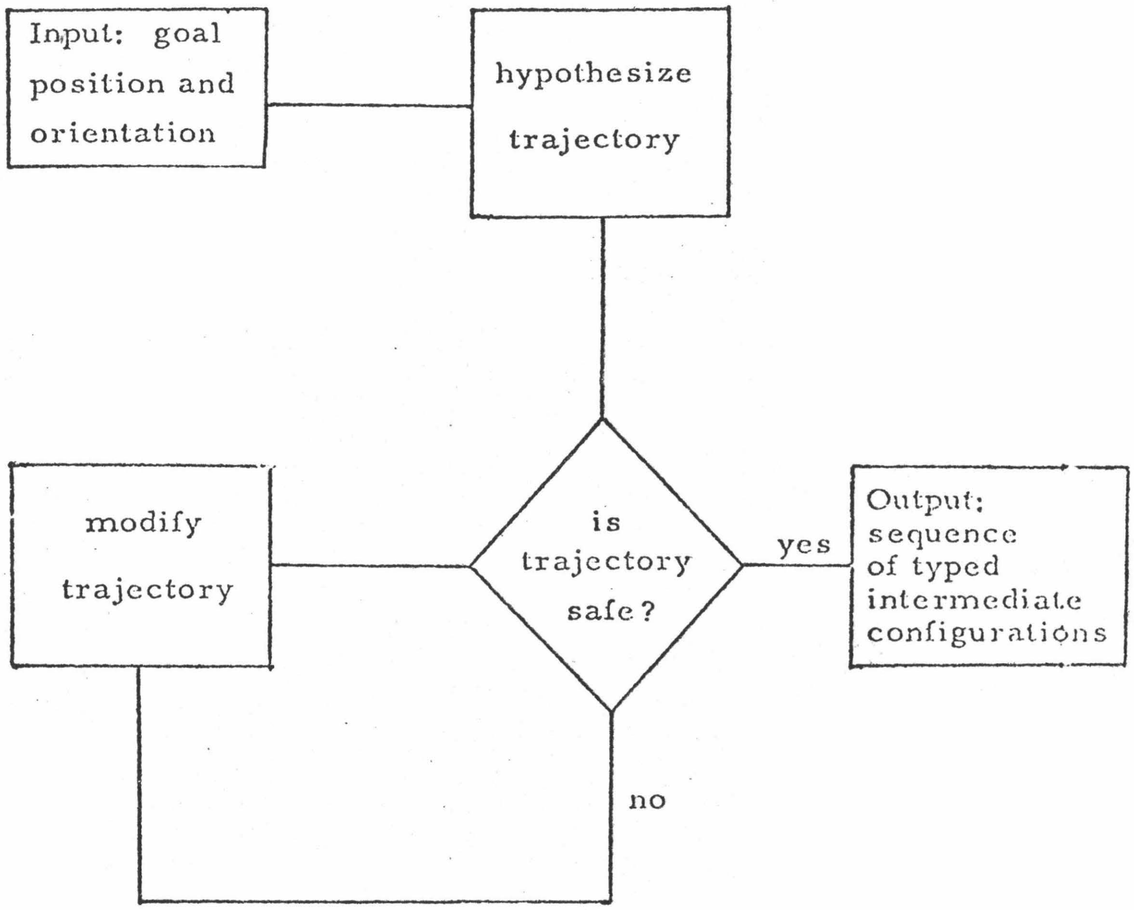

Input: goal position and orientation
hypothesize trajectory
is trajectory safe?
yes
Output: sequence of typed intermediate configurations
no
modify trajectory

Figure 2.1 Safe Trajectory Planning

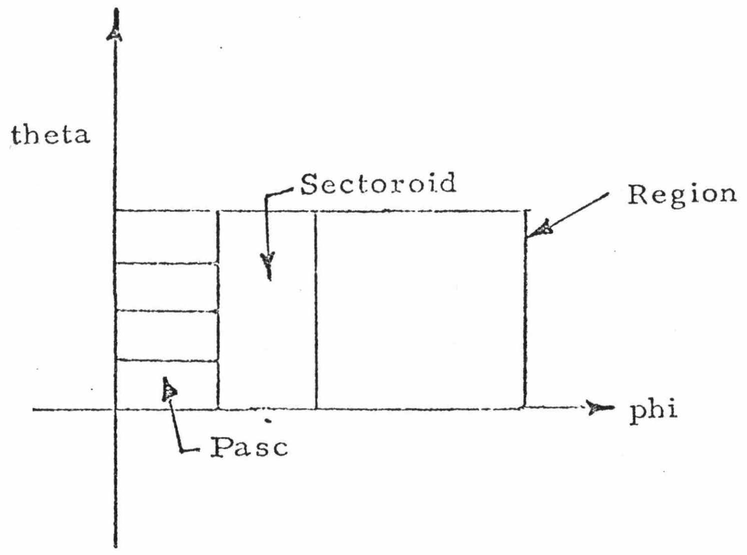


Figure 2.2 The Structure of Regions

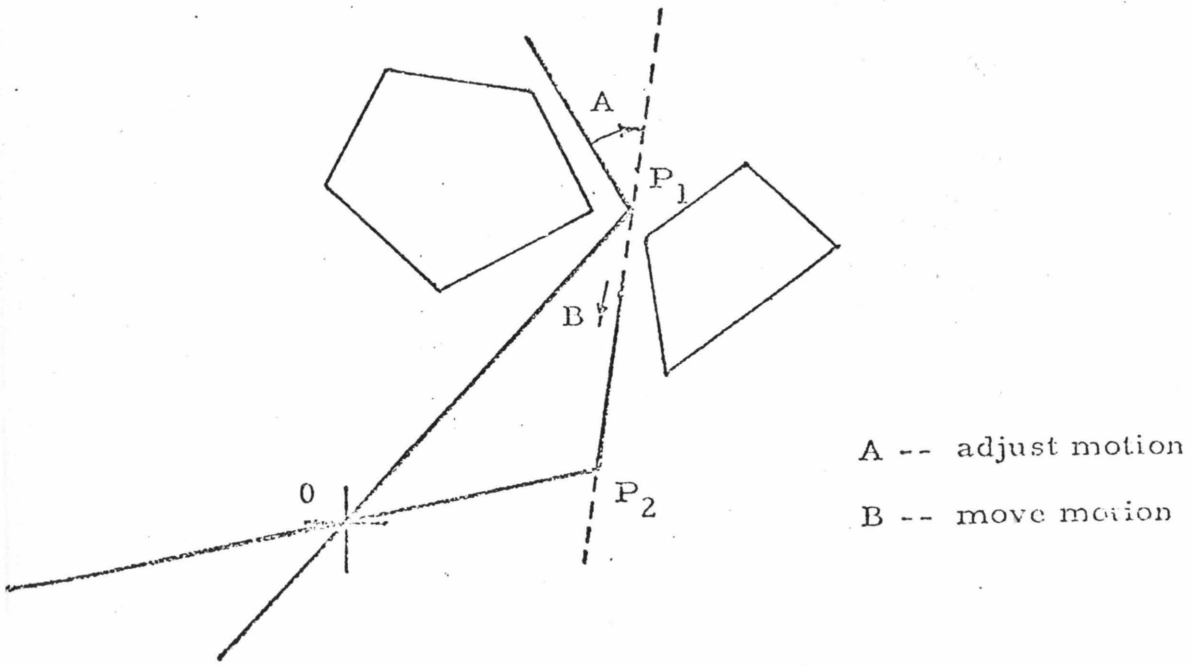


Figure 2.3 Terminal Phase Planning

# CHAPTER 3
# MANIPULATOR MODELS

This chapter presents models for a computer controlled manipulator and their relation to obstacle avoidance problems.

## 3.1 THE 2D MANIPULATOR

Figure 3.1 illustrates a 2D manipulator, a three link and three joint structure. The first of the links is called the shoulder; the shoulder is a line of fixed length and rotates about the origin. The next link is called the boom; the boom is a line that slides back and forth at the tip of the shoulder. The final link is called the forearm and the forearm is also a line that can rotate about the tip of the boom. The 2D manipulator has three degrees of freedom corresponding to the rotational capabilities of the shoulder and forearm and the sliding joint at the shoulder-boom connection. When looking along the boom at the forearm, the manipulator is either on the right or the left side of the shoulder. This gives rise to the notion of a right-handed and left-handed manipulator respectively, and is called the lateral property.

There is an equivalent description of the manipulator for trajectory planning purposes. This description is possible because the shoulder link is of fixed length and can rotate about the origin. The new model is obtained by replacing the shoulder link by a circle, called the post-circle. The post-circle has its center at the origin and its radius equal to the length of the shoulder link. As before the boom slides back and forth. In addition it rotates about the origin always remaining tangential to the post-circle. The number of essential links in the manipulator has been reduced to two; the boom now has two degrees of freedom and the forearm, as before, has a single degree of freedom. The lateral property is still valid for the alternative representation.

The advantage of this alternative formulation of the manipulator description is that it permits a useful simplification. The post-circle may be reduced to a single point. The boom then slides in and out and rotates about the origin and there is no longer any distinction between a right-handed and a left-handed manipulator. The 2D manipulator with the post-circle radius equal to zero is called a simplified 2D manipulator (see Figure 3.2). The collision detection and avoidance problem for the simplified 2D manipulator has all the essential characteristics of the problem for the general 2D manipulator and yet it is devoid of unnecessary complications. The rotational joints of the boom and forearm are called _theta_ and _phi_ joints respectively and the sliding joint of the boom is called the _r_ joint. Figure 3.2 shows these joint variables.

A set of computer programs implementing the solution to the collision detection and avoidance problem for the simplified 2D manipulator was written and tested. The computer representation of the manipulator has the r, theta and phi values, and several other attributes of the manipulator such as the x-y coordinates of the boom and forearm tips are also saved.

## 3.2 THE 3D MANIPULATOR

Since the simplified 2D manipulator is easier to visualize, I will first describe its three-dimensional analogue called the _simplified 3D manipulator_ (see Figure 3.3). The simplified 3D manipulator has two links. The boom and the forearm are both straight lines and have three degrees of freedom each. The three degrees of freedom of the boom correspond to the _r, theta and phi_ variables in a spherical coordinate system. The boom passes through the origin and its three degrees of freedom permit the boom tip to be positioned anywhere within a sphere with center at the origin. Two of the three degrees of freedom of the forearm let its tip trace the surface of a sphere with the boom tip at the center of the sphere. These two rotational joints of the forearm are called f theta and f phi respectively (the prefix f indicating forearm values). The last of the three degrees of freedom of the forearm, called f psi, lets it turn about its axis. For collision detection and avoidance purposes, this last degree of freedom is superfluous when the manipulator is not handling large objects.

For the three dimensional analogue of the 2D manipulator, the alternative representation of the 2D manipulator will be generalized. The post-circle now becomes a cylinder with its axis along the z-axis and is called the post-cylinder. The 3D manipulator again has two links, the boom and the forearm, each with three degrees of freedom. The boom slides back and forth; it rotates about the z-axis always remaining tangential to the post-cylinder with the tangential point lying in the X-Y plane; it goes up and down (theta variation) again remaining tangential to the post cylinder and the tangential point lying in the X-Y plane. The notion of left-handedness and right-handedness is valid for the 3D manipulator too. The 3D manipulator is right-handed if, when looking at the forearm along the boom, the manipulator is on the right side of the post-cylinder; otherwise, the manipulator is said to be left-handed.

Since the post-cylinder is fixed it is treated as an obstacle. The post-cylinder is replaced by a hypothetical infinite cylinder of radius equal to the distance of the axis of the boom from the axis of the post. Note that for the general 3D manipulator too, the boom and the forearm are straight lines.

The collision detection and avoidance system for the 3D manipulator has been implemented. I have essentially followed Lewis(1974) in solving the position problem for the 3D manipulator. The main differences are

1) Joint 2 or theta lies between 0 and pi and not between - pi and + pi. Lewis treats negative joint 2 values to imply right-handed configurations and positive values to imply left-handed configurations.

2) Joint 1 or phi values are not identical for the left- and right- configurations. Lewis has them identical.

3) The lateral property of the manipulator is explicitly represented and is not left as just the sign of an angle. As in the 2D manipulator, in addition to the joint angles, several other attributes describing the manipulator are included.

## 3.3 THE SCHEINMAN ARM

The 2D and 3D manipulators described above are abstractions of a class of computer controlled manipulators. The Scheinman arm shown in Figure 3.4 is an example of this class [Dobrotin and Scheinman(1973), Lewis(1974)]. It is a six degree freedom device allowing the forearm tip or the hand to be positioned anywhere and with any orientation within the limits of the joint angles. This is the manipulator that is used on the JPL robot (see Appendix 1). Abstractly, such a manipulator may also be described in terms of links and joints. There are six links, each connected to the next by a joint. Figure 3.5 shows the different joints and the coordinate frames for describing the link-joint pairs of the Scheinman arm. There are two kinds of joints, prismatic or sliding, and revolute. Link1 is called the post. Link2 is called the shoulder. Link3 is called the boom. Link4 and link5 are non-existent because the manipulator design is such that there are three revolute joints meeting at the tip of the boom. Link6 is the forearm. Except for joint 3, which is prismatic all the joints are revolute. Joint1 is called phi, joint2 is theta, joint3 is r, joint4 is f theta, joint5 is f phi and joint6 is f psi. The prefix "f" indicates that the angles refer to the forearm.

The links in the Scheinman arm are all solid objects. When the manipulator moves its links trace out a volume in three space called the <u>trajectory solid</u>. The manipulator colliding with an obstacle implies that the space occupied by the obstacle intersects with the trajectory solid. Collision detection involves checking whether the trajectory solid intersects with every obstacle in the environment. Greater the complexity of the shapes of the obstacles and the trajectory solid, the more expensive is the intersection check.

The intersection checks can be simplified if the manipulator links are treated as straight lines by applying some transformation to the obstacle shapes and sizes. If k is the radius of a cylindrical envelope of a manipulator link, then, for collision avoidance all we need is that the axis of the link be at least a distance k away from the nearest obstacle surface. If by application of an enlarging transformation to obstacles, every point on the surface of the enlarged obstacle is made at least a distance k away from the nearest point on the original obstacle, then for obstacle avoidance purposes, the manipulator links may be treated as just straight lines having no physical dimensions.

When a manipulator with straight line links moves in three space, a surface called the trajectory surface, is generated. Checking for intersection of a surface and a volume is computationally simpler than checking for intersection of two volumes. Furthermore, the enlarging transformation needs to be applied only once and from then the enlarged obstacle descriptors alone need to be used. This enlarging transformation is described in detail in Chapter 4.

Note that when the radius of the minimum bounding cylinders for the boom and forearm of the Scheinman arm is reduced to zero we get the generalized 3D manipulator. If obstacles are enlarged appropriately, the links on the Scheinman arm can be considered as straight line segments. Hence, from now on, for the 3D system we will be considering only the generalized 3D manipulator.

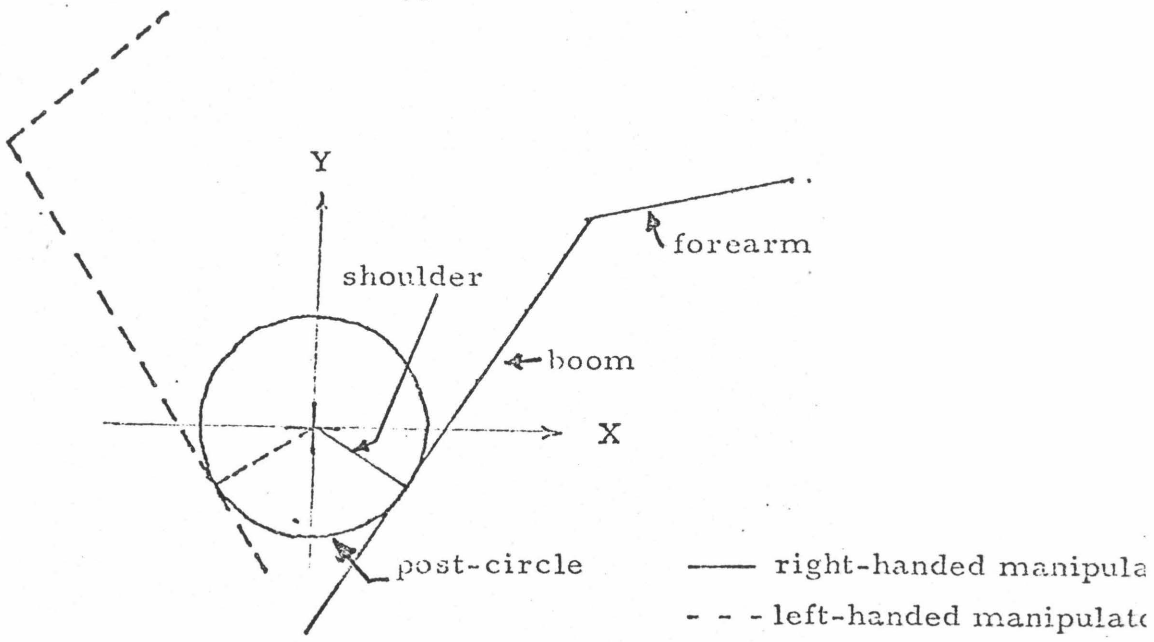


Figure 3.1 The 2-D Manipulator

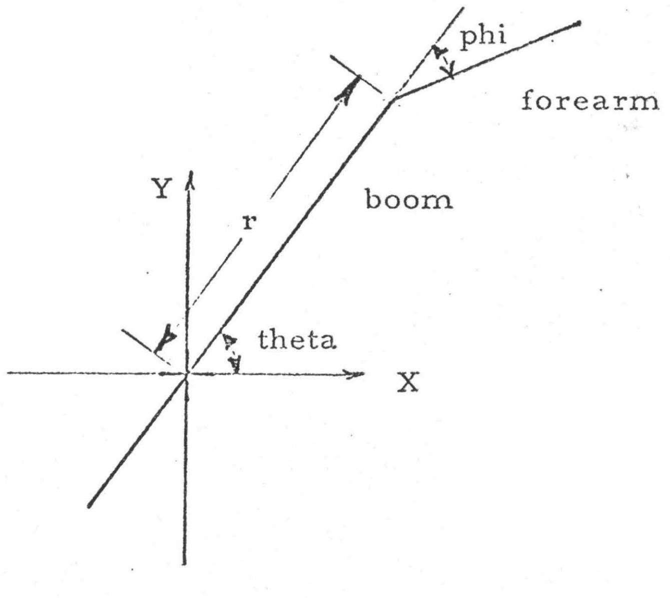


Figure 3.2 The Simplified 2-D Manipulator

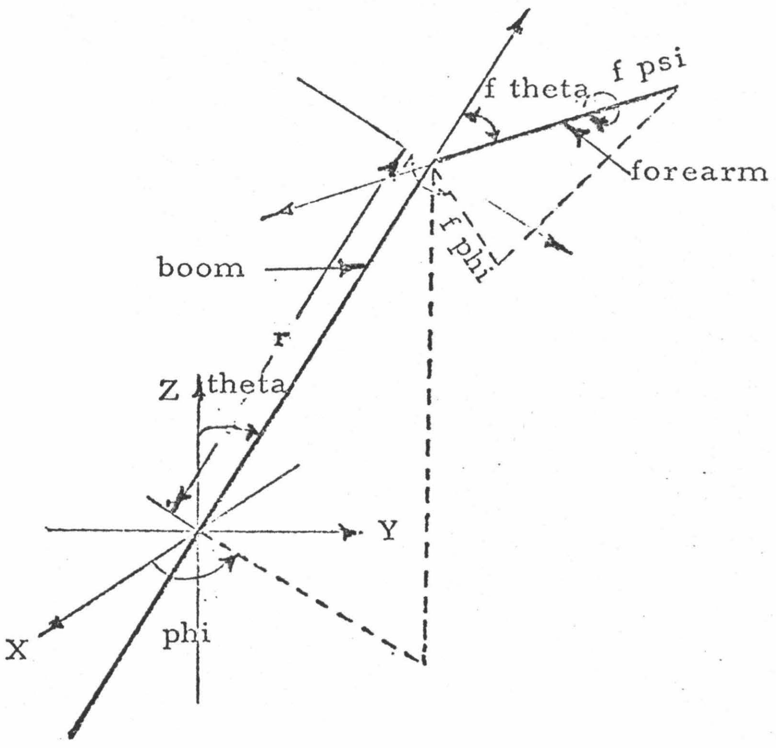


Figure 3.3 The Simplified 3-D Manipulator

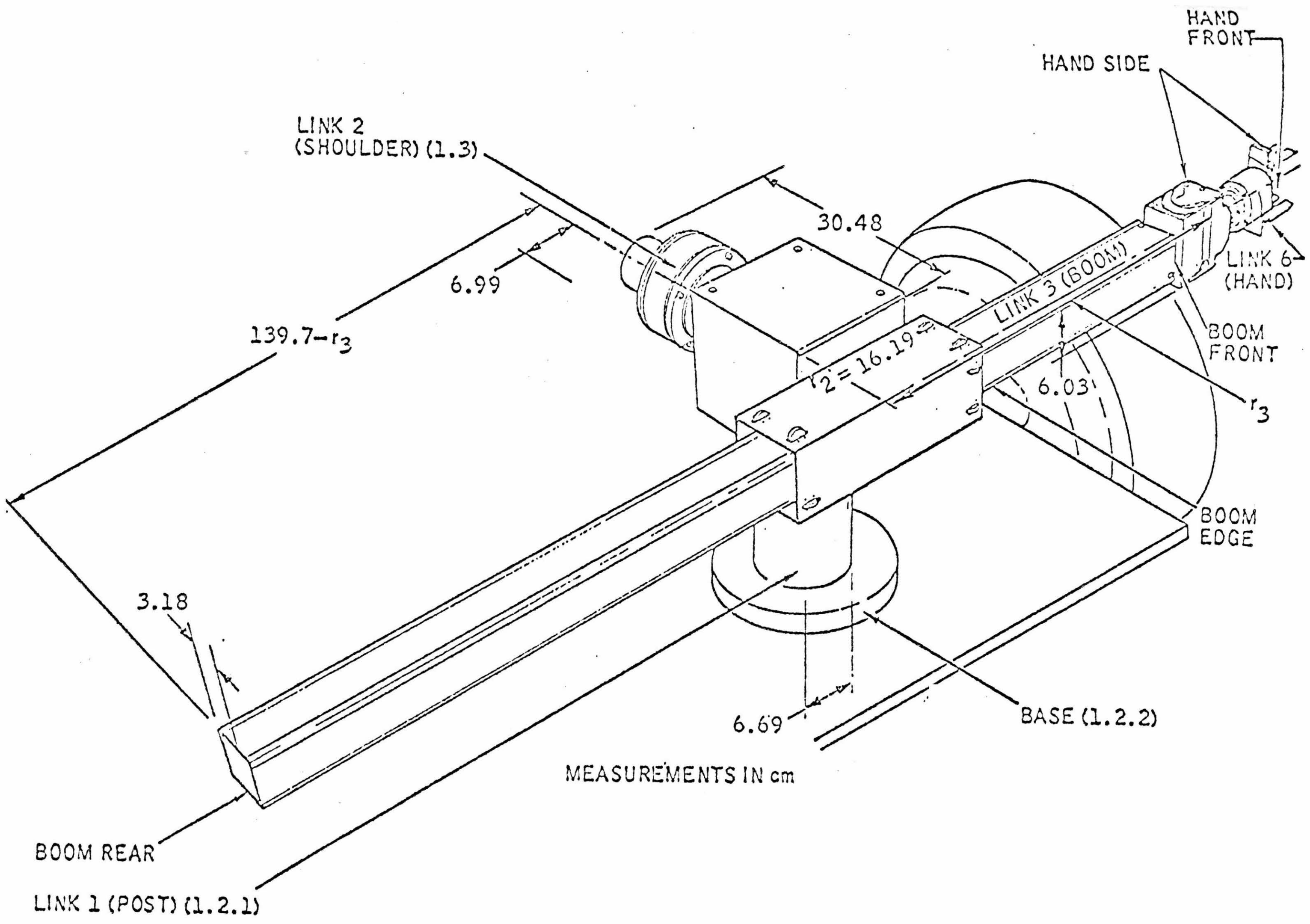


Figure 3.4 The Scheinman Arm

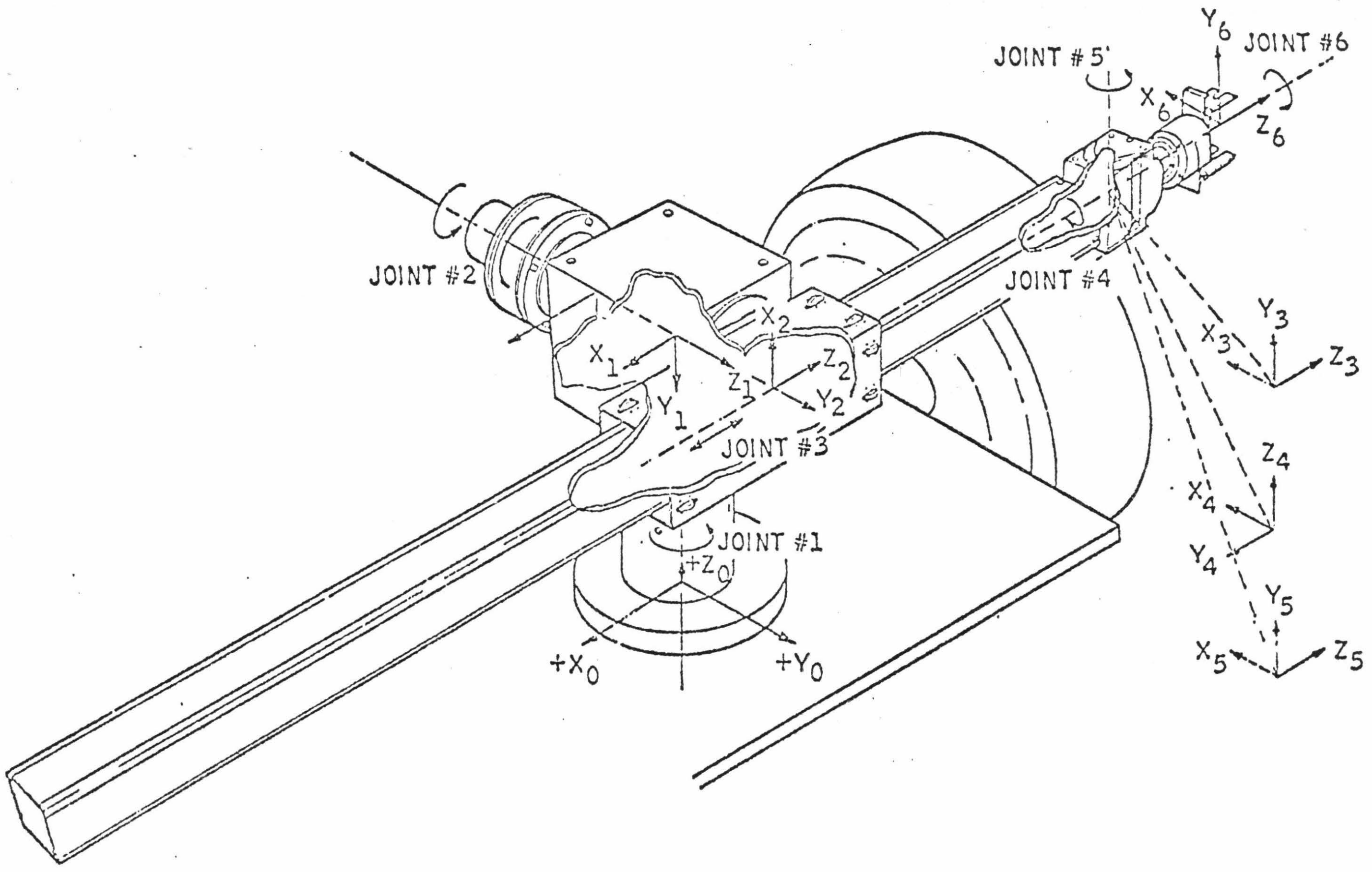


Figure 3.5 Reference Frames for Link-Joint Pairs of Scheinman Arm

# CHAPTER 4
# ENVIRONMENT DESCRIPTION

In a broad sense the environment will include the manipulator, the obstacles, the objects that are to be manipulated and the free space within the reach of the manipulator. The representation of the manipulator has already been described. Free space representation will be described in the following chapter. This chapter will be concerned only with the representation of obstacles. From here onwards, the term <u>environment</u> will be used in a restricted sense to describe the set of obstacles in the workspace of the manipulator.

The manipulator operates in a <u>static</u> <u>environment</u>. The manipulator is the only active agent. The trajectory planning system can be initialized to plan in different environments. In between such initializations, the environment is assumed to remain unchanged except for the changes brought about by the manipulator's actions, such as moving objects around. Furthermore it is assumed that the environment initializations are infrequent in comparison with the number of trajectories planned. This was referred to earlier as the <u>infrequent</u> <u>environment</u> <u>initialization</u> <u>hypothesis</u>. With this assumption it becomes reasonable to spend some computational effort in generating internal representations of the environment which can simplify trajectory planning. To put it differently, since trajectory planning will be performed more often than environment initialization, any transformation of the environment descriptions to reduce the computational load on the trajectory planning routines is justified. The infrequent environment initialization assumption is justified in practice. The scenarios envisioned for the JPL robot always require the robot to get to a site, explore the site using scene analysis and then perform a variety of tasks that involve manipulation. These tasks include picking up and transporting rock and soil samples, deploying scientific instruments etc. The number of trajectories planned easily outnumbers the number of sites visited.

## 4.1 THE NATURE OF OBSTACLES

Obstacles will be both regular and irregular in shape. Man-made objects on the robot (Figure A1.1) such as the platform, the wheels, the wheel motors, the TV rack etc. are regular objects and can be described as cylinders, parallelopipeds, or as unions of these shapes. Irregular objects would primarily be boulders and rocks and the manipulator may need to maneuver near them. Any representation of obstacles that is chosen should be capable of handling these irregular shapes. Since compactness of representation is crucial, the irregular shapes will have to be approximated by regular shapes. The important questions are what regular shapes to use for the approximation and how good an approximation is essential.

In the discussion on manipulator models I indicated how collision detection is performed; it is done by simulating the motion of the manipulator in cartesian space and checking whether the area swept, by the manipulator links during motion, intersects with the volume occupied by the obstacles. The numerical complexity of these intersection checks increases with the complexity of the shapes of trajectories and obstacle surfaces. Considerable simplicity can be achieved by using polyhedra - plane faced objects - to approximate obstacles. Lines and planes are represented by linear equalities and handling linear expressions is numerically very simple. If there is no restriction on the number of faces of a polyhedra and if both concave and convex polyhedra are allowed, a three-dimensional object can be approximated to any arbitrary degree of accuracy by a polyhedron.

How accurate a polyhedral approximation to a regular or irregular shape should be depends on a number of factors. From the view of collision detection and avoidance, the storage requirements and the time required to analyze an obstacle will both increase with the number of faces of the polyhedron, or equivalently, with a better approximation of the irregular obstacle. However, loss of maneuverable space decreases with a better approximation. If the manipulator will not be maneuvering near an obstacle, the loss of maneuverable space will not hurt and a crude approximation will suffice. If maneuverability is crucial, the price for better approximation will have to be paid.

For the current 2D and 3D systems, obstacle descriptions are input by a human, and the human decides about the degree of approximation. It is not inconceivable that scene analysis programs with some knowledge about the goals of the manipulator will be doing this in the future.

Obstacles are represented at different levels of detail. In addition to a polyhedral representation, an obstacle also has a description of its envelope, a parallelopiped in (r, theta, phi) coordinates. The non-intersection of a trajectory surface with the enclosing parallelopipeds clearly eliminates the obstacle from further considerations. Intersection of the two does not imply anything definite and a more careful check is called for. The details of the representation of obstacles in 2D and 3D are presented in sections 3 and 5 of this chapter.

## 4.2 PRIMARY AND SECONDARY MAPS

Obstacles are represented by their approximating polyhedra. These polyhedra may have any number of faces and may be concave or convex. The set of obstacles in the workspace of the manipulator is called a map.

In Chapter 3 on manipulator models, I said that the manipulator links can be represented by straight lines if obstacles in the environment were enlarged by the radius of the manipulator link. A primary obstacle is an obstacle obtained by enlarging a real obstacle description by the radius of the manipulator links. The set of primary obstacles is called the primary map. Figure 4.1 is an example of a primary map for the two-dimensional problem. The polygons in the figure represent primary obstacles. The details of the enlargement transformation are given in sections 3 and 5.

It turns out that planning the manipulator trajectory for the first three joints, or the boom trajectory, is computationally simpler than planning trajectories for the last three joints, the forearm trajectory. Checking for forearm collisions is what makes the planning expensive. If possible we would like to avoid having to check the safety of the forearm. This should be possible when maneuvering away from obstacles, where there are large chunks of free space. An accurate characterization of regions away from obstacles is obtained by introducing the notion of a secondary obstacle. A _secondary obstacle_ is a primary obstacle enlarged by the length of the forearm. The set of secondary obstacles is called the _secondary map_. The importance of the secondary map arises from the following observation. Suppose that in the initial and final configurations, the manipulator does not collide with any secondary obstacle. Suppose further that there exists a boom trajectory, from the initial to the final configuration, that is safe from collisions with any secondary obstacle. The forearm can then be guaranteed to be free of collisions all along the proposed trajectory, independent of its orientation.

It was mentioned in section 2.2.2 that it would be desirable to have a transformation whereby one could avoid having to consider the lateral property of the manipulator at each step of the planning process. Appropriate one-time-only transformations are used to generate a _left primary map_ and a _right primary map_ and a _left secondary map_ and a _right secondary map_. The polyhedra descriptions in these maps reflect the manipulator configuration characteristics.

## 4.3 2D ENVIRONMENT

This section describes the implementation of the environment of the simplified 2D manipulator system. The data structures are described first and the descriptions of the operations that can be performed on them follow. Decomposition, dilation and enlargement are the three operations that are described. Finally the format of the input obstacle descriptions is discussed.

### 4.3.1 Data Structures

Obstacles are represented by closed polygons (Figure 4.1). Their descriptions are given in a right-handed coordinate system with origin at the base of the simplified 2D manipulator. Obstacle description includes a fixed set of attributes and a body. The body is represented by a linked list. A linked list representation is essential because the decomposition operator chops up the obstacle, resulting in the body being rearranged, and this rearrangement is easily carried out by pointer adjustments.

The fixed attributes of the obstacle include a description of its (r, theta) envelope, the coordinates of the center of gravity, the number of entries in the body of the polygon and the type. Since the theta values have a period of 360 degrees, the ordering of the real numbers cannot always be used to decide whether a given angle is greater or less than another angle. A special ordering which takes the circularity into consideration has been defined. The details of this circular ordering definition are given in Appendix 2. The envelope description includes the rmin, rmax, theta-min and theta-max values. The type of the polygon indicates whether the origin is inside the polygon, on an edge, at a corner of the polygon or outside the polygon. Primary and secondary obstacles are obtained by enlargement transformations and therefore may have the origin on their inside or on the boundary.

The body is a linked list of edge-corners. The order of these edge-corners is the order in which they are encountered when traversing the boundary of the polygon in an anticlockwise manner. Each edge-corner has a pointer to the next edge-corner and descriptions of the corner and edge that form the edge-corner. The description of a corner includes its cartesian and polar coordinate values and the slope of the line joining the origin to the corner. The descriptors of an edge consist of coefficients of the equation of a line collinear with the edge. The signs of these coefficients are so adjusted that points interior to the polygon are on the positive side of the edge.

### 4.3.2 Operations

The decomposition transformation chops up a concave polygon into disjoint convex components whose union gives the concave polygon. Figure 4.2 is Figure 4.1 redrawn with concave polygons decomposed into convex polygons. The dotted lines inside the concave polygons show where the decomposition was effected. The main idea is as follows: Check every corner of the polygon. If all the corners are convex the polygon is convex. Otherwise at the first concave corner, extend the edge of the previous edge-corner to cut the polygon into two parts. Apply the decomposition transformation to the two parts recursively. Since the sibling polygons have fewer corners than the parent polygon, the recursion terminates. The algorithm that has been implemented in the 2D system is an iterative version of the recursive algorithm described above. The iterative version is much faster but the careful book-keeping that needs to be done complicates its description.

The enlargement transformation enlarges an obstacle. Given a polygon OBS and a distance k, the procedure returns a new polygon NOBS. Every point on the boundary of NOBS is at least a distance k away from its nearest point on the boundary of OBS. Enlarging convex polygons is simpler than enlarging concave polygons. Hence all concave polygons are replaced by their convex subcomponents and then the enlarging transformation applied.

The simplest scheme for enlargement is shown in Figure 4.3 which shows the original polygon OBS and the enlarged polygon NOBS. NOBS is generated by drawing a line parallel to every edge at a distance k and outside OBS. NOBS has as many edges and corners as OBS. The main shortcoming of this scheme is that the distance d between corresponding corners P, P' say, is given by

$$d = k / \sin( A / 2 )$$

where A is the internal angle between the edges meeting at P. As A gets smaller d increases. Large values of d will result in making unavailable useful maneuverable space.

A refinement of the earlier scheme is shown in Figure 4.4. In addition to drawing lines parallel to edges of OBS, lines are drawn perpendicular to the angular bisectors of the internal angles of the polygon. These lines are drawn outside OBS and at a distance k from the corner. The refined scheme gives twice the number of edges and corners in NNOBS as in OBS. NNOBS will need approximately twice the amount of storage for its attributes as NOBS but the loss of maneuverable space is reduced considerably.

Dilation is the last of the transformations that may be performed on a polygon. Dilation requires the following to be done for each edge of the polygon. If the foot of the perpendicular from the origin to the edge falls inside the edge then the edge is split into two at the foot of the perpendicular; a new edge-corner is introduced into the body of the polygon at the foot of the perpendicular, and the size of the polygon incremented by one. Computing intersection of the given edge and a line through the origin and perpendicular to the edge, and determining whether the point is interior to the edge is straightforward. Figure 4.5 shows a polygon and two virtual corners P and P' that get added as a result of dilation.

Given any finite segment of a straight line, the point on it and closest to the origin is one of the following:

a) the foot of the perpendicular from the origin to the line, if the foot is inside the finite line segment, or

b) one of the two end points.

Determining the point, that is closest to the origin and lies on the section of an edge, is done repeatedly by routines that refine and generate navspace approximations (described in Chapter 5). Dilation simplifies this computation by eliminating one of the two possible choices. As a result of dilation, no edge has the foot of the perpendicular from the origin strictly inside it. The desired point is, then, one of the end points of the section.

Dilation is another example of a one-time transformation which eliminates repeated computations. This elimination, however, has not been achieved without overheads. Apart from the overhead of computing the transformation, run-time storage requirements for obstacle descriptions will be higher.

### 4.3.3 Input specifications

The input consists of the number of obstacles followed by obstacle specifications. Obstacle specifications include the number of corners (or edges) followed by cartesian coordinates encountered on an anticlockwise traversal of the polygon boundary.

## 4.4 EXTENDING 2D IDEAS

In 2D obstacles are represented by closed polygons. These polygons may be concave or convex and may have any number of sides. Three transformations - decompose, enlarge and dilate - were defined for the polygons. A natural extension of the representation to three dimensions is to use polyhedra to model obstacles. Of course the structure of a polyhedron is much more complex and the storage requirements for storing the attributes of a polyhedron much more severe. Furthermore, some difficulty may be anticipated in the 3D versions of the three transformations decompose, enlarge and dilate. These extensions are described in the following paragraphs.

In 2D the <u>structure</u> of the polygon is described implicitly by the ordering of the linked list representation of the body of the polygon. In 3D we have to describe not just the polygonal faces but also the relationship between the faces. A variety of representations are possible (see for example, Newman and Sproull (1973), part IV). The problem is to determine which of the representations is the best. A knowledge of the 2D obstacle representation is not of much help. The solution to the 2D problem, however, had clearly identified how obstacle descriptions are used in trajectory planning - for computing charts (see Chapter 4) and for forearm planning (see Chapter 8). The representation which facilitated these computations was chosen and it is described in section 4.5.

A 3D <u>decomposition</u> operator exactly analogous to the 2D decomposition operator can be defined. Whenever two faces meeting at an edge form an interior angle greater than 180 degrees, one of the planes is extended to cut the polyhedron into two. The decomposition operator is then applied recursively to the two parts. This solution is neither elegant nor simple since very extensive book-keeping needs to be done when the two sibling polyhedra are generated. Rather than implement such a scheme, I decided to leave it to the human entering the environment description to decompose any concave polyhedra into convex subcomponents.

The enlargement transformation again has no simple 3D counterpart. In the 2D case, enlargement resulted in a polygon with twice the number of sides, one for each corner and edge of the original polygon. In the 3D a similar solution will result in thrice the number of faces in the new polyhedron, one for each face, edge and corner of the original polyhedron. The storage requirements for such a polyhedron are very severe. Furthermore, in the 2D case it is very easy to verify that the enlarged polygon is well defined. This is not so for the 3D case. As faces of the new polyhedron are generated one of them may "chop off" a face, corner or an edge that was generated earlier.

A simple scheme that avoids these problems is one that results in a new polyhedron each face of which is parallel to one face of the old polyhedron. This scheme is well defined and results in a polyhedron whose storage requirements are significantly less (since there are less number of faces, edges and corners) than the generalized enlargement would produce. The only drawback of the simple enlargement transformation compared to the general one is that the loss of maneuverable space is large whenever the original polyhedron has a sharp corner or two faces meeting at a small angle.

There is no direct generalization of the 2D dilation operator. The 3D version of <u>dilation</u> introduces a <u>virtual corner</u> whenever the foot of the perpendicular from the origin to a face or an edge falls inside the face or the edge. Unlike in the 2D case where dilation results in the introduction of a new edge-corner, the 3D dilation results only in the addition of a virtual corner to the description of the face or the edge. A virtual corner is a foot of the perpendicular from the origin to a face or an edge. It has the same attributes as any other corner and it differs from a corner in that it lies either strictly inside an edge or strictly inside a face.

## 4.5 3D ENVIRONMENT

This section presents the implementation of the obstacle descriptors for the environment of the full-fledged 3D manipulator. The data structures are described first, the operations next and finally the input specifications.

### 4.5.1 Data Structures

Obstacles are described by closed convex polyhedra. The human who inputs the environment data has to decompose concave polygons into convex components. Obstacles are described in a right-handed coordinate system with the origin at the center and top of the manipulator post. The orientations of the axes are as shown in Figure A.1. Obstacle descriptors include a fixed set of attributes, and a description of corners, edges and faces. The abstract data structures will be described first. The specific implementation decisions, wherein packed data structures and scaling transformations are extensively used, will follow.

The fixed attributes of an obstacle include the following

1) center of gravity,

2) description of its (r, theta, phi) envelope,

3) the number of corners, edges, virtual corners and faces,

4) pointers to the start of the descriptors of the corners, edges, virtual corners and faces, and

5) obstacle type.

The envelope description includes the minimum and maximum r, theta and phi (for both left- and right- handed configurations) values. As in the 2D case the circular ordering relation defined in Appendix 2 is used.

The obstacle type (table 4.1) is one of the following : free, bound, support or cover. An obstacle is of type <u>bound</u> if the origin is inside the obstacle, <u>cover</u> if the obstacle is above the X-Y plane and the Z-axis passes through it, <u>support</u> if the obstacle is below the X-Y plane and the Z-axis passes through it, and <u>free</u> otherwise. Obstacles of type support and cover have phi-ranges of 360 degrees and theta ranges of (theta, 180) and (0, theta) respectively for some theta. Bound obstacles have a 360 degree phi-range and 180 degree theta-range, while free obstacles have phi and theta ranges of less than 180 degrees.

TABLE 4.1 OBSTACLE TYPES

| Obstacle-type | Phi-range | Theta-range |
|---|---|---|
| Free | < 180 | < 180 |
| Support | 360 | (Theta, 180) |
| Cover | 360 | (0, Theta) |
| Bound | 360 | 180 |

Primary and secondary obstacles are obtained by enlargement transformations and therefore may have the origin on their inside or on the boundary. Virtual corners were defined in section 4.4. As in the 2D case, the point on an edge or a face that is closest to the origin is of interest. Instead of having to compute this information repeatedly it is best to compute it once and save the results of the computation as virtual corners of the polyhedron.

The fields describing a corner are the (x, y, z) coordinates and some joint angle values. The joint angles correspond to the r, theta and phi values of the boom if the boom tip were positioned at the corner. There are two phi values corresponding to the left-handed and right-handed manipulators.

Chapter 2 described left primary and left secondary maps and right primary and right secondary maps. In the implementation there is only one primary map and only one secondary map. Left-phi values are used when the manipulator is left-handed and right-phi values are used when the manipulator is right-handed. All other characteristics of obstacles are identical for the manipulator configurations.

The attributes of interest for an edge are its two end points, its length, the normalized direction cosines of the the edge and a field indicating whether there is a virtual corner on the edge. If there is a virtual corner, the field is a pointer to the virtual corner or else it is zero. The two end points define a direction of the edge, from the first to the second corner.

The description of a face has a body and the following fixed set of fields: the center of gravity of the polygon, the normalized direction cosines of the normal to the face and the distance of the face from the origin, the (r, theta, phi) envelope of the face, the number of edges and corners on the face, a pointer to the neighboring face and the type of the face. The signs of the direction cosines and distance are adjusted so that points interior to the polyhedron are on the positive side of the plane. The envelope of the face is similar to the envelope of the polyhedron. The body of the polygon consists of a sequence of e-c-entries. An e-c-entry has a direction, an edge pointer and a corner pointer. The sequence of edge and corner pointers enumerates the boundary of the polygon in order. The direction of an e-c-entry specifies whether the edge is traversed along or against its defined direction.

There is no order imposed on the organization of the corners and edges and virtual corners. However, the polygon entries are sorted by the minimum r value of the polygon envelope. The faces are therefore saved as a linked list and the start of the face list points to the face with the least r value.

Some comments on the implementation of the above data structures: Many of the fields of the various data structures described above require less than a full word descriptor. Since storage requirements are critical a packed representation is used. Some of the fields are of type real while others are of type integer. For various system reasons it was decided that a single array will be used to store all the fields of an obstacle. Real arrays could not be used because the hardware automatically normalizes real variable values and consequently destroys information in the packed data structures. If integer arrays are used without appropriate scaling of values the loss in precision would be intolerable. All real entries are therefore scaled and the scale factor is the same for all. It is not easy to give precise estimates of the overheads involved in accessing packed data structures and performing scaling operations.

## 4.5.2 Operations

In the 2D case the operations on the obstacle data structure were decomposition, enlargement and dilation. Decomposition for the 3D problem was not implemented. A scheme similar to the 2D problem should be easy to design. The enlargement and dilation transformations are described next.

The _enlargement transformation_'s functions are similar to those of the 2D enlargement operator. Given a polyhedron OBS and a distance k, the transformation generates a new polyhedron NOBS. Every point on the boundary of NOBS is at least a distance k away from its nearest point on the boundary of OBS. A face is drawn parallel to each face of OBS at a distance k and outside of OBS. The interior of the new set of faces defines NOBS. The details of the corners are computed by finding intersection of the new faces. Once the coordinates of the corners are known, the edge representations are easily computed. The computations of the rest of the attributes of NOBS proceeds as before. As mentioned in section 4.4, the loss of maneuverable space is large whenever OBS has two faces meeting at a small angle or when there is a sharp corner.

The <u>dilation transformation</u> adds virtual corners to the description of polyhedra. As in the 2D case, the purpose of dilation is to simplify the computation of the point that is closest to the origin and lies on the section of a face or an edge. Virtual corners were defined in section 4.4. Virtual corner descriptions are saved as attributes of the edge or face that is responsible for it.

Determining the foot of the perpendicular from the origin to a line or plane is easy. Determining whether a point is inside an edge is also simple, especially, if parametric representation of lines is used. Determining whether a point P lies inside a convex polygon in cartesian space is more complex. Figure 4.6 describes how the membership of a point P in a convex polygon is determined. P1 is a corner and CG is the center of gravity of the polygon. $\underline{c}$ is the vector from P1 to CG, $\underline{b}$ is a vector from P1 to P and $\underline{a}$ is a unit vector along an edge from P1. We evaluate the dot product

$$(\underline{a} * \underline{b}) \cdot (\underline{a} * \underline{c}) \qquad - (1)$$

where "*" is a cross product and "." is a dot product of two vectors. In Figure 4.6(a), (1) evaluates to a negative number and in Figure 4.6(b), (1) evaluates to a positive number. If for any edge of a face, (1) is negative, the point under consideration is outside the face. Lagrange's identity [Brand (1957), page 34] gives

$$(\underline{a} * \underline{b}) \cdot (\underline{a} * \underline{c}) = \begin{vmatrix} \underline{a} \cdot \underline{a} & \underline{a} \cdot \underline{c} \\ \underline{b} \cdot \underline{a} & \underline{b} \cdot \underline{c} \end{vmatrix}.$$

Since the direction cosines of edges are normalized, $\underline{a}.\underline{a}=1$ and so the determinant is easily evaluated.

One final remark: Since any cross section of a convex polyhedron is a convex polygon, the faces of a convex polyhedron are convex polygons. Thus the above analysis for determining the membership of a point P in a face is adequate for our purposes.

## 4.5.3 Input Specifications

The input consists of the number of obstacles followed by obstacle specifications. Obstacle specifications include the number of faces, edges, corners and the maximum number of edges per face, data on corners, edges and faces, and the face-corner structure. The maximum number of edges (or corners) per face is used in allocating storage. Data for corners consist of the (x, y, z) coordinates of the point, for edges they consist of the names of the two end points, and for planes they consist of the number of corners on the face followed by a list of corner-edge names. The edge names are positive or negative depending on whether the edge is traversed in its natural direction or not when the boundary of the face is traced. The face-corner structure data give the names of three corners on each plane. The names of corners, edges and planes are positive integers starting at zero and incremented by one.

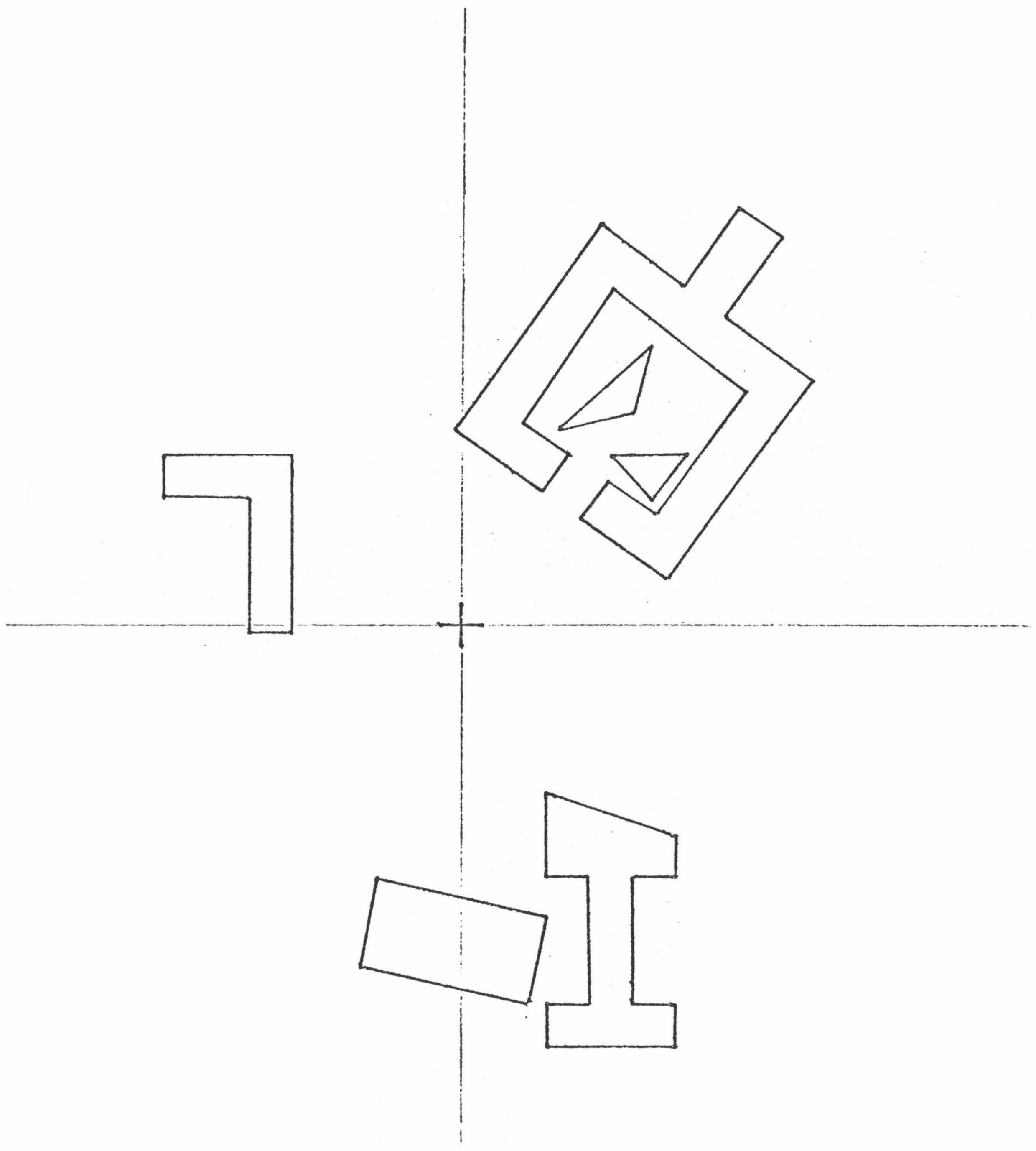


Figure 4. 1 A Primary Map

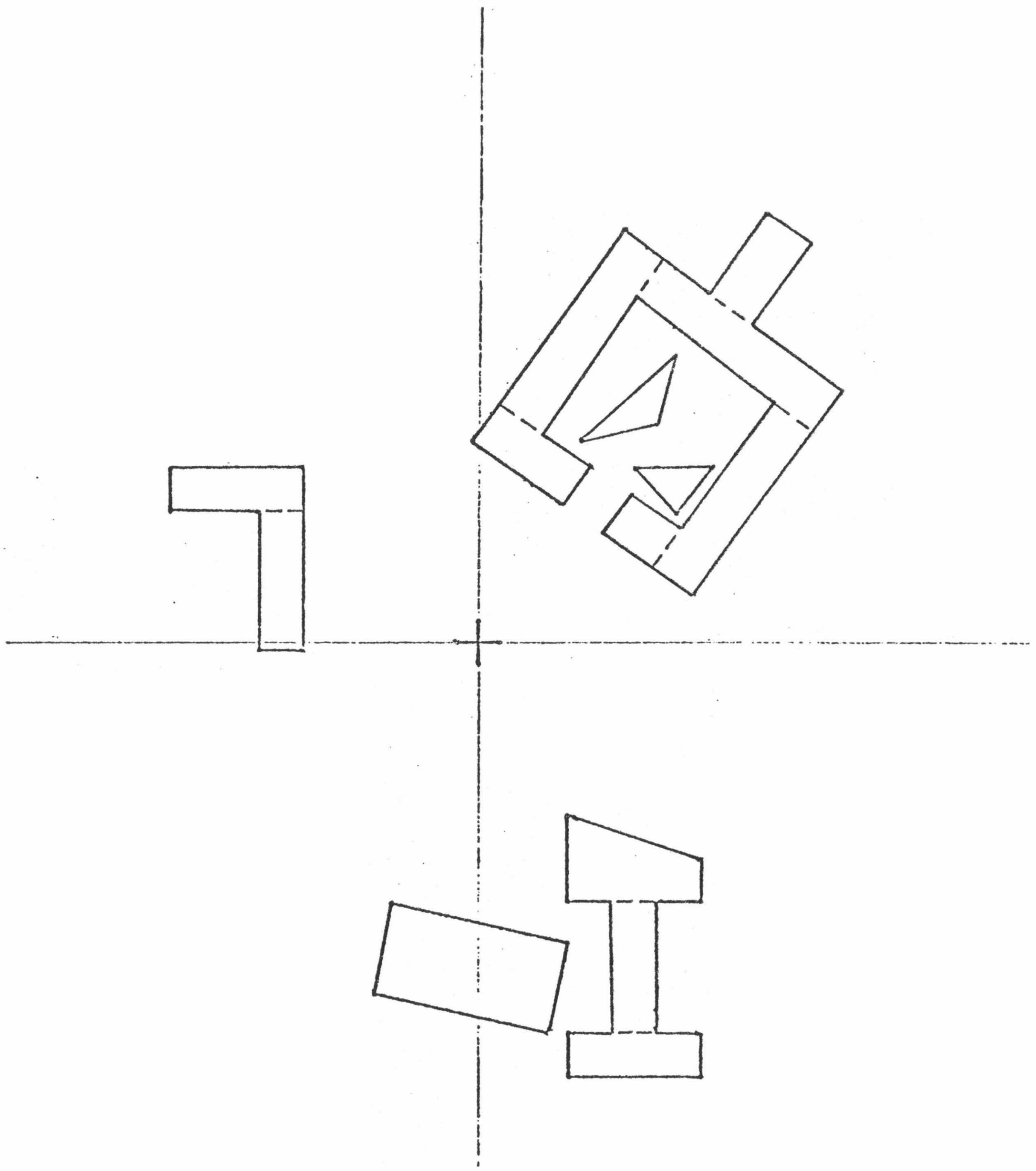


Figure 4.2 Decomposition of an Obstacle

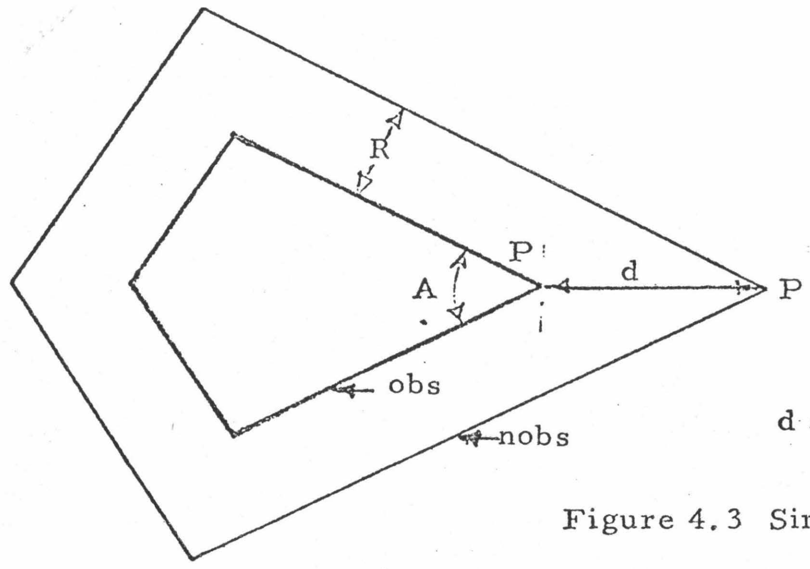


Figure 4.3 Simple Enlargement

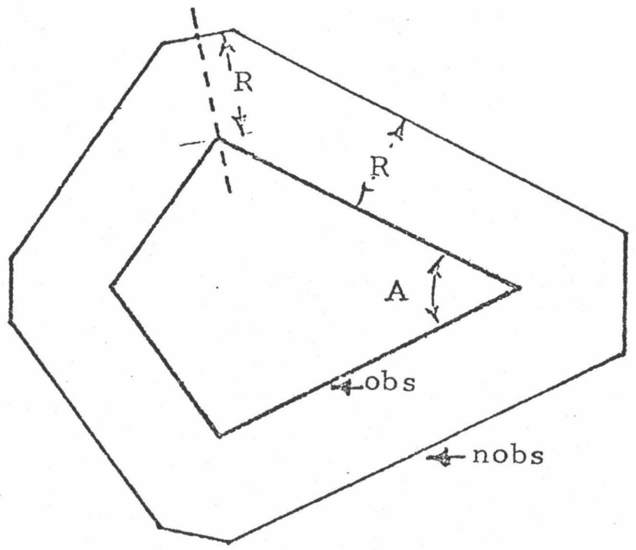
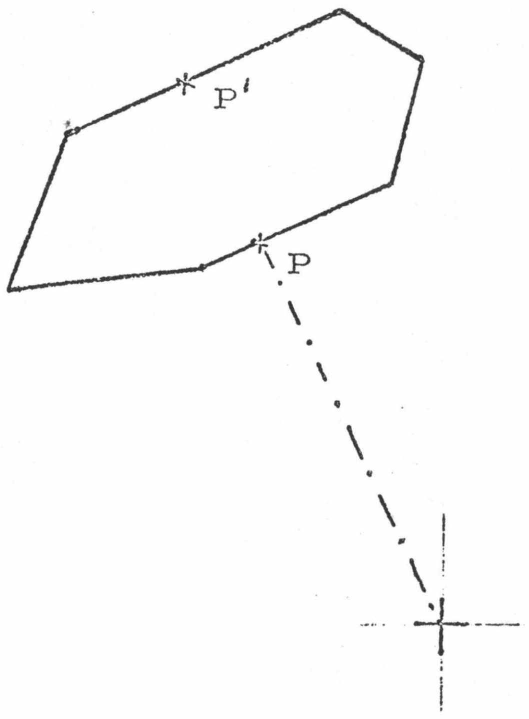
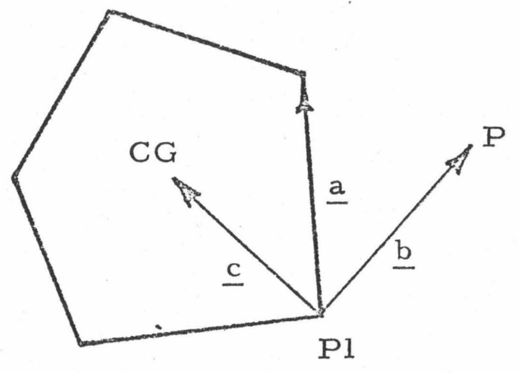


Figure 4.4 Enlargement Transformation

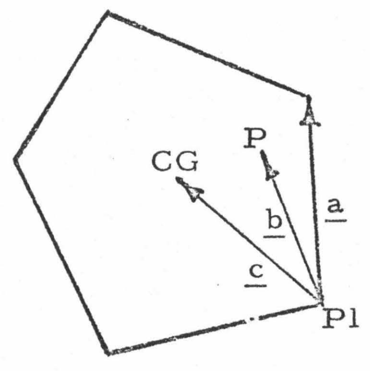


P, P' -- Virtual Corners

Figure 4.5 Dilation



(a) ($\underline{a} * \underline{b}$) · ($\underline{a} * \underline{c}$) negative



(b) ($\underline{a} * \underline{b}$) · ($\underline{a} * \underline{c}$) positive

Figure 4.6 Membership Determination

# CHAPTER 5

# FREE SPACE AND NAVSPACE MODELS

Obstacles are conveniently described in cartesian space and manipulator trajectories are best represented in joint variable space (a six-dimensional space). The complexity of the collision detection and avoidance problem is partly due to having these two diverse representations. If obstacles and trajectories could both be represented in one space, the <u>overhead of conversion</u> between the two spaces would be eliminated.

Additional complexity in the collision detection and avoidance problem arises because of the physical dimensions of the manipulator links. The enlargement transformation (Chapter 4) applied to obstacles permitted a very simple description of the manipulator as two straight line segments. Further simplification becomes possible if the manipulator is modelled in more abstract spaces where it can be considered as just a single line segment or as a single point instead of two straight line segments.

This chapter presents free space models that would avoid the conversion overhead problem, and transformations on these models that would admit simplified manipulator descriptions. <u>Free space</u> is the complement of the volume occupied by primary obstacles with respect to the manipulator's workspace. The dynamic chart model for free space and the algorithms for generating and refining them are presented here. The model simplifies the boom trajectory planning enormously. Using the model the motion of the boom in the maneuverable space of the manipulator can be reduced to the motion of a point in a chart. A simple extension of the model gives the notion of secondary charts, which is a formal characterization of the intuitive concept of free space relatively far away from obstacles. Within secondary charts the forearm can be ignored. This makes forearm planning trivial. Outside of the secondary charts, however, forearm planning has to be tackled in its full complexity.

This chapter begins with a discussion of the motivation for free space models and then describes three models. Each succeeding model is a refinement of the previous one. The first two models are deficient in many respects but their usefulness lies in clarifying important issues and leading to the dynamic chart model. All three models have been implemented for the simplified 2D manipulator and their characteristics are described here for the 2D problem. The dynamic chart model and some of its general properties applicable to both the 2D and 3D problems are described in section 4. Section 5 is on chart taxonomy. Section 6 presents the implementation of the 2D dynamic chart model.

Section 7 describes the obvious extension of the 2D solution to three dimensions and its problems. A critical evaluation of the 2D system identifies features whose complexity does not justify their utility. This makes it possible to find a more efficient 3D solution. Section 8 presents the implementation of the cleaner 3D dynamic chart model.

## 5.1 JOINT SPACE AND CARTESIAN SPACE

Obstacles in 3D are described in a cartesian coordinate frame and three values are required to describe a point. Such a description will be termed a _cartesian space_ representation. The 3D manipulator has six joints and a description of its configuration can be provided either as a six component vector specifying the various joint angles or as a position and orientation of the forearm. The former description is called a _joint variable space_ representation of the manipulator. A trajectory is a configuration as a function of time. The obstacles are naturally described in cartesian space and trajectories in joint variable space. If obstacles and trajectories could both be represented in one space, the overhead of conversion between the two spaces would be eliminated and the problem of collision detection and avoidance made more tractable.

The infrequent initialization hypothesis says that in any environment a large number of trajectories will be planned. Therefore representing obstacles in joint variable space would be better than representing trajectories in cartesian space. The transformation needed to represent obstacles in joint space would hopefully need to be only once for every environment. Representing trajectories in cartesian space on the other hand will require, for collision detection, a conversion into cartesian space every time a trajectory is planned.

A point in cartesian space needs only three values to identify it uniquely. Joint variable space is a six-dimensional space. There is therefore a redundancy in the specification of a cartesian space point making it impossible to find an isomorphism between cartesian space and joint variable space. Physically what this means is that the hand can be at any point in cartesian space in an infinite set of orientations. However, an isomorphism exists between cartesian space and <u>boom-space</u>, a subspace of the joint variable space generated by the three boom joint variables. For the general 3D manipulator the mapping between cartesian space and boom space is isomorphic only if the left- and right-handed configurations of the manipulator are accounted for explicitly. The existence of isomorphism simplifies boom planning but does not do much for forearm planning. The reason for this is that the world the forearm sees depends on the location of the boom tip and the boom tip can be located at an infinite set of points. Consequently there is an infinite set of mappings of the obstacles and free space into the forearm space. No finite characterization of this infinite set of mappings is possible and the best one can do is to compute a member of this set as and when the need arises and discard it later.

## 5.2 THE FIXED GRID MODEL

There exists a 1-1 and onto mapping between cartesian space and boom space. Free space comes in odd shapes depending on the shapes of the obstacles. The fixed grid model is a simple minded approximation scheme. It approximates the jagged free space available for maneuvering the boom by sectors in r-theta space. Figure 5.1 shows an example of the free space approximation. The polygons in the figure represent obstacles. Boom planning is done using single-joint at a time strategy.

There are two important problems with this model. The first is the loss of valuable maneuverable space due to the approximation. The arrow in Figure 5.1 points to a subset of the free space that is lost due to the approximation. If the loss is to be avoided the entire representation has to be computed to a very fine detail. This is expensive and is often unnecessary because the manipulator may never maneuver in certain areas. The second problem is that this representation can handle only those trajectories which involve single joint motions. If the boom tip traced any other type locus, boom collision checking would be expensive. Even with single joint motions of the boom, forearm collision checking is expensive. When the boom moves keeping its r-joint value constant the boom tip traces a circle. When the boom tip traces a circle the trajectory area generated by the motion of the forearm is complex, resulting in forearm collision checking being time consuming. Thus a planning strategy that is good for the boom will perform poorly for the forearm, and a strategy that works for the forearm will cause problems for the boom.

## 5.3 THE VARIABLE GRID MODEL

In order to find a representation of free space that avoids the loss of valuable maneuverable space that occurs in the fixed grid model, the variable grid model was introduced. An example of the free space approximation using this is shown in Figure 5.2. The polygons in the figure represent obstacles. The basic idea is to have the grid point placement be controlled by the obstacle descriptions. Each polygon is first approximated by a series of constant r and constant theta lines, called the r-theta envelope. Of these, the ones that are between the origin and the obstacle are used to set up the variable grid model. An attempt to find an r-theta envelope such that the extra space occupied by the r-theta envelope is less than some prespecified fraction of the obstacle area did not lead to anything interesting. So a few simple heuristics were used instead. This model suffers from the same two problems as the fixed grid model.

## 5.4 THE DYNAMIC CHART MODEL

The dynamic chart model solves the two problems that plagued the fixed and variable grid models. Examples of free space approximations using this are shown in Figures 5.3 - 5.5. The polygons in these figures are obstacles. The permissible boom trajectories are ones where the boom tip traces straight lines or one in which a linear relationship between the boom joint angles is maintained.

The single-joint-at-a-time trajectory problem is solved by deciding to model not the available free space but navspace, the set of boom tip locations for which the entire boom is safe. This is a crucial observation. For, in navspace, the boom can be considered to be a point and the movement of the boom no longer results in a complex trajectory envelope but in just a line. The boom trajectory planning no longer involves any intersection checks for areas or volumes but is concerned with joining two points by a line which is constrained to pass through well characterized regions. The areas enclosed by the dotted lines in Figures 5.3 - 5.5 are approximations to navspace.

The loss-of-maneuverable-space problem is solved by imposing a structure on navspace. Navspace is approximated by <u>charts</u> comprised of boxes in r-theta-phi space called <u>regions</u>. Figure 5.3 shows six regions each of angular width 60 degrees. The region boundaries are indicated by radial lines with thick arrows at their ends.

Navspace approximation is dynamic and can be changed by other high level programs. The approximating procedure is called <u>refinement</u> and the refinement level is called <u>resolution</u>. Resolution refers to the angular width of the region. The greater the resolution, or equivalently the smaller the angular width of a region, the better the region approximates the relevant part of navspace. Since there is a limit to the precision of placement of the hardware there is a limit to the maximum resolution handled by the system. Associated with each joint is the smallest angular change the hardware can resolve. The minimum of these ranges over the different joints is called the <u>minimum angular range</u> of the manipulator hardware. The minimum angular range determines the maximum resolution handled by the system.

The system can refine to a greater resolution areas where the manipulator needs to maneuver in, while elsewhere the resolution may be quite crude. This flexibility is very useful because refining every part of free space to the finest level possible is expensive and often quite unnecessary. This flexibility permits the system to decide where refinement is essential and what the resolution should be. If the resolution of a particular part of the environment is not adequate, the system can refine that portion of the free space. This is termed the <u>selective refinement</u> capability. Figure 5.4 is the same as Figure 5.3 except that the portion of the free space between OP1 and OP2 has been refined. Note that point S which was not within a region is now inside a region. Figure 5.5 is another example of refinement; it is the same as Figure 5.4 except that the portion of the free space between OP3 and OP4 has been refined. Point G is now brought within a region. This capability makes <u>incremental modifications</u> to the region data structures inexpensive. Incremental modifications are necessitated by minor changes in the environment that might result from the transporting of objects from one place to another.

Regions in 3D are made up of sectoroids and sectoroids are composed of pascs (see Figure 2.2). The pasc (parallelepiped in spherical coordinates) is the smallest unit. Pascs, sectoroids and regions are bounded by constant phi and theta surfaces. All pascs in a sectoroid have the same phi limits. All sectoroids in a region have the same theta limits. Pascs have associated with them a maximum and minimum r value, called rmax and rmin, signifying the safe limits of the boom extension. Situations occur where, for the given angular limits, there does not exist a safe boom position for any extension whatsoever. Such a situation is indicated by equal rmax and rmin values. The difference between rmax and rmin is called the safe limit interval. Similar to pascs, sectoroids and regions have associated with them maximum and minimum r values, called rmax and rmin, signifying the best possible safe limits of the boom extension. A region, sectoroid or pasc is considered impassable if the safe limit interval is less than some prespecified value.

In 2D regions are composed of sectoroids only. Similar to the 3D case, sectoroids and regions in 2D are again bounded by constant theta lines and have maximum and minimum r values associated with them. The definitions of safe limit interval and impassable are valid for 2D too.

It is worth reiterating the distinction between the concept of navspace and its approximation by charts, and the choice of a specific structure for the constituents of the chart. Navspace and its approximation by charts is very crucial to collision checking since it permits the boom to be considered as a single point. The reason for imposing a structure on charts is to have some selectivity in terms of what parts of navspace would be refined and to what level. The exact nature of the region and its components is irrelevant to collision checking. The choice of a box in r-theta-phi (for 3D) and a box in r-phi (for 2D) is dictated by the choice of a particular planning strategy described in Chapter 8.

## 5.5 CHART TAXONOMY

The motivation and some introduction to the notion of secondary obstacles and secondary maps have already been presented in Chapter 4 and the introduction to this chapter. The chart approximating navspace of the primary map is called the <u>primary chart</u>, and the chart for the secondary map is called the <u>secondary chart</u>. The pascs, sectoroids and regions are appropriately prefixed with primary and secondary according as they are members of the primary chart or secondary chart.

Again, the idea of secondary charts is a precise characterization of large chunks of empty space far away from obstacles. Within secondary regions, for collision detection and avoidance, the manipulator can be considered to be a single point. This simplifies trajectory planning significantly.

In order to be able to ignore the lateral property of the manipulator, left and right maps are generated. Corresponding to these two types of maps we have four charts, the <u>left primary chart</u>, the <u>right primary chart</u>, the <u>left secondary chart</u> and the <u>right secondary chart</u>.

Conceptually the primary and secondary charts are similar. Storage considerations necessitate the merging of the two into one. There is thus a left chart and a right chart and each has attributes of the associated primary and secondary chart. Since the organizations of the left chart and the right chart are the same only one of them, called the chart will be described in the following two sections. The chart components will be regions and a region will consists of sectoroids. In 3D sectoroids will be made of pascs. Regions, sectoroids and pascs have various attributes describing them. Of these the only ones that are not common to both primary and secondary charts are the r values indicating the maximum and minimum safe boom extensions. The r values for the primary chart and secondary chart components are computed using the primary map and the secondary map respectively.

## 5.6 CHARTS IN 2D

This section describes the data structures for charts, regions and sectoroids in 2D and the operators that can act on them. Figures 5.3 - 5.5 show examples of charts in 2D.

### 5.6.1 Data Structures

Charts are represented as linked lists of regions, permitting easy addition and removal of regions. Region description includes a fixed set of attributes and a body. The body is represented by a linked list. A linked list representation is essential because the refine-chart operation chops up regions resulting in a rearrangement of the body, and this rearrangement is easily carried out by pointer adjustments.

The fixed attributes of a region include the resolution, size, theta limits and their tangents, maximum and minimum r values for the primary and secondary charts between the theta limits of the region, and pointers to the beginning and end of the body. The resolution is an integer and indicates that the angular width of each sectoroid is resolution times the minimum angular range (see definition in section 5.4). The size gives the number of sectoroids per region. The theta limits specify the maximum and minimum theta values of the region. The theta interval is open at the lower end and closed at the upper end. Since the theta values have a period of 360 degrees, the circular ordering described in Appendix 2 is used to order the theta values. To avoid having to compute the tangents of the theta limits repeatedly, they are also saved. The r values indicate the maximum and minimum safe boom extension values. Finally the set of fixed attributes includes pointers to the start and end of the sectoroid list forming the body.

The body of a region is a linked list of sectoroids. The attributes of a sectoroid include its theta limits and their tangent values, the maximum and minimum r values for the primary and secondary charts between the theta limits of the sectoroid and a pointer to the next sectoroid. The theta limits are similar to the theta limits of the parent region. The angular width of the sectoroid is determined implicitly by the resolution of the region. The maximum and minimum r values for a sectoroid, again, are similar to the corresponding region attributes. The rmin and rmax of a region are respectively the minimum of rmin and maximum of rmax of the component sectoroids. The final attribute is a pointer to the next sectoroid, the sectoroids being arranged in increasing values of their theta-min value.

### 5.6.2 Operations

The permissible operations on charts are generate-chart and refine-chart. These are implemented using the region operation generate-region. The region operation is described first followed by the chart operations.

The generate-region operator is given the theta limits, the desired resolution and the primary and secondary maps, and it generates a region along with all its attributes. The only attributes that are complex to compute are rmax and rmin. We will be interested in computing the rmax and rmin for a sectoroid. The rmax and rmin for a region will then be easy to compute. In fact, we only need to know how to compute rmax for a sectoroid. To compute rmin, we first compute rmax for the sectoroid obtained by rotating its theta limits by 180 degrees; rmin is then the difference between the length of the boom and the rmax for the rotated interval. A function called maximum safe boom extension (MSBE) computes the rmax within any sectoroid.

MSBE first determines the subset of the map that intersects the given theta interval. The maximum safe extension, max-ext, is set to the maximum permissible boom extension. The obstacles in the map are sorted by their rmin and so are the obstacles in the chosen subset. The minimum of the maximum safe boom extension permitted by each of the obstacles is the desired rmax. The analysis of each obstacle proceeds as follows. If the rmin of the obstacle is less than max-ext, the obstacle is analyzed in detail. The detailed analysis includes the following for each edge of the polygon. The section of the edge lying within the specified theta limits is computed. The knowledge of the slopes of the region's theta limits makes this very easy. The introduction of virtual corners by the dilation transformation ensures that the extrema of the distance, from the origin, of points on the line are at the two end points of the line. Max-ext is set to the minimum of its current value and the minimum of the r value of the two end points of the line.

The <u>generate-chart</u> operator generates six regions each 60 degrees wide with a default resolution of 180. This resolution is equivalent to an angular width of 60 degrees for the sectoroid theta interval.

The <u>refine-chart</u> operator is given the theta limits, the desired resolution and the primary and secondary maps. The operator refines the given interval to a degree such that the resolution of every region in it is greater than or equal to the desired resolution. Figures 5.3 - 5.6 are examples showing selective refinement. All regions within the given interval with resolution better than the specified value are left untouched. A region that intersects the given interval has the common part chopped off from it. A new region having the desired resolution is developed over the common interval using the generate-region operation.

## 5.7 EXTENDING 2D IDEAS

In 2D a region is a box in r-theta space. There are no restrictions over the length of the theta interval. Rmax and rmin denote the maximum and minimum safe boom extensions over the given theta interval. This definition has a natural extension to the 3D problem. A region in 3D is specified as an arbitrary rectangle in theta-phi that has associated with it an rmax and rmin. Since in 2D regions can be of any angular width the corresponding theta-phi rectangle for the 3D region can have any theta width and any phi width.

The operators that are needed are, as before, generate-region, generate-chart and refine-chart. The next few paragraphs show how the last of these operators runs into considerable difficulty when a direct extension of the corresponding 2D operator is attempted.

In two dimensions, refine-chart first computes all regions that intersect the given interval. Those regions of this set that have a resolution better than the specified value are left alone while the others get chopped up into two sections, one lying within and the other outside of the given interval. New regions with the desired resolution are developed over the inside sections.

A similar approach at the 3D level leads to two problems. Figure 5.6 shows an example of region placement in the theta-phi plane. The dotted rectangle is the area that is to be refined. The first of the problems is that having the theta-phi space occupied by a random assortment of regions, it is difficult to find the set of relevant regions intersecting any given rectangle in theta-phi space. The other and the more important problem is that the chopped up sections of regions will no longer be rectangles in theta-phi. Figure 5.6 shows a region on the boundary that is chopped up and whose shaded part is no longer rectangular. Since by definition regions have to be rectangles in the theta-phi space, additional computations are necessary before the non-rectangular section can be considered as a region.

The solution to the first problem is to insist that regions be not arbitrary rectangles, but uniform squares covering the theta-phi plane. Figure 5.7 shows a uniform grid of regions in the theta-phi plane. This is no restriction at all, since in terms of approximating navspace, it is equivalent to the seemingly more general scheme.

There are two possible solutions to the second problem. The first is to decompose the concave remnant of the region into convex subcomponents and the second is to refine the entire problem region (without chopping it up) to the desired resolution. The first is no good because it is computationally expensive and is plagued with horrible book-keeping chores. So we adopt the second solution.

To summarize, we require that

1) all regions be squares of the same size in the theta-phi space, and

2) an entire region will get refined if it intersects the given area and has a resolution less than the desired value.

These requirements simplify the algorithms for refine-chart. Generate-chart is similar to its 2D counterpart. The complexity of generate-region, however, is considerably more in 3D than in 2D, because of the extra dimension involved.

With this introduction I will describe the data structures for charts, regions, sectoroids and pascs and the three permissible operations on them.

## 5.8 CHARTS IN 3D

### 5.8.1 Data Structures

A chart is a two-dimensional array of regions. The position of a region in the array indicates its position in the theta-phi plane. Regions are squares in the theta-phi plane (30 degrees wide). In all there are seventy-two regions. There could have been 18 or 288 or some other integer number of regions. If there are too many regions, the storage requirements will increase. If there are too few regions, each of them will occupy a large theta-phi square. The reason for imposing structure on navspace through the introduction of regions etc. was to provide some selectivity in terms of what parts of navspace would be refined and to what level. Having very few regions would destroy this goal. We wanted a number that is neither too small nor too large. This led to the choice of seventy-two as the number of regions. Region description includes a fixed set of attributes and an array of sectoroids. The sectoroid description also includes a fixed set of attributes and an array of pascs.

The fixed attributes of a region include the theta and phi limits, the resolution, the grid-size, the region-size and the rmax and rmin for the primary and secondary charts over the given theta-phi square. The theta and phi limits specify the maximum and minimum theta and phi values, respectively, of the region. The two intervals are equal and are both open at the lower end and closed at the upper end. The circular ordering defined in Appendix 2 is used for ordering the phi values. The resolution is an integer and indicates that the angular width of each sectoroid and pasc is the resolution times the minimum angular range (see definition in section 5.4). The phi angular width of a sectoroid is the same as the theta angular width of a component pasc and they are both equal to the grid-size. There are as many pascs in a sectoroid as there are sectoroids in a region and region-size indicates this number. Rmax and rmin indicate the maximum and minimum safe boom extension values.

The sectoroid attributes include two things. The first is the phi-maximum of the phi interval over which the sectoroid is defined. The second consists of rmax and rmin for the primary and secondary charts over the sectoroid theta-phi area. The theta-phi area covered by the sectoroid, is the full theta interval of the region but the phi interval is only of size equal to the region grid-size and extends up to the phi-maximum of the sectoroid.

The pasc attributes are similar to the sectoroid. The first is the theta-maximum of the theta interval over which the pasc is defined. The second consists of rmax and rmin for the primary and secondary charts over the pasc theta-phi area. The pasc theta-phi area is of width equal to the region grid-size and extends up to the theta-maximum of the pasc and the phi-maximum of the parent sectoroid.

### 5.8.2 Operations

The allowable operations on charts are generate-chart and refine-chart. These are implemented in terms of the region operator generate-region.

The generate-chart operator generates a 12 * 6 array of regions each with a default resolution of 90. This resolution is equivalent to an angular width of 30 degrees which is the angular size of the region. Generate-chart uses the operator generate-region described below.

The refine-chart operator is given the theta and phi limits of a rectangular area, the desired resolution and the primary and secondary maps. The operator ensures that all regions of the primary and secondary charts that intersect the given area will have a resolution greater than or equal to the desired value. It does so by first computing a list of all regions which intersect the given area. Those regions of this set that have a resolution better than the desired value are left untouched. The others are replaced by newly generated regions that have the desired resolution. The new regions are obtained by using the operator generate-region described next.

The generate-region operator is given the theta and phi limits of the region, the desired resolution, the primary and secondary maps and whether the region generated is for the left or right chart. The operator generates a region along with all the attributes of the region.

The only attributes that are complex to compute are the rmax and rmin fields. We will be interested, primarily, in computing the rmax and rmin for a pasc. The rmax for a sectoroid is then the maximum of the rmax of the sectoroid pascs and the rmin for a sectoroid is the minimum of the rmin of the sectoroid pascs. Similarly the rmax and rmin for a region are computed by knowing the corresponding values for the region's component sectoroids. In fact, we only need to know how to compute rmax for a pasc. To compute rmin, we first compute rmax for the pasc obtained by rotating its theta and phi limits by 180 degrees and changing the manipulator configuration from left to right or right to left; the difference between the length of the boom and the rmax for the rotated interval is the value of rmin. A function called maximum safe boom extension (MSBE) computes the rmax within any pasc.

MSBE computes rmax by finding, for all the planar faces of obstacles that go through the theta-phi box of the pasc, the point on them and inside the theta-phi box that is closest to the origin. Constant-theta surfaces are cones and consequently their intersection with planes gives second degree curves. Finding the closest point to the origin on a planar figure bounded by a second degree curve is computationally expensive. A conservative simplification is to replace the theta-phi boxes by a minimum bounding viewbox that is a pyramid whose axis extends to infinity. Two of the faces of the pyramid are the constant-phi surfaces. The remaining two faces enclose the constant-theta surfaces. Figure 5.8(a) shows a cross sectional view of the theta-phi box and the minimum bounding viewbox. Figure 5.8(b) shows the projection of the minimum bounding viewbox on the X-Y plane. Note that one face of the viewbox is tangential to the outer cone while the other is strictly inside the inner cone. The apex of the viewbox is at the same point where the theta-phi box had its apex.

MSBE first computes the subset of the map that has obstacles that have a non-trivial intersection with the viewbox. The maximum safe extension of the boom, max-ext is set to the maximum permissible boom extension. The obstacles in the map are sorted by their rmin and so are the obstacles in the chosen subset. The minimum of the maximum safe boom extensions permitted in the viewbox by each of the obstacles is the desired rmax.

The analysis of each obstacle proceeds as follows. If the rmin of the obstacle is less than max-ext, the obstacle is analyzed in detail. Analysis of an obstacle means the analysis of the corners, edges and faces of the obstacle. Since the faces of an obstacle are sorted by rmin of the faces, the number of faces that need to be considered is reduced.

Corner Analysis : For each corner of the obstacle, if the distance of the corner from the origin is less than max-ext and the corner is inside the viewbox then max-ext is set to the distance of the corner.

Edge Analysis : The following computations are done on each edge of the obstacle. If there is a virtual corner on the edge then max-ext is updated with the distance of the foot and the analysis of the edge is over. <u>Updating</u> max-ext with a distance is assigning max-ext the minimum of its current value and the distance. Otherwise, the section of the edge lying within the viewbox is computed using a generalization of the Sutherland and Cohen clipping algorithm [Newman and Sproull (1973)] and max-ext is updated with the distances of the end points of this section.

Face Analysis : The following computations are carried out on each face of the obstacle. If the rmin of the face is less than max-ext then the face is analyzed in detail. If there is a virtual corner on the face and it lies inside the viewbox then max-ext is updated with the distance of the virtual corner and the analysis of the face is over. If the face is outside the viewbox then also the analysis of the face is over. Otherwise, a complete face analysis is called for. This involves the analysis of four pseudo-edges generated by the intersection of the four faces of the viewbox and the given obstacle face. What this means is that, with respect to the viewbox, the effective part of the obstacle face may be much less than the entire face. Since the closest point to the origin lies somewhere along the boundary of the effective part, the possible pseudo-edges generated by the intersection of the viewbox faces with the obstacle face have to be considered. The pseudo-edge analysis computes the following for each face of the viewbox. If the obstacle face does not intersect the infinite viewbox face then that viewbox face is ignored. Otherwise, the standard edge analysis is carried out on the finite edge generated by the intersection of the finite obstacle face and the infinite viewbox face.

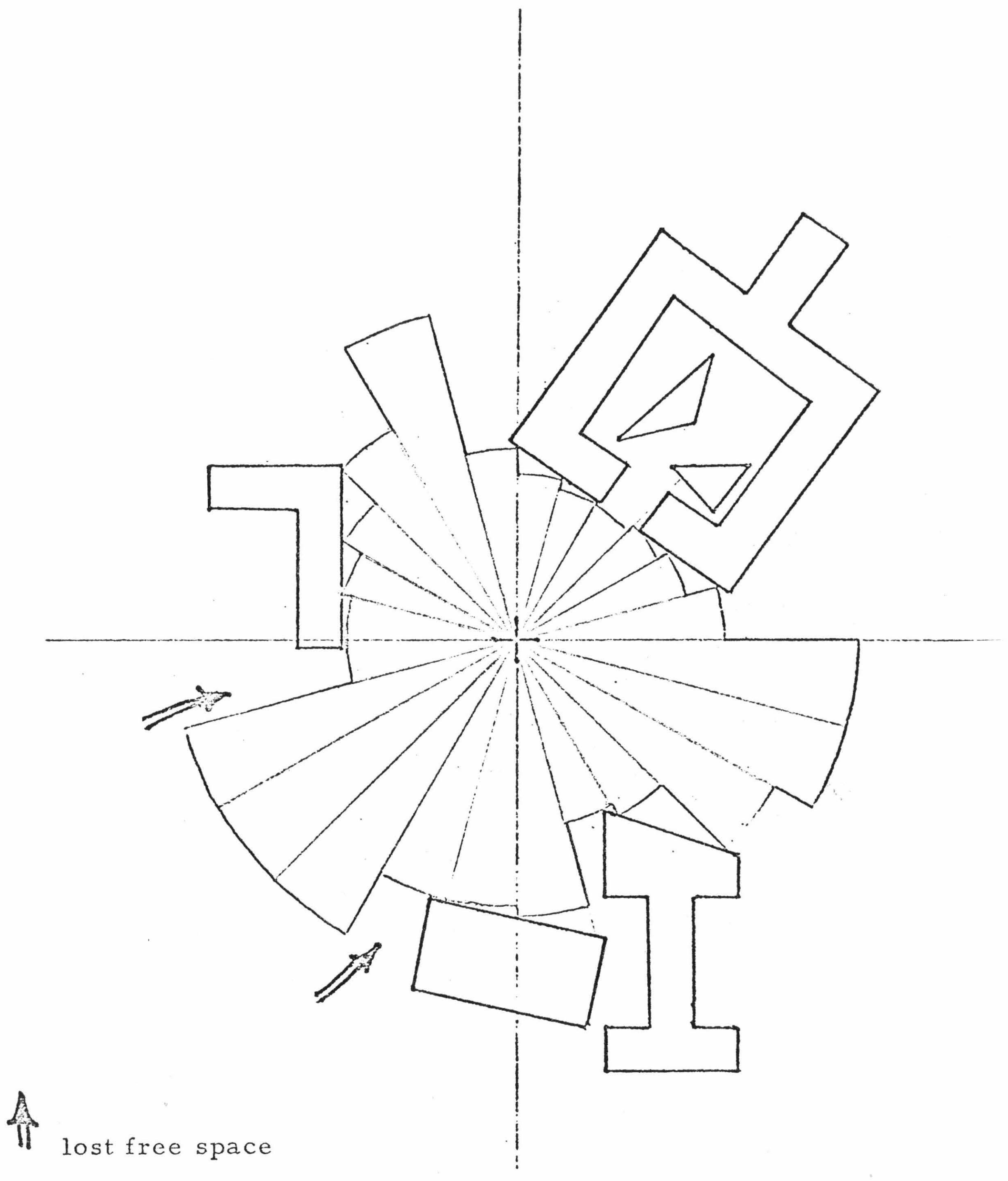


Figure 5.1 The Fixed Grid Model

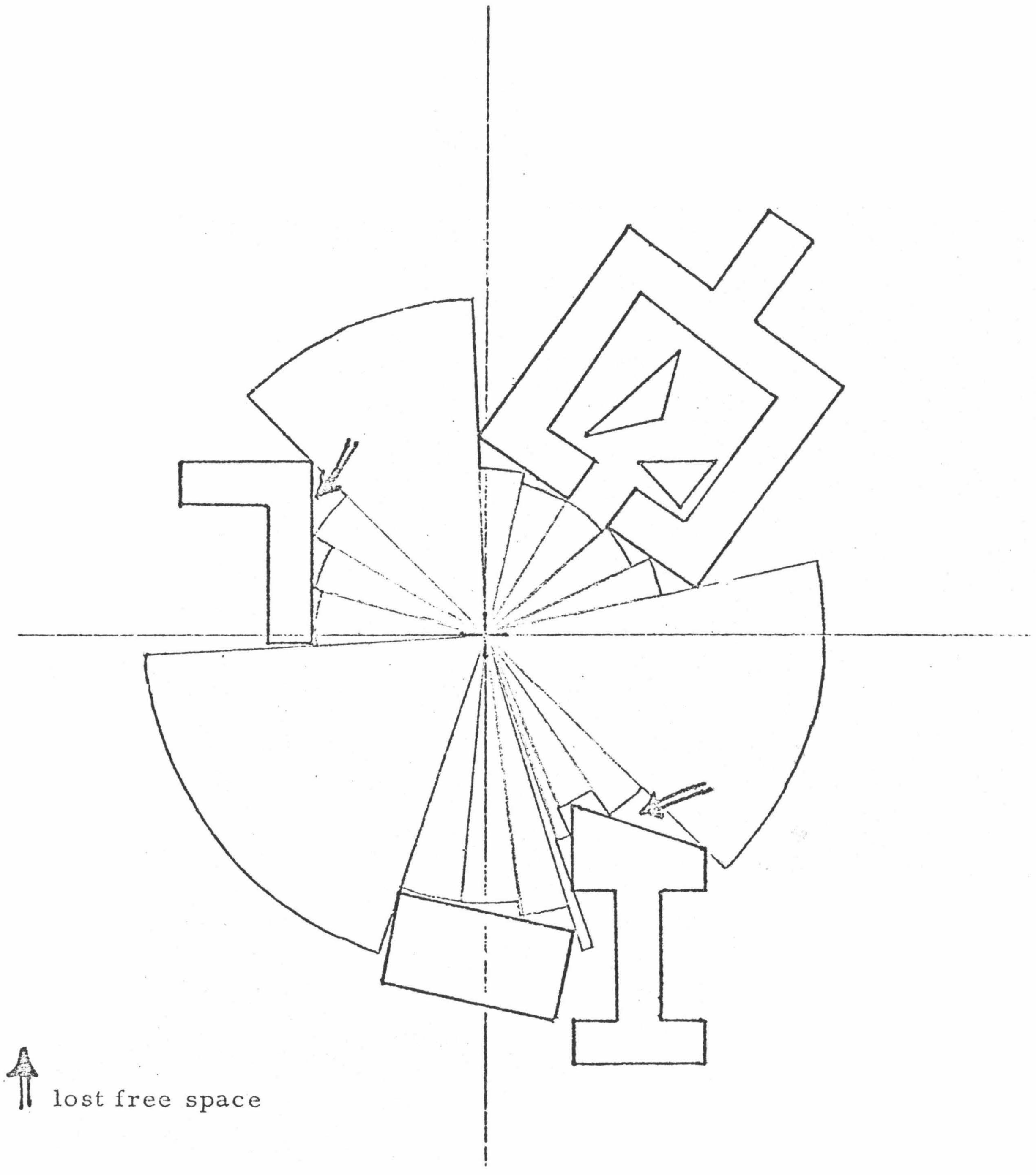


Figure 5.2 The Variable Grid Model

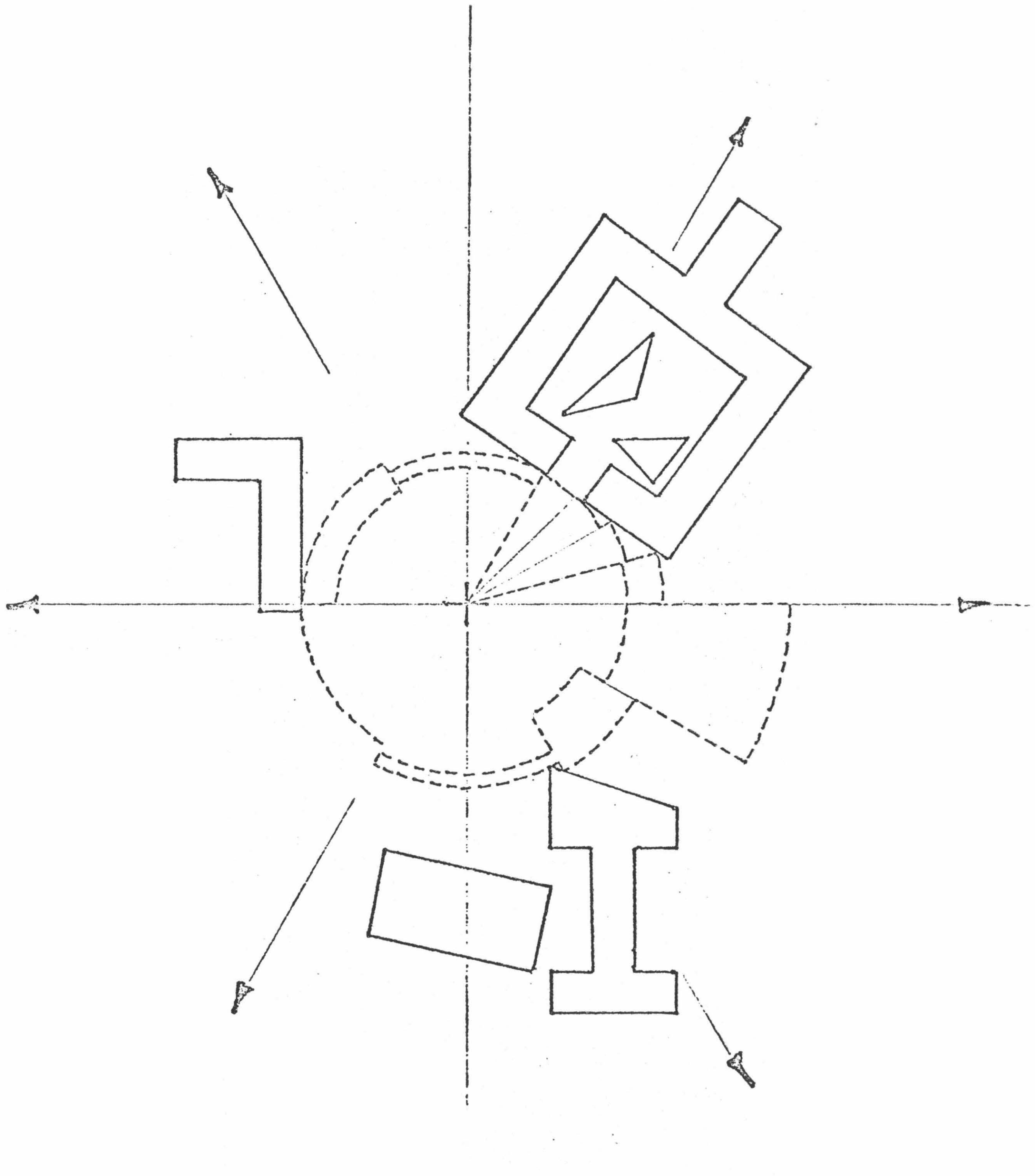


Figure 5.3 The Dynamic Chart Model

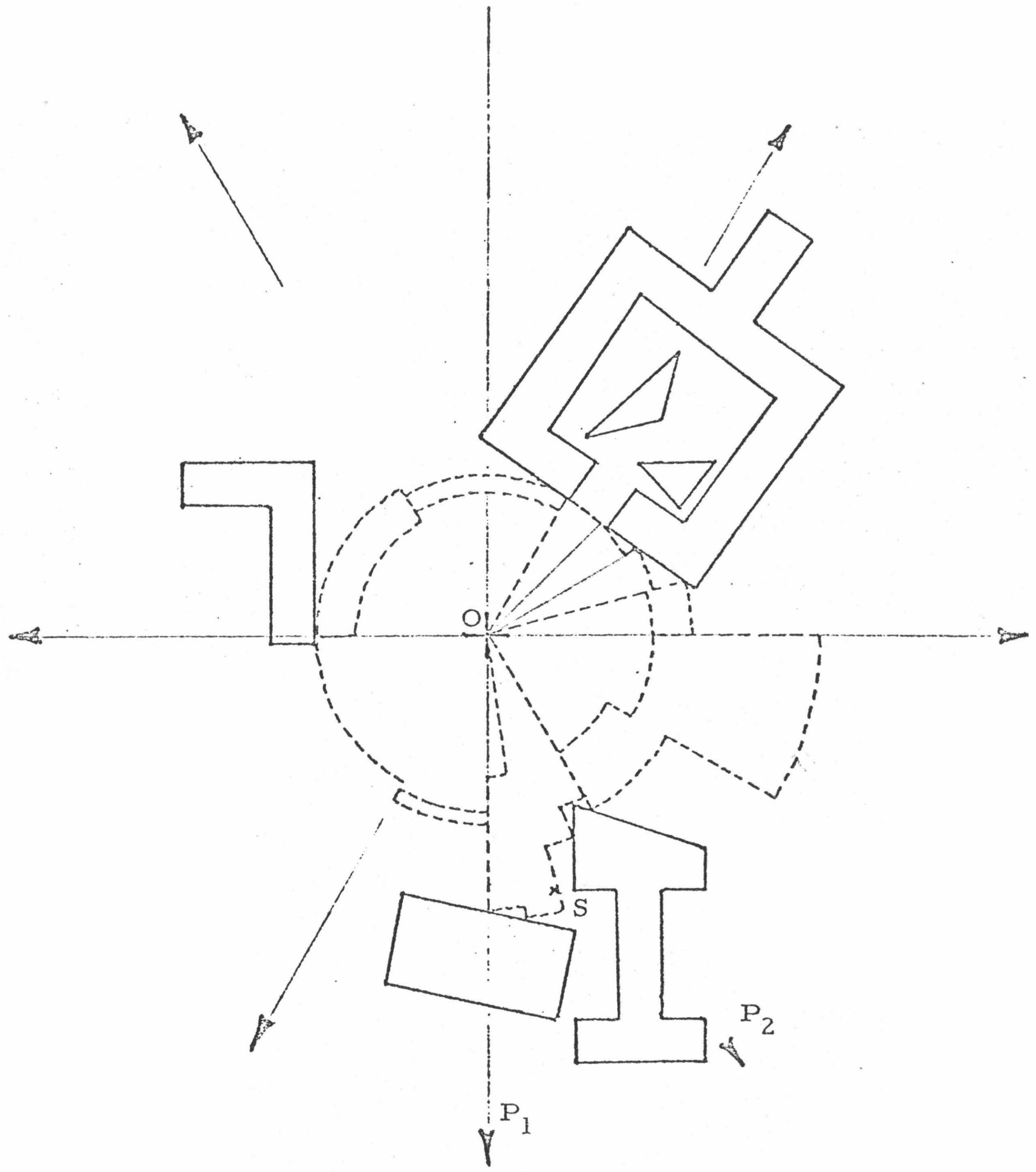


Figure 5.4 Chart Refinement - 1

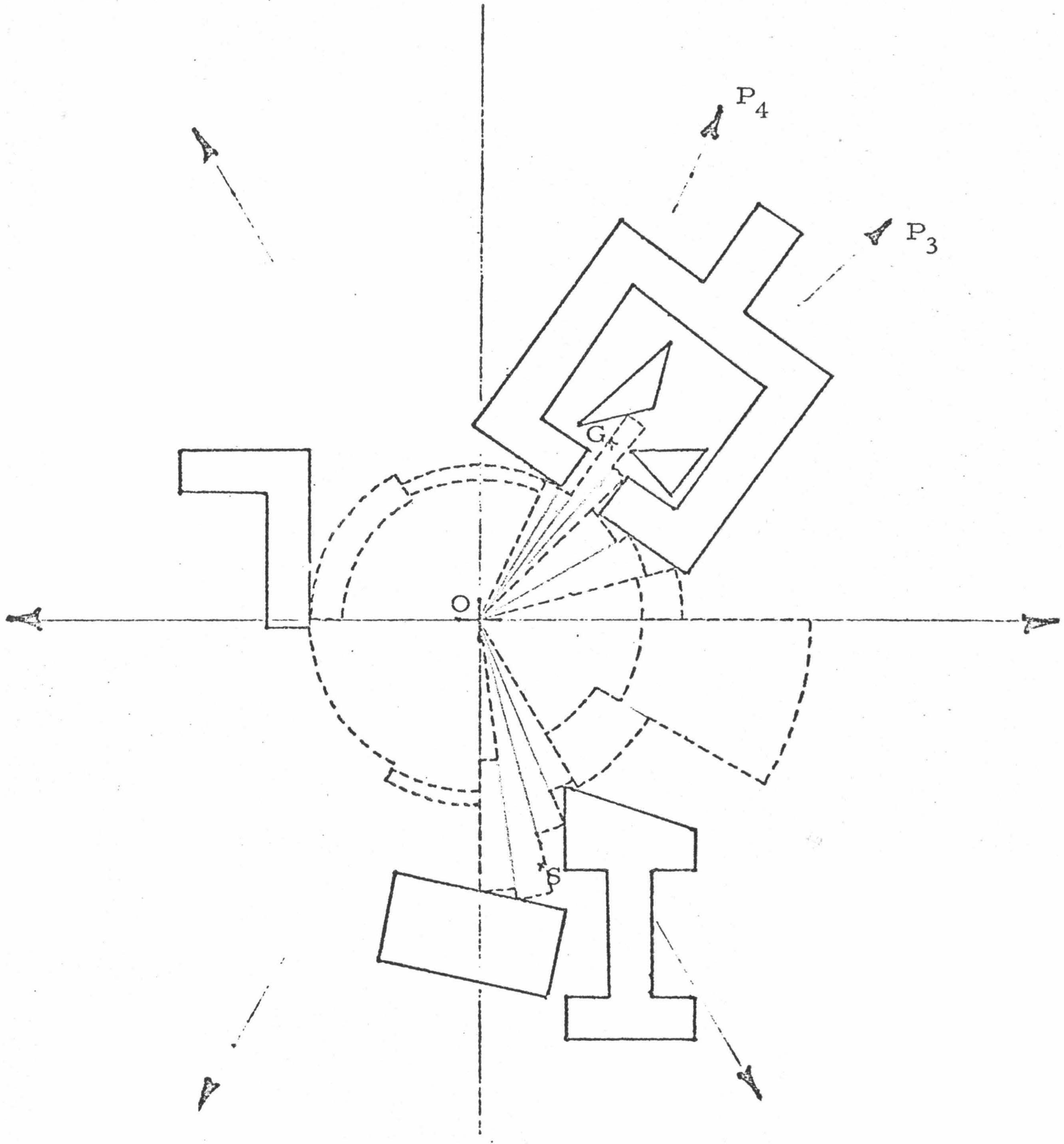


Figure 5.5 . Chart Refinement - 2

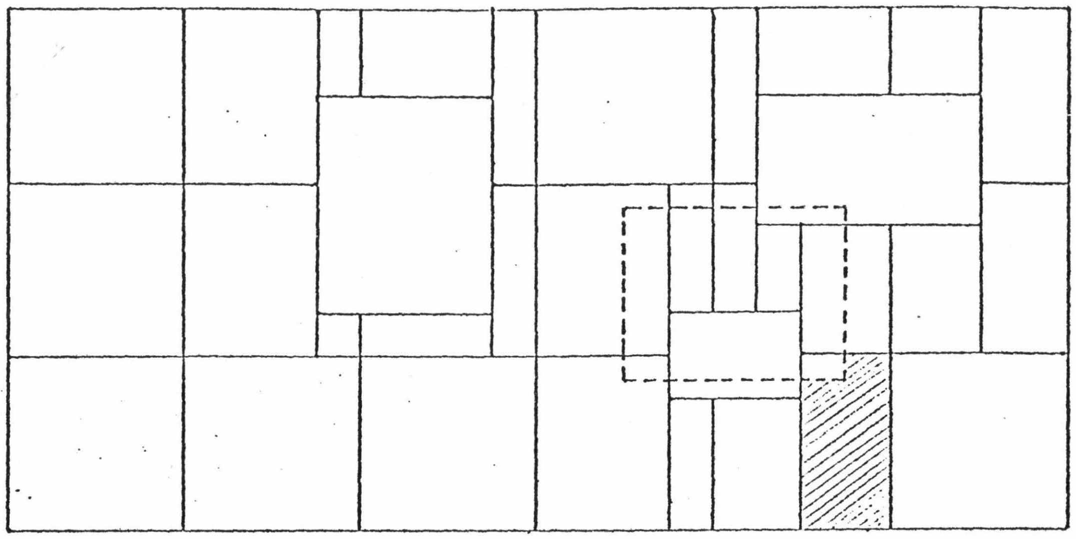


Figure 5.6 Random Placement of Regions

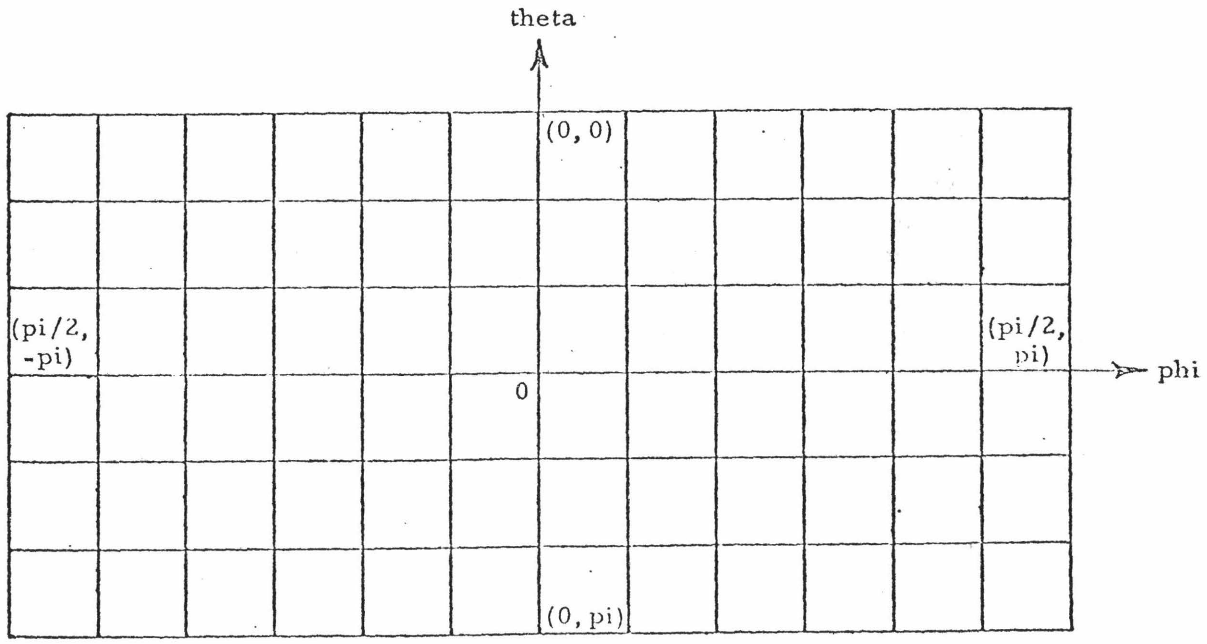
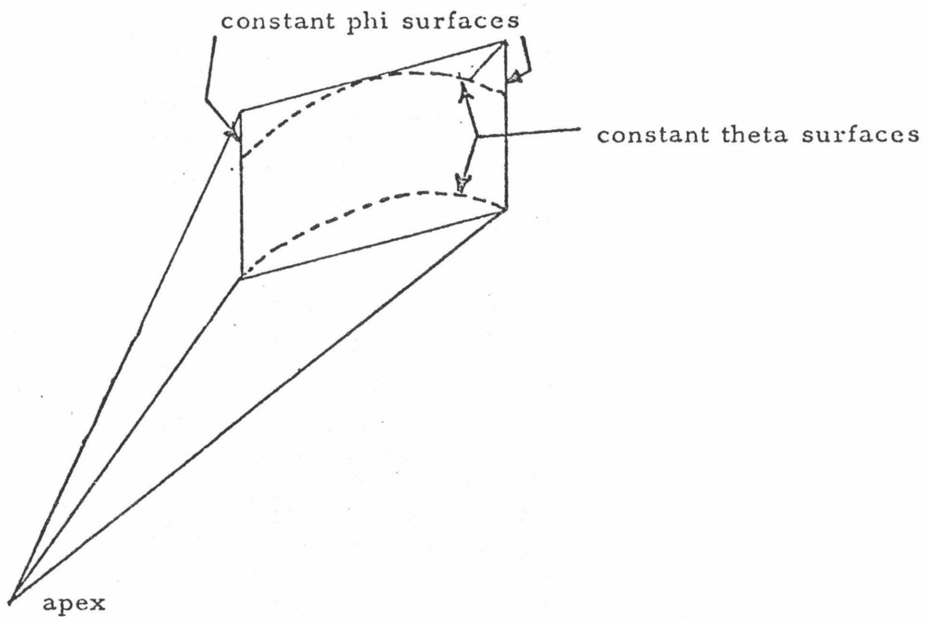
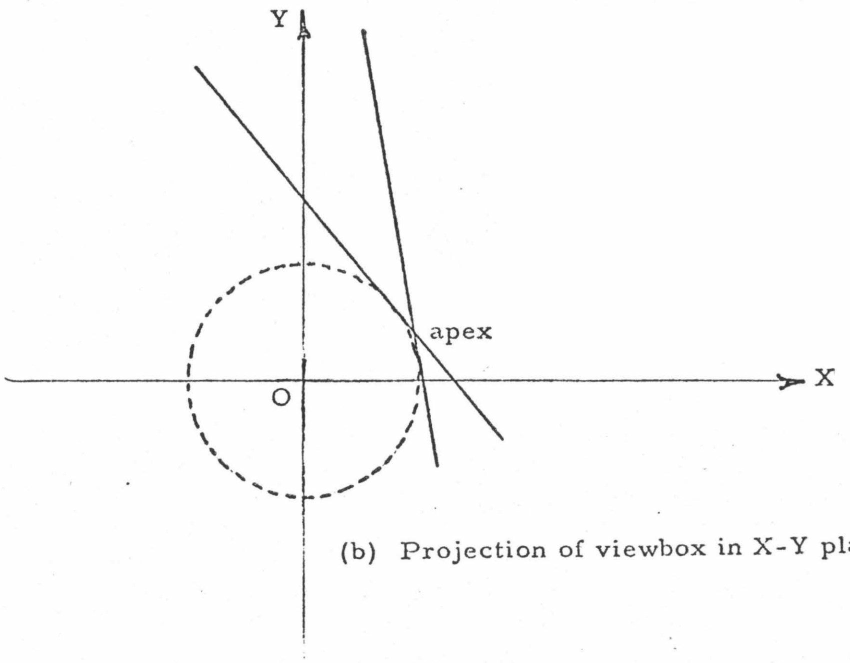


Figure 5.7 Uniform Grid of Regions



(a) Viewbox and cross section of theta-phi box



(b) Projection of viewbox in X-Y plane

Figure 5.8 Minimum Bounding Viewbox

# CHAPTER 6
# TRAJECTORY MODELLING AND CALCULATION

When the manipulator's links move they trace a volume in 3-space called the <u>trajectory envelope</u>. The representation hierarchy of section 2.2.1 showed how it is possible to reduce the complexity of the trajectory envelope from a two-element solid to a single surface or even a single line. The single surface is called the <u>trajectory surface</u> and the line the <u>trajectory trace</u>. Collision detection requires determining the intersection of the trajectory envelope and the obstacle faces. The complexity of this task depends on the nature of the trajectory envelope. A surface or even a line can be such as to make this intersection check numerically very expensive. It is therefore imperative that we look for additional constraints to further reduce the complexity of collision checking. Since obstacle faces are planes in cartesian space, if the trajectory surface were a plane in cartesian space collision checking would be simple. This chapter discusses trajectory primitives that will simplify collision checking.

For the trajectory surface to be a plane in cartesian space, the boom tip locus must be a straight line in that space. The cartesian space straight line locus for the boom tip was implemented for the 2D problem. There is an extension of the 2D algorithm for three dimensions. Unfortunately this 3D extension has no simple decomposition and is, computationally, an expensive solution. To understand why a natural extension of the 2D algorithm is computationally expensive requires an understanding of how boom tip loci are planned. The reader will therefore have to wait till Chapter 8 where boom planning is discussed. Section 8.4 on "Extending 2D Ideas" discusses the problems with cartesian space straight line loci for the 3D boom tip. We therefore settle for a straight line boom tip locus not in cartesian space but in boom space; and this is relatively easy to compute. Since straight lines in boom space have no linear counterpart in cartesian space, this curve is approximated by a sequence of straight lines in cartesian space. Forearm planning is done along these approximated sections so that plane faced trajectory surfaces are generated.

Chapter 2 stated that the collision detection and avoidance system's activities resulted in a list of typed intermediate configurations of the manipulator, where the type indicated the nature of the subsequent section of the trajectory. The constraints discussed in the first two sections of the chapter will be examples of the type specifications.

The first two sections of this chapter present the 2D and 3D trajectory primitives. The third section briefly describes the trajectory calculations that need to be done for running the hardware.

## 6.1 2D TRAJECTORY PRIMITIVES

The straight line in cartesian space constraint is implemented for the 2D system and so the discussion will be restricted to the 2D problem. The constraints require that the boom tip always move along straight line sections in cartesian space. The forearm is restricted to two types of motion. When the boom tip is stationary the forearm tip traces a circle, and when the boom tip is moving the forearm tip traces a locus that is a straight line parallel to the boom tip line.

The trajectory surface is a parallelogram when the boom is moving and is the sector of a circle when the boom is stationary. The first is a figure bounded by straight lines and the second has straight lines and a second degree curve for a boundary. Checking for intersection of polygons and the parallelogram is very quick. With the sector, the intersection checks are somewhat more expensive. Circles are expensive during intersection checks because a square root computation takes 10 to 20 times the time for a simple arithmetic operation.

## 6.2 3D TRAJECTORY PRIMITIVES

The constraints require that the boom joint angles be linear in each other during the motion from the start to the goal. The boom tip thus traces straight lines in boom space. Boom planning is now made numerically simple compared to the straight line in cartesian space trajectory.

The forearm motion is again restricted to one of two types depending on whether the boom tip is moving or is stationary. When the boom is stationary the forearm tip moves on the surface of a sphere such that the forearm is always in the same plane. When the boom tip is moving, we would like the forearm tip to move such that a linear trajectory surface is generated in cartesian space. Unfortunately, the cartesian space description of the straight line in boom space is nonlinear. To simplify matters, the boom space straight line is approximated by a sequence of cartesian space straight lines. Along any such cartesian space straight line segment the forearm tip moves along a straight line parallel to the segment (both the segment and the forearm tip being in the same cartesian space plane).

The trajectory surface is therefore a parallelogram when the boom is moving and is the sector of a circle when the boom is stationary. The parallelogram is easy to handle. The sector of a circle is somewhat more complex for the same reasons mentioned in the 2D case.

## 6.3 TRAJECTORY CALCULATIONS

The terms "trajectory" and "trajectory locus" were defined in section 1.2. The trajectory planning routines compute a trajectory locus. To generate a trajectory a sequence of positions and orientations along the trajectory locus is selected. The position problem is solved for each of the intermediate configurations. Interpolation polynomials are computed for each joint; these polynomials specify the behavior of the manipulator between intermediate configurations. The set of sequences of polynomials in time, one for each joint, specify the trajectory. The polynomial sequences are used by the trajectory servoing routines for running the hardware. If the intermediate configurations are "close" to each other, the trajectory will result in the manipulator tracing a curve in joint space that is close to the trajectory locus that was planned. This will guarantee safety of the manipulator. This section briefly reviews previous results on polynomial trajectories.

Polynomial trajectories are trajectories where the time histories of joint angles are specified as polynomial sequences. Polynomial trajectories have been extensively studied by Paul(1972) and Lewis(1974). The popularity of polynomial trajectories arises from the facts that continuity of joint variable position, velocity and acceleration can be guaranteed and the coefficients of the polynomials are calculable non-iteratively.

The five cubic polynomial sequence is used to compute the time history of a joint between any two intermediate positions. This trajectory appears to minimize the "wander" and "overshoot" problems that occur with other polynomial sequences such as the cubic-quartic-cubic or quartic-cubic-quartic [Lewis(1974)].

Joint angle limit violations have already been accounted for by the collision detection and avoidance routines. Joint angle acceleration limits, however, need to be taken care of. The extrema of the acceleration will occur at end points of the trajectory section because the acceleration of a cubic trajectory is linear. The ratio of the maximum acceleration to the limit acceleration can be determined for the relevant joints, the maximum of these ratios computed and the time interval scaled up proportionate to the square root of this acceleration rate.

This guarantees observation of the acceleration limits and eliminates the need to recompute polynomial coefficients. The details are presented in Lewis(1974).

The trajectory typing is used implicitly in various places. Consider the trajectory calculation in 3D for example. With straight line trajectories in boom space, only one polynomial sequence has to be computed. The other two joint angles are linearly related to the first one and so they are easily computed once the first one is known. However, polynomial trajectories have to be computed for each of the forearm joints. Cartesian space straight line trajectories are calculated by computing polynomial trajectories through a large number of points along the cartesian space line.

# CHAPTER 7
# GOAL FEASIBILITY AND IMPOSSIBLE SITUATIONS

The first section of this chapter discusses the goal feasibility analysis that is carried out before planning begins. The second section discusses how the planning system is constantly on the watch for situations where it would be unable to find a safe plan. If as a result of some partial planning activity the system realizes that a goal is unattainable, the system will immediately abandon the planning and inform the human supervisor of the failure.

## 7.1 FEASIBILITY STUDY

The static analysis is done before any planning is attempted. It includes boom placement safety and forearm placement safety checks. If the goal boom tip position is within a primary pasc, boom placement is feasible. Otherwise, the system repeatedly refines the area in the immediate vicinity of the goal until either the goal boom tip is within a pasc or the resolution limit is reached and the system returns complaining that the goal is not feasible. The area of the chart that undergoes refinement is similar in the two systems. For the 2D case the area is ten degrees on either side of the goal theta value. For the 3D case it is ten degrees on either side of the theta and phi values of the goal configuration. A Fibonacci incrementing scheme is used in the 2D case to determine the resolution at the next refinement. In the 3D case a simple doubling (binary) scheme is employed i.e. the angular width of pascs/sectoroids is halved at each new try.

The forearm feasibility check is very simple. If the forearm is safe from collisions in the final configuration of the manipulator, forearm placement is feasible. Otherwise the system complains.

## 7.2 IMPOSSIBLE SITUATIONS

Unlike the feasibility study of the last section, this section refers to analysis that is carried on during trajectory planning. Though the manipulator may be safe in the goal configuration, there is no guarantee that it can be maneuvered into that position. An example of such a situation is given in figure 7.1. The figure shows the start and goal configurations of the manipulator. The manipulator cannot get to G because the shortest distance between A and B along a line through the origin is less than the length of the boom. Such a situation cannot be identified by the goal feasibility analysis phase. It is identified during the mid-section planning by checking the safe limit intervals of all the regions that the boom tip locus passes through. If the safe limit interval is below a prespecified limit, further refinements of that region are attempted. In 2D the refinement is carried on until the safe limit interval exceeds the prespecified limit or the resolution limit is reached; in the latter case the system returns failure. In 3D the refinement is carried on to a resolution level eight times greater than the starting resolution of the region i.e. the region is refined three times. At the end of each refinement the safe limit interval is checked and if it is greater than the prespecified limit the refinement is terminated. If even at the end of three attempts the safe limit interval is below the limit, the trajectory is modified to pass through a neighboring region. The system can recognize situations when the start or goal boom tip position is completely enclosed by impassable regions since this occurs when there are no more neighboring regions available for subgoal placement. On such occasions the system complains that the goal is not feasible.

The above analysis ensures the feasibility of maneuvering the boom into the final configuration. Maneuverability of the boom into the final position does not ensure that it will be possible to maneuver the forearm safely along the proposed boom tip locus. Figure 7.2 gives such an example. In the figure S and G are the start and goal boom tip positions. The dotted line is the boom tip locus. At point A, the distance ABC is less than the length of the forearm, and thus there is no way to maneuver the forearm safely along the proposed boom tip locus. At points on the boom tip locus that are relatively far away from obstacle faces the forearm will not, in general, be the source of any insurmountable difficulties. Close to obstacles, as is often the case near the starting and goal configurations, freezing the boom tip locus with complete disregard to the forearm can lead to problems. This is the motivation for introducing the terminal phase planning stage as distinct from the mid-section planning phase. The terminal phase is responsible for planning of maneuvers close to obstacles and the mid-section phase deals with planning of maneuvers relatively far away from obstacles. This separation greatly reduces the chances of the forearm getting stuck.

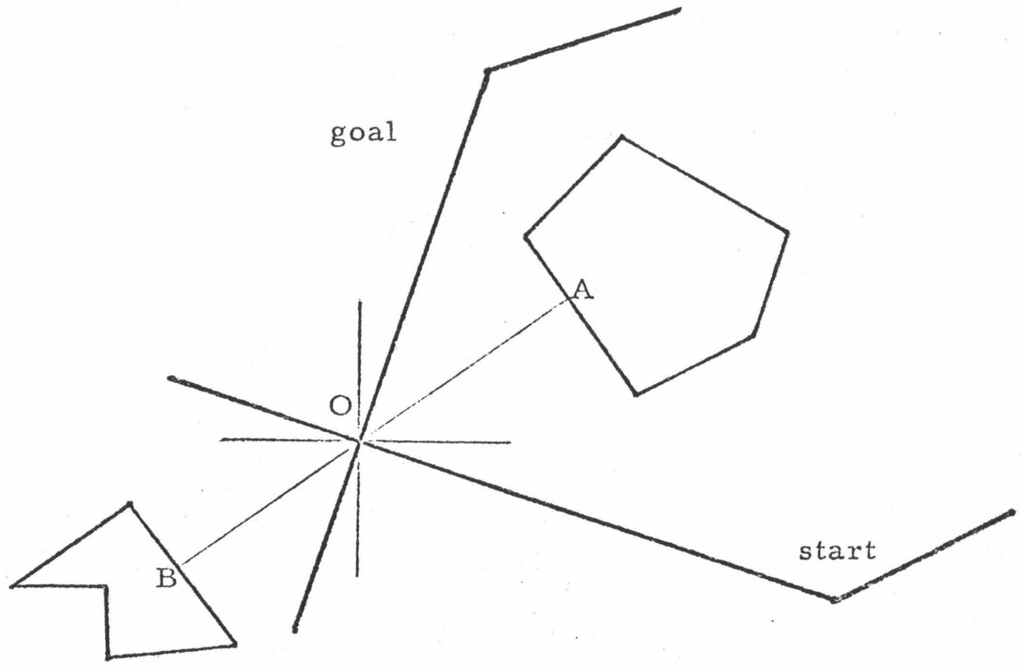


Figure 7.1 Blocked Boom

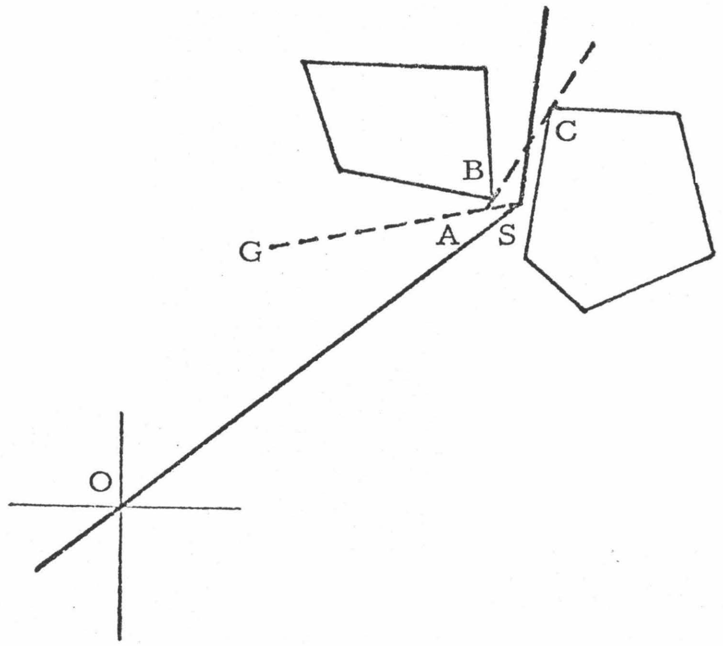


Figure 7.2 Blocked Forearm

# CHAPTER 8
# MID-SECTION PLANNING

An activity that a manipulator routinely performs is that of transporting objects. In the simplest case the object is resting on a flat support both in the initial and final states. These supports are obstacles that the manipulator must avoid bumping into. Of course the situation could be more complex. The object might be inside a cave like recess or may need to be deposited in a box etc. Specific obstacle configurations (cave, crater, channel etc.) suggest specific heuristics for maneuvering near or about them. Maneuvering in the absence of obstacles likewise suggests special heuristics.

Trajectory planning is decomposed into two different phases called <u>mid-section phase</u> and <u>terminal phase</u>. Each phase embodies a <u>planning strategy</u> and specific <u>heuristics</u>. Heuristics may be refined or added to the strategy without destroying the flavor of the strategy. Terminal phase planning uses obstacle configuration dependent heuristics while mid-section planning uses obstacle independent heuristics. The nomenclature arises from the observation that near the start and goal, obstacle configuration specific heuristics are most likely to be useful, while in between, the obstacle independent heuristics are probably more useful. Mid-section planning is discussed in this chapter while terminal phase planning and the interactions between the two are presented in the next.

Mid-section planning strategy views trajectory planning as consisting of boom planning and forearm planning. Of the two components boom planning is considered to be more important. Once a boom trajectory is planned, the planning of a forearm trajectory to follow the boom is attempted. If a safe forearm trajectory is found then trajectory planning is over. Otherwise the failure is analyzed and the analysis used to modify the boom trajectory. After boom trajectory modification another attempt at forearm planning is made. If the system cannot find a safe trajectory after a prespecified number of iterations, it returns a failure.

There are two good reasons for considering boom planning to be more important than forearm planning. First, the boom on the JPL arm is almost four times as long as the forearm. Thus the boom is likely to be the more constraining of the two. Second and more importantly, the notion of navspace (see Chapter 5) and its approximation by charts permits the boom to be treated as a single point. Thus boom planning is reduced to path planning for a point through the chart and does not involve any intersection checks that normally go with planning the motion of a finite sized link. This reduction makes the boom a natural candidate for planning before the forearm.

Mid-section boom planning operates in two modes. The first mode is used when terminal phase planning is also a part of the trajectory planning process and the second mode is used when mid-section planning alone is used to plan the complete trajectory. In the first mode mid-section planning is a one-shot affair and a direct trajectory from the start S to the goal G is planned. In the second mode, a safe point P is made a subgoal. A safe point is a point in a secondary pasc, or if the safe limit interval of the secondary pasc is very small, it is a point in a primary pasc whose safe limit interval exceeds a prespecified value. Planning then proceeds from S to P and from G to P. Using the reversibility principle the G to P trajectory is reversed and the two trajectory sections together form the complete solution.

The reason for introduction of these two modes of operation is best conveyed by an example. Suppose mid-section planning alone is used for the problem of Figure 1.1. If a direct trajectory from S to G is planned, the system will have no idea as to what orientation the forearm should be in as the manipulator approaches the goal configuration. Since, in the goal configuration, forearm maneuverability is severely restricted, a wrong choice of forearm orientation as the manipulator enters the channel near the goal will require expensive backtracking. To avoid this two direct trajectories, one from the S towards G and the other from G towards S are planned. The two trajectory sections are matched at a safe point P. If terminal phase planning were incorporated, the peculiarities of obstacle configurations near S and G would be handled by the terminal phase planning routines. The mid-section planning would then be responsible for maneuvering in areas away from obstacles and a direct trajectory computation would suffice.

## 8.1 PATH PLANNING FOR A POINT

Within navspace, the boom can be considered to be a point and boom planning reduces to path planning for a point through the chart approximating navspace. Point path planning is based on an adaptation of the well known algorithm for approximating a curve by a sequence of straight lines such that every point on the curve is within distance e from the line segment approximating the portion of the curve the point is on. The recursive algorithm is best explained using figure 8.1. Let line AB be the first approximation to the curve. Let point C be the point on the curve farthest from AB. If the distance of C from AB is within the tolerance limit then AB is the desired approximation. Otherwise the curve is split at C to form curve AC and curve CB. The algorithm is now applied recursively to sections AC and CB.

A straightforward adaptation of this simple algorithm serves as the basis for boom planning. Note that the above algorithm works even if different thresholds were used for different parts of the curve. This observation gives a clue as to how the linear approximation algorithm can be used for point path planning. Note that each region has associated with it an rmax and rmin that specify the safe interval limit, or an e, within which the boom tip must lie when it goes through that region.

Figure 8.2 shows how the modified linear approximation algorithm works. The dotted lines show adjacent regions of a part of a chart and their safe limit intervals. Every pair of neighboring regions has a non-trivial intersection of their (rmin, rmax) interval. S and G are the start and goal boom tip locations. Boom planning conceptually proceeds as follows. Join the start and goal boom tip positions by a straight line. Find the set of regions the line passes through. For each region compute the maximum and minimum r value of the points on the section of the boom trajectory passing through it. If the r values are within the (rmin, rmax) interval for every region the trajectory passes through, then boom planning is over. Otherwise determine the region where the violation is worst and introduce a point inside it as a subgoal and repeat the above process recursively. In Figure 8.2 the arrow points to the place where the violation is the worst and shows the subgoal P. The final desired locus consists of the two line segments SP and PS.

A further generalization of the simple recursive curve-approximation algorithm is possible. The approximating line segments need not be straight lines but can be any desirable curve. In fact for the 3D system the algorithm is used to plan a boom tip locus that is linear in the boom joint angles.

## 8.2 BOOM PLANNING IN 2D

### 8.2.1 Preliminaries

Boom planning is equivalent to finding the path of a point through the charts; the path consists of a sequence of cartesian space straight lines joining the start (S) and goal (G) boom tip locations.

The theta = 180 degrees position is a dead zone for the 2D boom. Given S and G there are two ways of getting to G. One of these would require going through the 180 degree line and this trajectory is avoided. The <u>permissible direction of travel</u> is the one that does not cross the 180 degree mark. The circular ordering defined in Appendix 2 works only for angles less than 180 degrees and so if the angular spread between S and G in the permissible direction of motion is more than 180 degrees, a subgoal is introduced near about the theta = 0 degree line. This would ensure that between any two subgoals the angular spread in the permissible direction of travel is less than 180 degrees. Note that subgoals are always introduced at safe points.

Furthermore, it is convenient to assume that the theta value of the goal is not less than the theta value of the start. In case it is not so, S and G are interchanged, a safe trajectory planned and finally the points along the locus reversed. The reversibility principle (section 2.3) justifies interchanging S and G and reversing the locus.

## 8.2.2 <u>The Main Algorithm</u>

This subsection describes the main algorithm for direct mid-section trajectory planning between two boom tip locations S and G. The algorithm is an elaboration of the scheme outlined in section 8.1. It is assumed that

1) S and G are such that the theta value of G is greater than the theta value of S,

2) the boom tip locus will be in the smaller of the two angular ranges between S and G, and

3) the theta = 180 degree line will not cross the boom tip locus.

Assumptions (2) and (3) are made valid by introduction of suitable subgoals.

Region list computation The first step of the algorithm computes the list of regions, called Rlist, the straight line SG passes through. Figure 8.3 shows an example where Rlist has five regions in it. Rlist is then sorted in increasing order of the minimum theta value of the regions. If the safe limit interval of any of these regions is less than a prespecified value, the region is refined until either the safe limit interval is large enough or the resolution limit is reached. In the latter case the trajectory planning is abandoned.

Trajectory fixing - 1 The second step attempts to get the straight line locus to lie within region safe limit intervals. Adjacent regions that have intersecting (rmin, rmax) intervals are grouped together. If Rlist is decomposed into more than one such group, subgoals are introduced in the boundary region of each group. Figure 8.3 shows two such groups, the first containing regions R0 and R1 and the second containing R2, R3 and R4. Two subgoals P0 in region R1 and P1 in region R2 are introduced. P0 and P1 are connected by a radial line. Simple heuristics decide on the r value of P0 and P1. At the end of this process the boom tip locus will be a sequence of straight lines SP0, P0P1 and P1G. Each element of this sequence is either radial or passes through a set of regions where every pair of neighboring regions has a non-trivial intersection of their (rmin, rmax) interval. The radial section of the trajectory is safe and the algorithm of section 8.1 is applied to each non-radial section.

Trajectory fixing - 2 At the end of the second step, the boom tip locus is a sequence of straight lines and each element of this sequence is either radial or non-radial. The non-radial section lies within the (rmin, rmax) intervals of the regions it passes through. The third step is very similar to the second step and operates with sectoroids instead of regions.

The sequence of straight line segments obtained after the three step planning process is the desired safe boom tip locus.

Extreme r values To simplify computing the extreme r values of a straight line through a region, it is seen that the foot of the perpendicular from the origin to the straight line does not fall inside the line. Figure 8.4

shows examples where the dotted lines show three adjacent regions. S and G are the start and goal boom tip locations and the foot of the perpendicular to SG from the origin is at P. P is inside SG.

If the foot is inside the line and is also inside the (rmin, rmax) interval of a primary sectoroid, then the foot is treated as a subgoal. In Figure 8.4(a) P is inside a sectoroid (solid boundaries) and is introduced as a subgoal. If the foot is inside the line but not inside a primary sectoroid, a subgoal is introduced at the same theta value as the foot such that the foot is within a primary sectoroid and this algorithm is repeated recursively for the two straight line sections so obtained. Figure 8.4(b) shows P lying outside the sectoroid r-limits. Q is therefore introduced as a subgoal and the foot of the perpendicular routine applied recursively to SQ and QG. Simple heuristics decide on the r value of Q.

At the end of this process it is assured that the extreme r values of any section of the candidate boom tip locus is at the end points of the section. This operation is carried out on every non-radial section of the boom tip locus.

## 8.3 FOREARM PLANNING IN 2D

The boom tip locus is a sequence of straight line sections. No forearm planning is required for parts of these sections that lie within the secondary chart. This section first discusses forearm planning for a segment of the boom tip locus that is completely outside the secondary chart. Subsection 8.3.3 contains a discussion on how this scheme is used to handle boom tip locus sections that are partly inside and partly outside the secondary chart.

The forearm is restricted to two types of motions called the circle and pgram motions. During <u>circle motion</u> the hand traces a circle and the boom is stationary. During <u>pgram motion</u> the hand traces a line parallel to the boom tip locus generating a parallelogram for the trajectory surface. Pgram motions occur when the boom moves. Forearm motions consist of sequences of pairs of circle and pgram motions. Circle motion computations determine the best forearm orientation the forearm can be placed in for the subsequent pgram motion. The pgram motion continues until the parallelogram generated by the forearm motion is just short of touching an obstacle or the end of the current boom tip locus section is reached. If the former happens then another pair of circle and pgram motions follows.

After every pair of circle and pgram motions a check is made to see whether the manipulator has advanced. If the manipulator joint angles are different then planning is continued. Otherwise forearm planning is abandoned and the system returns to the top level requesting a modification of the boom trajectory. Implementation of the boom tip trajectory modification has not been completed. In the current implementation the system only prints a failure message.

## 8.3.1 Circle Motion

Figure 8.5 illustrates definitions connected with circle motion. The forearm joint angle of -pi (in practice, the lower forearm joint angle limit) is called the most favored orientation. The safety of the boom trajectory having already been guaranteed, if the forearm is placed at its most favored orientation, the chances of a forearm collision would be reduced tremendously. However it may not be possible to achieve such a forearm placement due to obstacles. The angular interval, about the forearm's current orientation, over which it can move safely is called the S-interval (for safe interval). With regard to forearm placement, the best that can be done is : maximum (most favored angle, minimum (S-interval) ). Circle motion computations determine this angle.

Now, the forearm circle is defined as a circle with center at the boom tip and the length of the forearm as radius. To determine the S-interval, the system first computes the r-theta envelope of the forearm circle. Next it makes a list of all the obstacles whose r-theta envelope intersects the forearm circle's r-theta envelope. For each obstacle it determines the forbidden phi interval, which is an interval of phi values for which there is a forearm and obstacle collision. The complement of the union of the forbidden phi intervals is a set of safe phi intervals. Of this set, the interval that contains the current forearm phi value is the one that determines the forearm mobility. The forbidden phi interval of an obstacle is the union of the forbidden phi intervals of its edges. This is computed by determining the phi values of the intersection of the edge with the forearm circle. Note that an edge is ignored if it is completely outside the forearm circle.

## 8.3.2 Pgram Motion

Figure 8.6 illustrates definitions connected with pgram motion. During pgram motion the boom is moving and the forearm tip traces a line parallel to the boom tip locus. In Figure 8.6 GS and F1F2 are the boom tip and forearm tip loci respectively. Such a forearm motion generates a parallelogram, called pgram, and hence the name pgram motion. Pgram motion computations determine how far this parallelogram can be extended along the current boom tip locus section. Two terms that will be used in the following discussion will now be defined. The parallelogram whose base is the full length of the current boom tip locus section is called the <u>full pgram</u>. The line collinear with the forearm at the begining of the current boom tip locus section is called the <u>I-line</u>, for initial line. In Figure 8.6, parallelograms GSF1F2 and PSF1Q are the full pgram and pgram respectively and SF1 is the I-line.

The system first determines the r-theta envelope of the full pgram. Next it makes a list of all obstacles whose r-theta envelope intersects the above envelope. For each obstacle, the point on its boundary and within the full pgram and closest to I-line is called the <u>danger point of the obstacle</u>. The <u>danger point of an edge</u> is similarly defined. From amongst the set of danger points, the one closest to I-line represents the point the forearm will first collide with if it attempts to trace the full pgram. The corresponding point (P, in Figure 8.6) on the boom tip locus determines the extent of safe travel along the current boom tip locus and the given initial forearm orientation. A circle motion at this point will reorient the forearm and ready the system for another pgram motion.

Danger point determination The danger point of an obstacle is the danger point of one of its edges. The danger point of an edge is determined by finding the point on the section of the edge that lies inside the full pgram and is closest to the I-line.

A computationally simpler scheme for determining the danger point of obstacles is the following: Define ftlocus to be the finite line traced by the forearm tip as it moves. In Figure 8.6 line F1F2 is the ftlocus. Consider the set of all corners of the obstacle that are inside the full pgram, and the points of intersection of every edge of the obstacle and ftlocus (both treated as finite line segments). The point in this set that is closest to I-line is the danger point of the obstacle.

## 8.3.3 Partial Forearm Planning

This subsection describes forearm planning along boom tip locus sections that are partly inside and partly outside the secondary chart. The two cases that need to be considered are one where the end of the section alone is within a secondary chart and the other where the beginning alone is inside. The first case is handled by the general techniques outlined earlier in this section. In the second case the circle motion computation can be eliminated and the forearm placed in the most favored orientation. Being within a secondary chart guarantees the safety of the forearm independent of its orientation.

## 8.4 EXTENDING 2D IDEAS

Cartesian space straight line trajectories for the boom tip worked very well for the 2D problem. Unfortunately they don't work as well for the 3D system. This section discusses in some detail the reasons for their failure. The reasons are not obvious. It was only after a substantial part was implemented that some of the issues were cleared. The section also provides the motivation for the choice of boom space straight line trajectories for the boom tip. The section concludes with a discussion of extensions of 2D forearm planning to 3D.

### 8.4.1 Cartesian Space Straight Lines

Section 5.8 on "Extending 2D Ideas" discussed the reasons for introducing a uniform grid of regions to approximate navspace. A simple extension of the 2D ideas will result in the following algorithm for boom planning using cartesian space straight lines. Compute the list of all the regions the trajectory will go through. Next, for each region compute the maximum and minimum r values of points along the section of the trajectory through it. The analysis then proceeds as outlined in section 8.1. If no part of the trajectory lies outside the (rmin, rmax) intervals of the regions it goes through the trajectory is safe. Otherwise a subgoal is introduced at the worst violation point and the algorithm applied recursively to the two halves of the trajectory.

As a consequence of operating in 3D the above algorithm has two severe problems. The first is the computation of the list of regions the trajectory passes through. This is an expensive computation that has to be repeated every time a subgoal is introduced because the region list changes with a change in the boom tip locus.

The second problem is the determination of the maximum and minimum r values along a section of the trajectory that lies inside a region. These problems arise because one of the region boundaries is the constant theta surface which is a cone, a second degree surface. Determining the end points of a trajectory section inside a region requires computing the intersection of a straight line with a second degree surface in cartesian space. This requires a square root computation and square root computations are 10 to 20 times more expensive than simple arithmetic operations.

Note that in the 2D system the counterpart of the constant theta surface is absent and therefore all the computations were just solutions of linear equations. One solution to the second problem is to conservatively approximate the region boundaries by a minimum bounding viewbox with cartesian space planar surfaces. Having to compute these surfaces every time trajectory planning is done is very expensive. On the other hand, saving the computations will cost in storage. The solution is no longer attractive when one considers the fact that such an approximation has to be done at three levels - regions, sectoroids and pascs.

There is more than the square root computation that makes cartesian space straight lines bad and that is the constant conversion between the boom space and cartesian space at every point in the planning stage.

There is a partial solution to the above problems and it involves the definition of a region. The constant phi surface is linear in cartesian space and so if the constant theta surface were replaced by another surface that is linear in cartesian space all the overhead associated with the square root computations and the conversion between the two spaces (boom and cartesian) would be avoided. A linear surface to replace the constant theta surface is the surface's tangent plane that is mid-way between the region's phi boundaries. Regions are now no longer boxes in r-theta-phi space but are bounded by constant phi surfaces and planes through the origin that are tangential to cones mid-way between the phi-boundaries. Sectoroids and pascs also have a similar structure. The complexity of generate-region, generate-chart and refine-chart are quite close to that of the same operators with the older definition. The storage requirements are more, however, because storing characteristics of a plane through the origin requires three words while a constant theta surface needs only one word of storage.

There is one problem that the above representation of a region has not yet solved and that is having to recompute the region list every time a subgoal is introduced. This recomputation has to be done at the level of sectoroids and pascs too. The overhead of this need for repeated recomputation is very significant. Thus though part of the problems associated with the constant theta surface were solved, cartesian space straight lines for the boom tip locus had to be abandoned.

## 8.4.2 Boom Space Straight Lines

The better solution is to choose boom-space straight line trajectories for the boom tip and stick to the original constant theta and constant phi boundaries of regions. The advantages are several. Space requirements are lesser for the original region definition. Constant theta and constant phi surfaces are very simple surfaces in boom space and so determining maximum and minimum r values along a trajectory section inside a region is trivial. Subgoal introduction without having to recompute the region list is possible with proper decomposition of the planning process. This is achieved by planning the boom tip locus first in the theta-phi space alone and then in r-space. The 3D point path planning problem is thus reduced to a two-dimensional plus a one-dimensional problem which is far simpler than anything that was possible with the cartesian space straight line trajectory algorithms.

The reader might, at this stage, wonder why boom space straight line trajectories were not chosen in the first place. I touched upon this matter in the introduction to Chapter 6 on "Trajectory Modelling and Calculation". Recall that the notion of navspace permits the trajectory envelope to be reduced to a single surface and that collision detection requires determining the intersection of this surface with obstacle faces. The complexity of this task depends on the nature of the trajectory surface. Since obstacle faces are planes in cartesian space, intersection checking would be simple if the trajectory surface were a plane in cartesian space. Now, the trajectory surface will be a plane in cartesian space if

(1) the boom tip locus is a straight line in that space, and

(2) the forearm were then restricted to move such that the hand traced a line parallel to the boom tip locus. This is what led us to try cartesian space straight lines for the boom tip locus.

Now that we have settled for boom space straight line trajectories for the boom tip locus what happens to the complexity of the trajectory surface? Since boom space straight lines have no simple representation in cartesian space, the trajectory surface will be complex. But the boom tip locus can easily be approximated by a sequence of cartesian space straight lines and the forearm planning routines need not know how the boom tip locus was arrived at. This will make the trajectory surface a set of planes and collision checks should therefore be simple.

One final point that is worth noting is that even though we have to approximate the boom tip locus by cartesian space straight lines, it needs to be done only once for every boom trajectory planning. Thus the conversion overhead problem identified at the begining of Chapter 4 is reduced to a one-time conversion overhead. The advantage arises because an expensive computation is pulled out of a loop and is done outside.

### 8.4.3 Forearm Planning

In 2D the forearm is restricted to circle and pgram motions. The natural extension of the circle motion would permit the forearm motion to move anywhere on the surface of a sphere with length of the forearm as radius and the boom tip as the center of the sphere. Of course this motion would be subject to joint angle limits. The natural extension of the pgram motion would require that the forearm tip trace a line parallel to the boom tip locus and thus generate a parallelogram for a trajectory surface.

Now, trajectory hypothesis and modification, and collision detection are expensive if the forearm tip is allowed unrestricted motion on the surface of the above mentioned sphere. Therefore we have to impose additional constraints on the forearm motions to keep the planning problem tractable. The constraint we impose is to require that the forearm always travel in a cartesian space plane. Thus piece-wise linear cartesian space planar surfaces are the only surfaces that the forearm may generate. With this restriction, forearm planning is essentially similar to the 2D forearm planning problem. In fact forearm planning is the only component of the 3D solution that is obtained as a natural extension of the corresponding component of the 2D solution.

## 8.5 BOOM PLANNING IN 3D

Boom planning in 3D is also equivalent to finding the path of a point through the charts. This time the path consists of a sequence of boom space straight lines from the start(S) to the goal(G) boom tip locations.

There is a lot of similarity between the 2D and 3D planning systems. First, there is a permissible direction of travel defined for the phi joint. This direction avoids the (-175, 175) degree dead zone for the phi joint. Second, a subgoal is introduced, if necessary, near the phi=0 plane to ensure that between any two subgoals the angular spread in the permissible direction of travel is less than 180 degrees. Lastly, steps are taken to ensure that the phi value of G is not less than the phi value of S.

There is a difference between the two systems in that the 3D manipulator has the lateral property. The goal being specified as a cartesian space position and orientation of the hand, the manipulator can be at the goal in either the left- or right-handed configurations. The initial choice is the same lateral configuration at the goal as at the start. If it is not possible to find a trajectory to the goal maintaining the same lateral configuration a subgoal with theta=0 is introduced. With theta=0 the boom is vertical. From the vertical position, the boom can turn the manipulator into either a left-handed or a right-handed arm and this makes configuration switching possible.

Thus, all that needs to be described is boom path planning with no configuration switching and the phi angular spread less than 180 degrees. Boom space straight line trajectories are planned first in the theta-phi space and then in the r space. The details are described below.

### 8.5.1 Theta-phi Space Planning

The discussion here deals with the two-dimensional theta-phi space and a straight line means a curve linear in theta and phi. A straight line joining S and G is chosen as the desired locus. A list of primary regions through which this line passes is computed. Certain minimal checks on the safe limit intervals of the regions are made. If for example a region is impassable a fixed number of attempts are made to further refine the region. If the region is still impassable, subgoals are introduced to avoid this region. Note that this is possible only because it is in 3D. The heuristics minimize the number of subgoals and aim for subgoals in regions with large safe intervals. Figure 8.7 shows how a region R is avoided. In Figure 8.7(a) R can be avoided by introducing the subgoal A or two subgoals P and Q. Subgoal A is chosen since the number of subgoals is smaller. In Figure 8.7(b) since the number of subgoals is identical, the choice is made on the basis of the safe limit intervals of the regions in which the four points lie. If the start or goal boom tip position is completely boxed in by impassable regions the system complains that the goal is not feasible.

### 8.5.2 R space planning

The theta-phi space planning resulted in a sequence of theta-phi space straight lines which passed through regions whose safe limit intervals were above a prespecified threshold. R joint planning is next done at three levels - region, sectoroid and the pasc level. At each level two things happen. First a list is made of the elements - region, sectoroid or pasc - the trajectory passes through.

Next the trajectory goes through a refining process identical to the one described in "Trajectory Refining-1" of section 8.2.2. The only difference is that boom space straight line loci are used instead of cartesian space straight lines. Hence the determination of the extreme r values of the section of the locus through an element is trivial - the extreme values are at the end points.

The sequence of boom space straight line segments obtained after the theta-phi space and the three level - region, sectoroid and pasc - r space planning is the desired safe boom tip locus.

## 8.6 FOREARM PLANNING IN 3D

The boom tip locus is a sequence of boom space straight line sections. No forearm planning is required for parts of these sections that lie within the secondary chart. The parts that are outside are approximated by cartesian space straight lines. The forearm planning for a cartesian space straight line segment of the boom tip locus that is completely outside the secondary chart is discussed in this section. The simplifications that are possible as a consequence of the boom tip locus segment being partly within and partly outside the secondary chart is exactly similar to the 2D case described in section 8.3.3 and so will not be described again.

As in the 2D case the forearm is restricted to two types of motions called the sphere and pgram motions. During sphere motion the boom is stationary and the forearm moves in the plane formed by the lines passing through the initial and final forearm configurations. During pgram motion the boom is moving and the hand traces a line parallel to the cartesian space straight line approximating the boom tip locus. Pgram motion generates a parallelogram for a trajectory surface and hence the name. Sphere and pgram motions are counterparts of the circle and pgram motions of 2D.

The boom tip locus is approximated by straight line segments. Consider the plane generated by the movement of the boom when its tip moves along a straight line section. The forearm joint angles fphi, which will place the forearm in this plane, and ftheta=-90 is called the most favored orientation. The safety of the boom trajectory having already been guaranteed, if the forearm is placed at its most favored orientation, the chances of a forearm collision would be reduced tremendously. The presence of obstacles might make it difficult if not impossible to maneuver the forearm from its current orientation into the most favored orientation.

If the forearm is not already in the most favored orientation, there is an infinite number of ways to maneuver the forearm into this orientation. The one we permit is the simplest for determining whether the forearm will collide with an obstacle when attempting to do so. It requires that the forearm travel in the plane determined by the lines passing through the initial and the most favored orientations. Such a plane is called the <u>forearm</u> <u>plane</u>. Sphere motion computations determine how far the forearm can travel in this plane before a collision will occur. Having placed the forearm in this "best" orientation, a pgram motion is attempted.

The pgram motion computations determine how far the forearm can travel, generating the parallelogram trajectory surface as it goes along, before a collision will occur or the end of the current boom tip locus section is reached. In the former case another pair of sphere and pgram motion follows. After every pair of sphere and pgram motions a check is made to see whether the manipulator has advanced. If the manipulator joint angles are different then planning is continued, otherwise forearm planning is abandoned and the system returns to the top level requesting a modification of the boom trajectory.

As mentioned in section 8.4.3, with the restriction that the forearm movement be such that it trace a planar surface, 3D forearm planning is similar to 2D forearm planning. 3D forearm planning requires computations similar to the 2D forearm planning computations and similar to the intersection checking of planes conducted by MSBE described in section 5.8.2. Since feasibility of the ideas involved in forearm planning has been shown by implementations elsewhere, forearm planning for the 3D system was not implemented.

## 8.6.1 <u>Sphere Motion</u>

The sector traced by the forearm if it moved from its starting orientation to the most favored orientation is called the <u>forearm plane sector</u> (FTS). Figure 8.8 shows an example of a FPS. O is the boom tip, OS and OG are the starting and most favored orientations of the forearm.

The system first determines the r-theta-phi envelope of FPS. Next it makes a list of all the obstacles whose r-theta-phi envelope intersects the above envelope. Every face of every obstacle in this list that has its r-theta-phi envelope intersecting the r-theta-phi envelope of FPS is analyzed. If the face and FPS do intersect, the point on the line of intersection that makes the smallest angle (equivalently, the smallest tangent of the angle) with line OS is computed. The minimum of such angles determines the limit to which the forearm can be safely maneuvered in the forearm plane.

Since the boundary of FPS is a second degree curve, determining the point that makes the smallest angle with OS requires square root computations. To speed up sphere motion computations, FPS can be approximated by a sequence of triangles, each triangle subtending an angle, say, no larger than 30 degrees at O. Determining the point of greatest constraint will then be reduced to solving linear equations. The saving in speed has been achieved at the cost of a conservative approximation to the area swept by the forearm.

### 8.6.2 Pgram Motion

As in the 2D case, we define the terms I-line and full pgram. The line collinear with the forearm at the begining of the current boom tip section is called the *I-line*, for initial line. The *full pgram* is the cartesian space parallelogram whose base is the full length of the current boom tip locus section and one of whose sides is I-line.

The analysis of the extent to which the forearm can travel before a collision will occur proceeds exactly like the analysis for sphere motion computations. Instead of FPS use the full pgram and instead of determining the smallest angle a point on the line of intersection (of the face and the plane) makes with OS, determine a point closest to I-line.

The point on the boom tip locus corresponding to the limiting safe forearm position determines, for the given initial forearm orientation, the extent of safe travel along the current boom tip locus section.

## 8.7 SUMMARY AND CONCLUSIONS

This chapter introduced the notions of mid-section and terminal phase planning. Mid-section planning deals with maneuvering the manipulator in regions relatively far away from obstacles. Section 8.1 discussed a well-known linear approximation algorithm and an adaptation of it that serves to make it useful for boom planning. Section 8.2 discussed cartesian space straight line trajectory planning for the boom tip and section 8.3 introduced primitives for forearm motions in 2D. Section 8.4 discussed obvious extensions of 2D ideas. It showed that in 3D boom space straight line boom tip loci are better than cartesian space straight line loci, and that the 2D forearm planning would work for 3D. Section 8.5 discussed 3D boom planning and section 8.6 3D forearm planning. Chapter 9 presents terminal phase planning.

The heuristics used in planning are very simple. It is the powerful representation schemes that lets the system plan safe trajectories within reasonable computation times. This dissertation is an example of the observation that with better and more elaborate models of the environment the system can get by with simpler and simpler planning strategies.

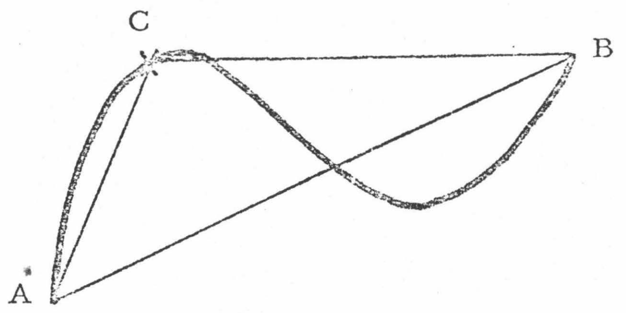


Figure 8.1 Linear Approximation

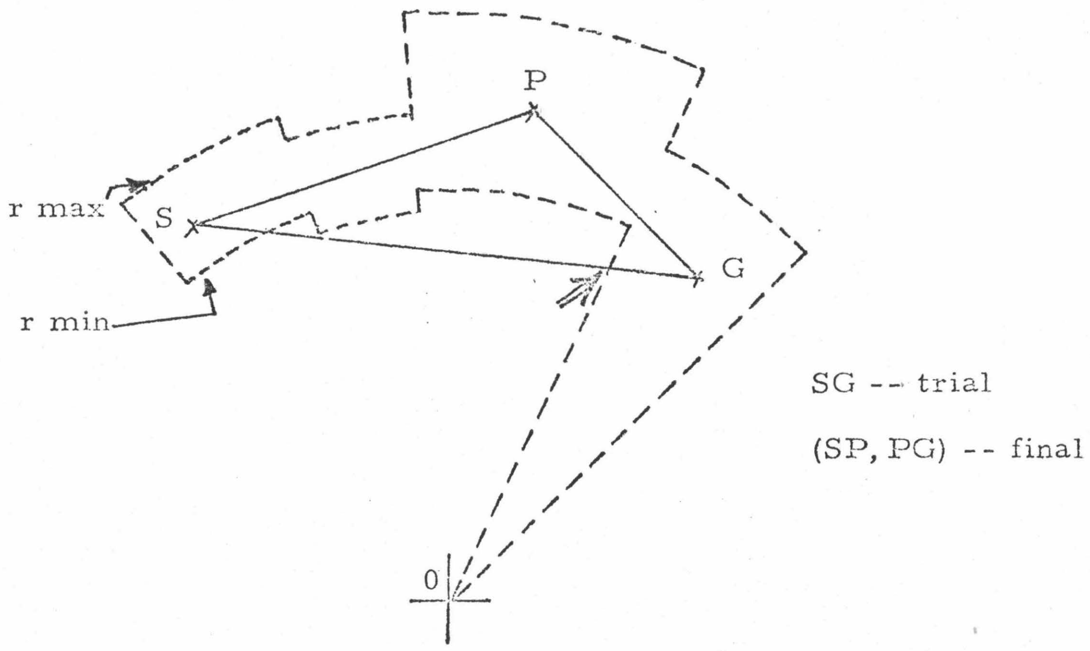


Figure 8.2 Modified Linear Approximation

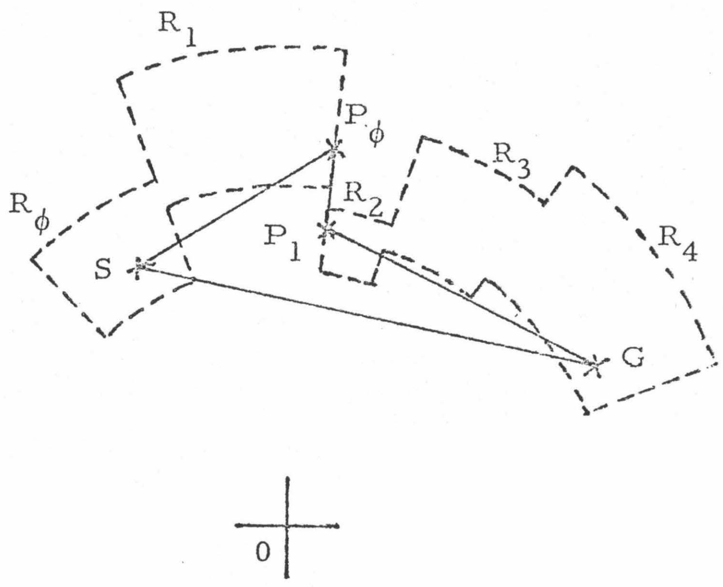
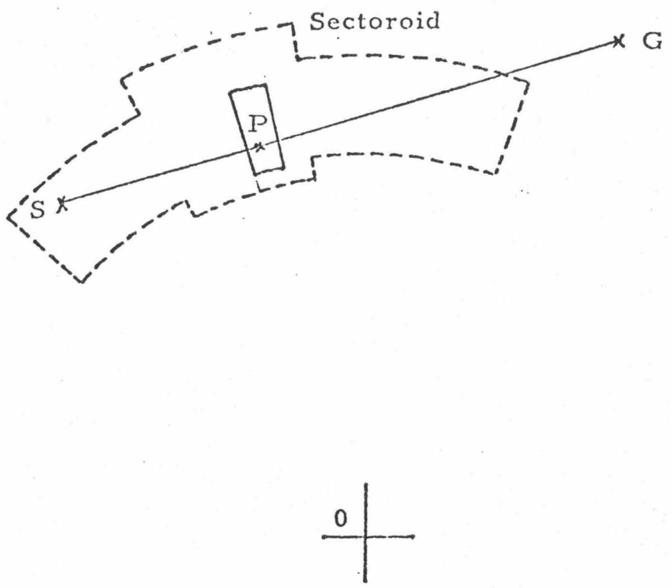
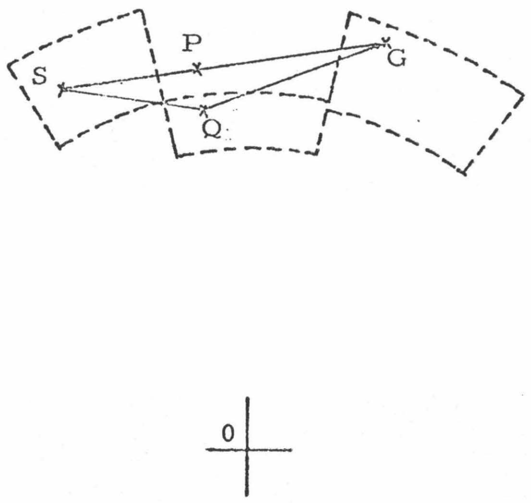


Figure 8.3 Trajectory Fixing



(a) Foot of perpendicular inside



(b) Foot of perpendicular outside

Figure 8.4 Extreme r Values

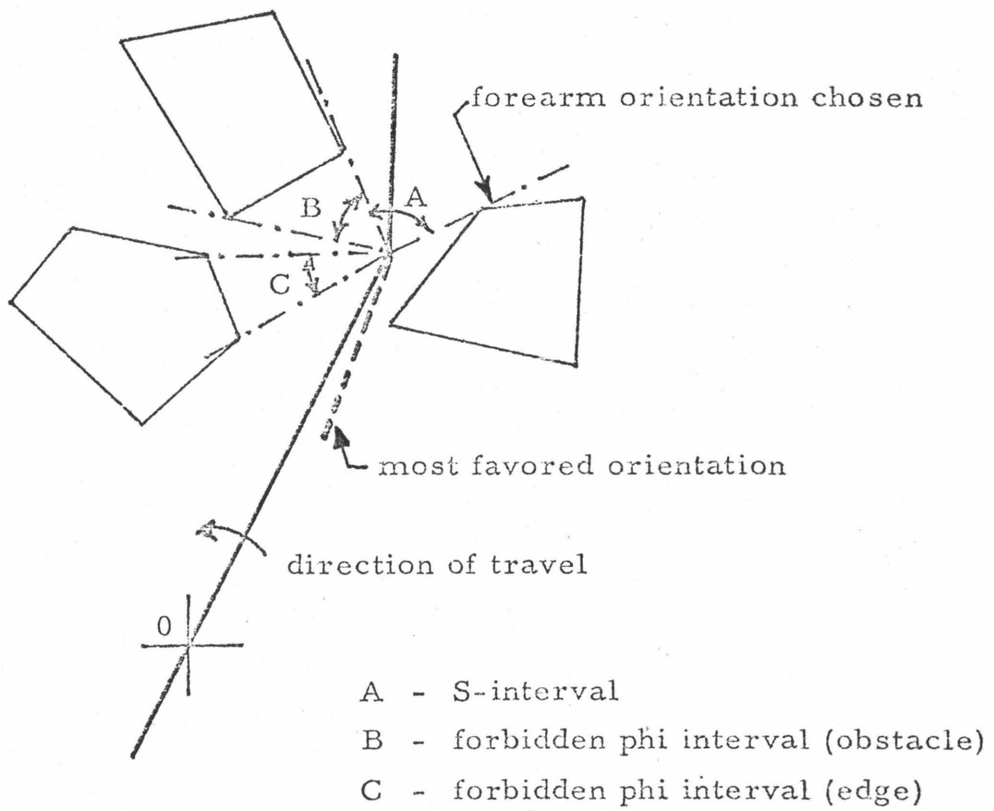

forearm orientation chosen
A
B
C
most favored orientation
direction of travel
0
A - S-interval
B - forbidden phi interval (obstacle)
C - forbidden phi interval (edge)

Figure 8.5 Circle Motion

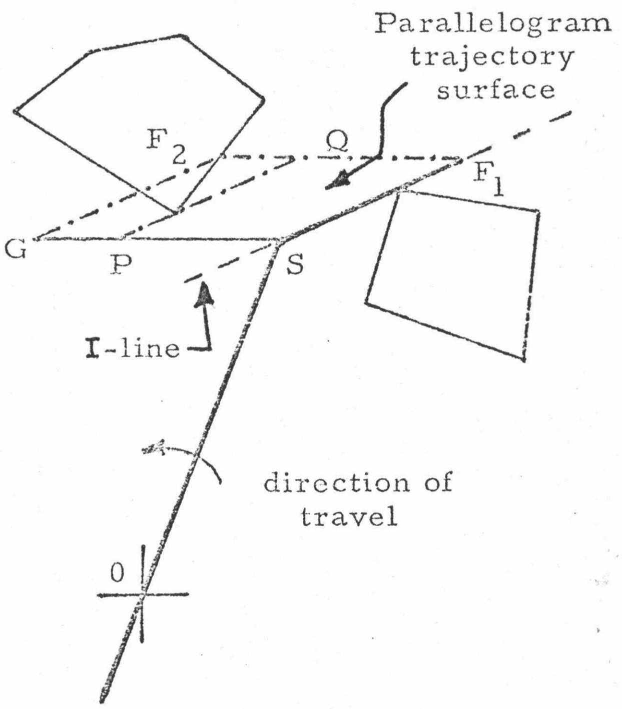
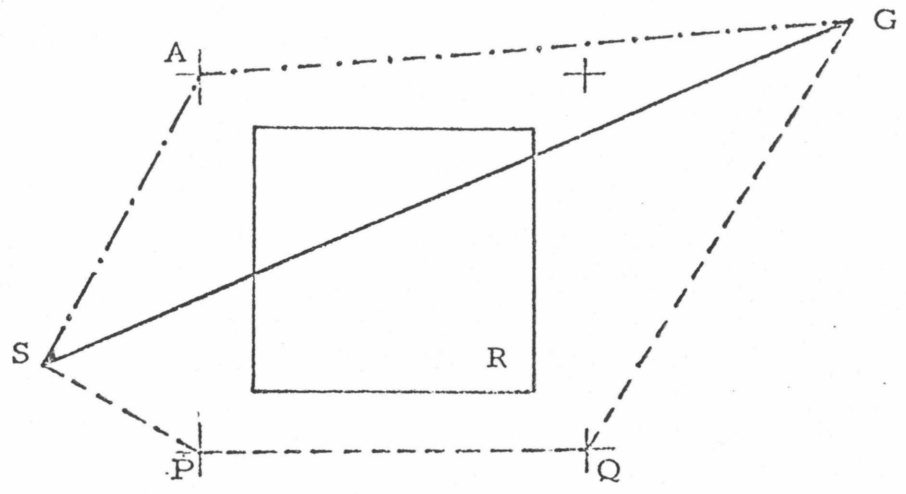
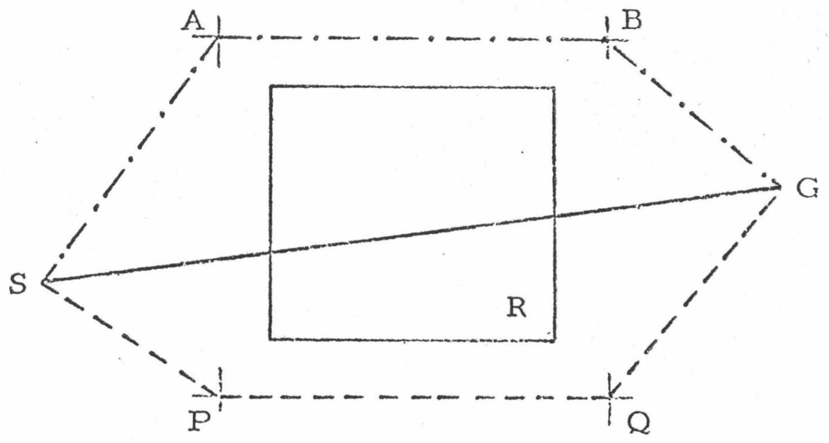


Figure 8.6 Pgram Motion



(a)



(b)

Figure 8.7 Theta-phi Space Heuristics

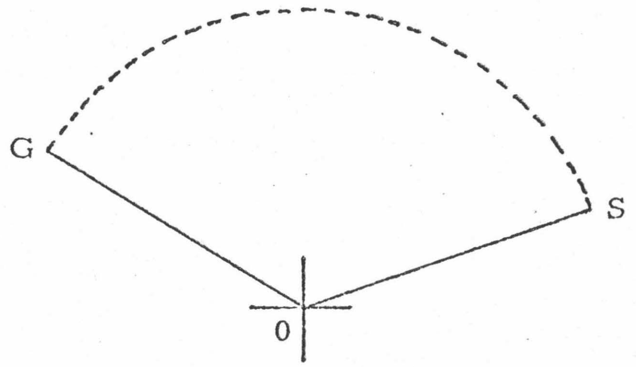


Figure 8.8 Forearm Plane Sector

# CHAPTER 9
# TERMINAL PHASE PLANNING

Trajectory planning is decomposed into two different phases called mid-section phase and terminal phase. Mid-section planning was presented in Chapter 8 and this chapter deals with terminal phase planning. Each phase embodies a planning strategy and specific heuristics. Heuristics may be refined or added to the strategy without destroying the flavor of the strategy. Terminal phase planning uses obstacle configuration dependent heuristics and the nomenclature arises from the observation that near the start and goal obstacle configuration specific heuristics are most likely to be useful.

Unlike the mid-section phase strategy, the terminal phase strategy consists of planning pairs of adjust and move motions. A sequence of such pairs of motions puts the manipulator in a state from which the mid-section strategies can take over.

The move motions are simple. The boom tip moves along a line collinear with the forearm and away from the hand, and the forearm maintains its orientation in cartesian space. This motion continues until either the boom tip reaches a safe point (and terminal phase planning is over) or a potential collision is recognized. In the latter case, the system proceeds with another adjust and move motion pair. At the end of every such pair of motions a check is made to see that progress is being made. If the manipulator joint angles remain unchanged, the system returns a failure.

The adjust motion orients the forearm to reduce the chances of a collision during the subsequent move motion. For this motion, the nature of obstacle configurations is more important than the nature of the obstacle itself. Thus the fact that the obstacles form a cave like structure is more important to terminal phase planning than the fact that one of them is a prism. Obstacle configurations can be very neatly classified and the process of identifying them gives, for any boom tip location and forearm orientation, the amount by which the forearm can move in a given direction.

## 9.1 2D CHANNELS

The world the forearm sees, for a given boom tip location, is a circle with center at the boom tip and length of the forearm as radius. For forearm collision checking purposes this world can be characterized by S-interval. S-interval was defined in section 8.3 to be the angular interval, about the forearm's current orientation, over which it can move safely. If the end points of the S-interval are not the forearm joint angle limits then the forearm is restricted in its motion on both sides, not by joint angle limit stops but by obstacles. Such an interval is referred to as a <u>2D channel</u> and is shown in Figure 9.1. The adjust motion heuristic for a 2D channel is to position the forearm in the center of the 2D channel.

## 9.2 3D CONFIGURATION TYPES

In three dimensions the world of the forearm, for any given boom tip location, is the sphere with center at the boom tip and the length of the forearm as radius. For forearm collision checking purposes this world is characterized by S-area. <u>S-area</u> is a 3*3 square in fphi-ftheta space and is centered at the (fphi, ftheta) value of the forearm. Each square of S-area is of side 5 degrees. It has a 0 or 1 associated with it according as it is safe or not for the forearm to maneuver within the solid angle represented by the square. Figure 9.2 shows an example of S-area with center at (pi/4, pi/4) and where the size of the component square is 5 degrees. A non-maneuverable square is shown shaded.

The maneuverability considerations take into account forearm joint angle limits. Thus a square in S-area will be not maneuverable if joint angle limits would be violated when the forearm is inside it. Elsewhere the constant fphi and constant ftheta surfaces are first conservatively approximated by planes. Computations are then carried out to determine whether any part of an obstacle lies within this pyramid and if so the original square in S-area is marked non-maneuverable.

The obstacle configurations in the immediate vicinity of the forearm are classified according to the S-area patterns. Figure 9.3 shows some patterns and their names. Figure 9.3(a) is a crater; it could be a box too. S-area shows that the forearm is enclosed on the top, bottom, the sides and the front. Figure 9.3(b) is an arch. The forearm is now enclosed on the top, bottom and the sides but not in the front and back. Figure 9.3(c) is a right overhang and the forearm is enclosed on the top, bottom and the left. Similarly, Figure 9.3(d) is a left overhang. Figure 9.3(e) is a channel in 3D, having side and bottom enclosures. Other configuration types can similarly be identified.

The adjust motion heuristics for these different obstacle configurations are determined in a straightforward manner. When inside a crater or an arch the forearm retains its current orientation. For all other configurations, the forearm moves to the boundary between the center square and the best neighbor square. The best neighbor square is defined as the one that is maneuverable and has the maximum number of maneuverable neighbors. If there is more than one such candidate one of them is picked up at random.

## 9.3 COMMENTS

Terminal phase planning uses the adjust and move motions described in this chapter. The adjust motion computations for the 3D problem are expensive if done in software. This is because the maneuverability of 9 squares has to be determined and each of them is an involved computation. Terminal phase planning has not been incorporated into either the 2D or 3D systems. Incorporating them into the 2D system will be simple but the 3D system would require considerable programming effort.

We saw that terminal phase planning using software is expensive. If terminal phase planning were done at execution-time, the computations could be speeded up with some hardware support. I am suggesting the use of proximity sensors with a sensitive volume spread over the solid angle subtended by a element of S-area. The sensor turns on whenever there is an object within (1) the solid angle monitored, and (2) within a distance equal to the length of the forearm. The sensors are all rigidly attached to the forearm. The logic for analyzing real-time data from an array of 9 or even 25 of these sensors is quite simple; the logic incorporates the simple adjust motion heuristics described in the last section.

In section 1.1, when describing a programmable manipulator system, I had indicated that very little is known about modifying trajectories dynamically based on sensory data the system may acquire during execution. The use of proximity sensors described above would be a step in investigating execution time strategies.

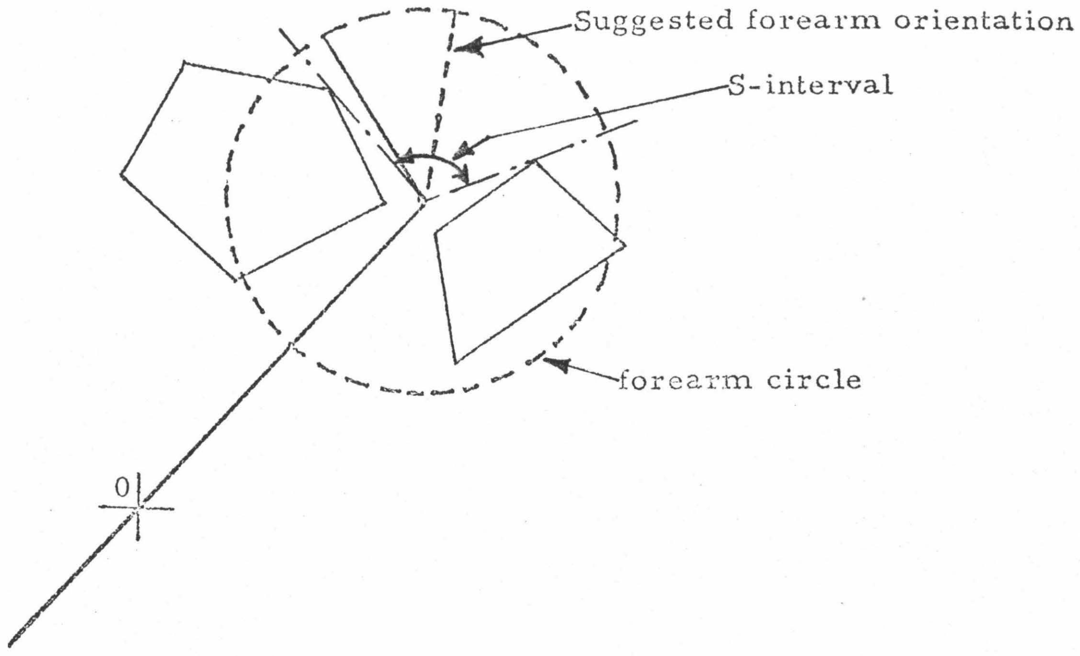


Figure 9.1 2-D Channel

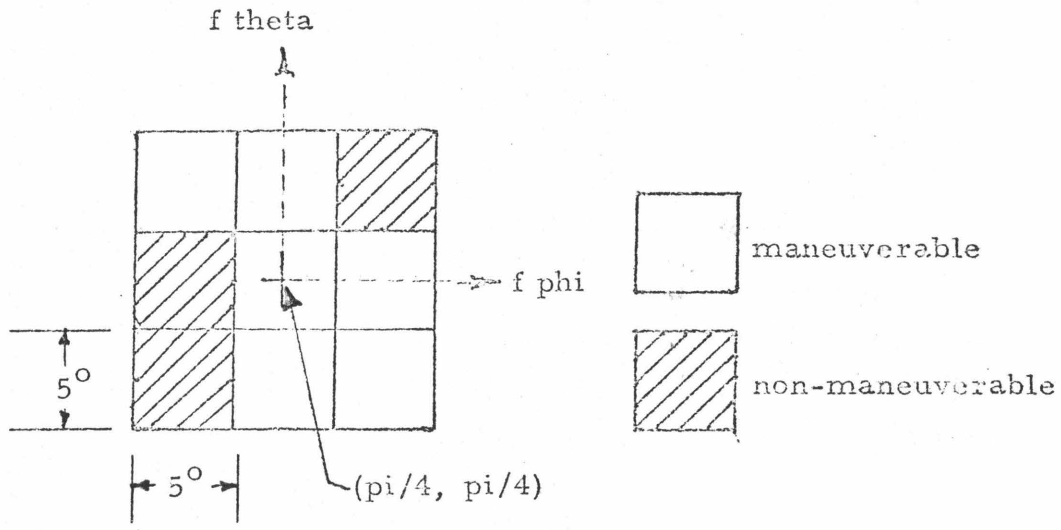
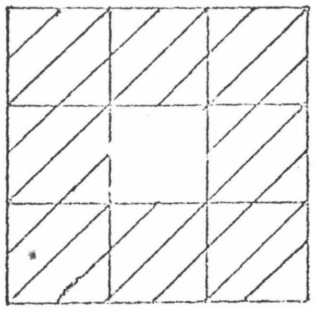
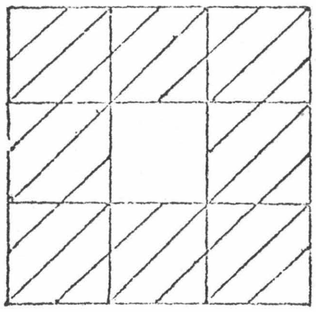
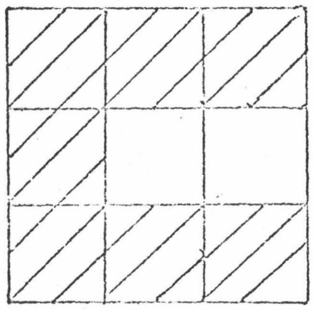
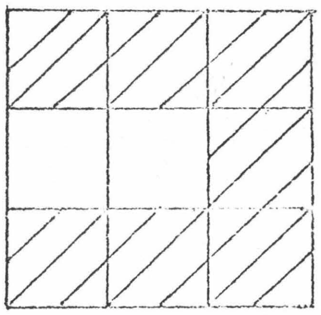


Figure 9.2 S-area


(a) Crater


(b) Arch


(c) Right Overhang


(d) Left Overhang

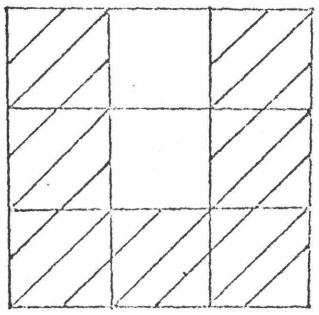

(e) 3-D Channel

Figure 9.3 Obstacle Configuration Types

# CHAPTER 10
# SYSTEM DETAILS

## 10.1 WHY TWO SYSTEMS?

My goal was to solve the collision detection and avoidance problem for the 3D manipulator. The difficulty of visualizing 3D objects added to the complexity of the problem. The natural thing to do was to consider a two-dimensional analogue of the problem and then hope to generalize it to three dimensions. Of course such generalizations do not always work and since the 2D system was very large (approx. 125 pages of source programs) it was indeed a big gamble that I took.

Attempting to solve the 2D problem was extremely helpful. Operating in two dimensions made it easy to visualize the different problem spaces and transformations between them. Again, it is easier to try out ideas in two dimensions than in three dimensions. This relative ease of experimentation led me to implement the fixed and variable grid models of free space, and the experience so gained enabled me to hit upon the dynamic chart model, the most crucial component of my solution. Finally, having carried the solution to completion for the 2D problem, I had identified all the necessary components and their interactions for a solution in the 3D case.

I had expected the generalization to three dimensions to be quite straightforward. I was wrong on three aspects and was right on one aspect of the generalization. The four parts that needed generalization were

1) obstacle descriptions and definitions of transformation on obstacles

2) chart and region structures

3) boom planning and nature of the boom tip locus

4) forearm planning

In the first three cases the 3D solution is far more complex than the 2D one. Only in the fourth case is the 3D solution comparable in complexity to the 2D problem. I will not go into the details of what is a natural extension of the 2D solution and what its problems are. These details have been presented in sections 4.4, 5.7 and 8.4, each titled "Extending 2D Ideas". Section 8.4 discusses both boom and forearm planning.

In conclusion I can only say that the task of finding a solution to the 3D problem would have been extremely difficult, if not impossible, had I not solved the 2D problem. I make this claim even after having said that in seventy-five percent of the cases, a direct generalization of the 2D solution to the 3D had not worked. I do so because the solution to the 2D problem gave me considerable insight into the collision detection and avoidance problem.

## 10.2 IMPLEMENTATION DETAILS

The solution presented in the preceding chapters has been verified by designing and implementing computer programs. I would have liked to attach listings of the programs along with this report. The large size of these programs makes it difficult, if not impossible, to do so. Almost all the programs were written in SAIL [VanLehn (1973)] and run on a DEC PDP10 computer. The PDP10 has a KA10 processor. The latest processor for the same computer operates about six times faster than the KA10. This should be borne in mind when looking at the performance figures for the implementation. Below are described a few details of the final versions of the 2D and 3D systems.

### 10.2.1 2D System

The 2D system consists of about 125 pages or approximately 5000 lines of SAIL code. The system is organized into modules as follows

Representation

1. Environment - Decompose, enlarge and dilate routines
2. Free space - generate chart and region and refine chart

Planning

1. Boom Planning
2. Forearm Planning
3. 2Dplan - Planning executive

Utility Routines

1. Input / output - Environment and free space models need not be generated every time; once computed they are saved on disk and read whenever needed
2. Plotting routines for the Tetronix and HP-plotter for visualizing obstacle transformations and navspace models. These were based on some routines written by Scott Roth.

Executive

Armplan - program that coordinated all the above modules

Core requirements for the entire system are 40K or more depending on the number of obstacles the environment contains. The problem shown in Figure 1.1 is solved in 3 seconds of CPU time. This timing is for unoptimized code (array bound checking suppression etc. is not done). Generate-chart and refine-chart routines all take less than a second of CPU time. The output of the collision detection and avoidance system has yet to be interfaced with a real manipulator.

### 10.2.2 3D System

Due to core limitations of 56K per job, the 3D system had to be organized as a multi-pass system. The passes and the length of the SAIL source text in pages are as follows

Environment Data Processing (40 pages)

Navspace Model Generation (35 pages)

Boom Planning - Mid-section (50 pages)

Forearm Planning - Mid-section (30 pages)

Terminal phase planning (30 pages)

Trajectory Calculation (10 pages)

The trajectory calculation routines have been implemented before [Lewis (1974)]. I have implemented the first three passes. The fourth and fifth passes have been designed to a very detailed level but the implementation not completed.

The core requirement for each pass is well below the 56K limit. Execution times for the passes are in seconds of CPU time and I think it should be possible to plan complete trajectories within 10 seconds. Given a KL10 processor on the same PDP10 computer, this means that trajectories can be planned in a few seconds.

## 10.3 A CRITICAL REVIEW OF THE CURRENT IMPLEMENTATION

### 10.3.1 Size of software

The core requirement for the 3D trajectory planning system that does collision detection and avoidance is very high, especially, if one wants to build a minicomputer system controlling a manipulator. There are two comments I would like to make about this.

1) The system handles a large number of special situations which seldom arise in practice. A significant reduction in size is possible if one is willing to give up "completeness" of solution and eliminate many of the special checks.

2) One of my goals was to make the planning times comparable to the execution times. With this in mind any piece of data that was used more than a few times was computed and saved; the saving in time is paid for in terms of storage. A careful look at the programs might indicate where some of the storage can be reduced at a small price in execution time.

## 10.3.2 Subgoal Characterization

Whenever a collision is detected subgoals are introduced. The subgoal is always a configuration. Complete specification of a configuration is very often too much commitment. It would be desirable to have the planning system capable of planning with a set of possible configurations and delay deciding on the specific configuration until further downstream.

A significant portion of the blame for the current characterization of subgoals lies in the use of the linear approximation algorithm for trajectory hypothesis and trajectory modification (see Figures 8.1-8.2). The ideas behind the dynamic chart model that permit very simple manipulator descriptions are valid independent of the basis for hypothesizing and modifying trajectories; they admit a single point description of the boom and consequently make boom collision detection almost trivial. So it should be possible to find different underlying strategies for hypothesizing and modifying trajectories that will permit better subgoal characterization.

### 10.3.3 Handling Large Objects

The current versions of the implementation assume that the manipulator only transports objects smaller than its hand. Incorporating the capability of transporting larger objects is conceptually not difficult. The following points need to be considered for handling large objects

1) Because of the selective refinement capability it is easy to make incremental changes to the charts. These changes are necessitated by the changes in the environment caused by the moving of obstacles.

2) The object to be transported is first enclosed by a minimum bounding parallelopiped. A contraction transformation, the inverse of the enlargement transformation, is applied to the parallelopiped model. All collision detection and avoidance in the primary problem space is carried out using the smaller parallelopiped representation resulting from the contraction transformation. Contraction applied to objects being transported around, nullifies the enlargement done on obstacle descriptions.

3) Primary problem space representations alone, and not secondary problem space representations, should be used for planning when the hand is carrying an object larger than its hand.

4) Boom planning is carried out as before but forearm

planning needs to be augmented by collision checks for the volume generated by the object in the hand. The types of motions the forearm executes are the same as before. The volume of space traced out by the object in the hand due to the motion of the manipulator is a polyhedron. Collision detection would now require determining intersections of polyhedra. Since the size of objects will vary there is no hope of finding an inexpensive one-time only transformation which will simplify this task.

# CHAPTER 11
# CONCLUSIONS

This thesis presented a solution to the safe trajectory planning problem for mechanical manipulators. It discussed computer implementations of the solution for 2D and 3D manipulators with sliding joints. Section 1 of this chapter recapitulates the problem, section 2 presents the key ideas in the solution, and section 3 suggests directions for future work in this field.

## 11.1 SAFE TRAJECTORY PLANNING

We are interested in the safe trajectory planning problem for computer controlled manipulators with two links and multiple degrees of freedom. The system is given a complete description of the part of the environment in which the manipulator is to maneuver. The input is the goal position and orientation of the hand. The output is the trajectory locus, specified as a sequence of typed intermediate configurations, the type indicating the nature of the subsequent section of the trajectory. Trajectory calculation routines compute the trajectory from the trajectory locus generated by the planning programs. The executive system in charge of running the hardware uses the trajectory for servoing the joints on the manipulator.

The solution presented permits the manipulator to transport objects which can be enclosed within the minimum bounding cylinder approximating the manipulator link. Modifications to the solution to handle large objects were presented in section 10.3.3. Extensions permitting dealing with manipulators having only rotary joints(see Figure 11.1) are indicated in section 11.3.4.

## 11.2 KEY IDEAS IN THE SOLUTION

This section briefly summarizes the key ideas in the solution. Chapter 2 discussed the theoretical framework which tied together the points that will be discussed here. Chapters 3-9 provided more details on them.

(1) Simplified Manipulator Descriptions and Trajectory Primitives A simple and direct model for the manipulator is that of two connected cylinders, one representing the boom and the other the forearm. This dissertation identifies alternative problem spaces of increasing abstraction that permit simplified manipulator descriptions. The manipulator can be modelled as two line segments, a single line segment or unbelievably as a point! This thesis identifies primitive trajectory types. These primitives along with the simplified manipulator descriptions make collision detection, and trajectory hypothesis and modification

numerically tractable.

(2) Navspace and Charts Navspace is the single most important concept that reduces the complexity of the safe trajectory planning task. It permits the manipulator to be considered as consisting of just the forearm or as just a single point - the boom tip. Since navspace comes in odd shapes it is hard to characterize, but this dissertation provides ways of characterizing and using navspace. Some of the important ideas are

a) Navspace is approximated by easily describable entities called charts.

b) The approximation is dynamic and is under program control.

c) The approximation can be selective, and thus it is easy to make incremental modifications to the charts.

d) The concepts of navspace and its approximation by charts are independent of any planning strategies.

3) Transformations with Minimality Property The use of multiple representation spaces, for reducing the complexity of the trajectory planning problem, is made more effective by the use of transformations that satisfy the minimality property (see section 2.2.2). The enlargement, dilation and survey transformations satisfy the minimality property. If this had not been so the advantage gained by using the

alternative problem spaces would have been offset by the expensive computations required to generate them.

(4) <u>Trajectory Planning in Empty Space and Collision Avoidance</u> There are two ways to look at safe trajectory planning. The first concerns itself with planning trajectories in empty space; obstacles enter into consideration only indirectly in that they determine what part of the maneuverable space is free. The second considers obstacles alone; free space considerations are of secondary importance. This thesis shows how these complementary views can be used to advantage in the safe trajectory planning problem. Specifically, the boom planning problem is treated as planning trajectories in empty space and forearm planning is treated as a collision avoidance problem.

(5) <u>Cartesian Space and Joint Space</u> Obstacles are naturally described in cartesian space and trajectories in joint space. If obstacles and trajectories are both represented in one space, collision checks would not require the constant and expensive conversion between the two spaces. This thesis shows how it is possible to get the best of both cartesian space and joint space representations, and yet avoid the constant conversion overhead problem. The trick lies in decomposing the

planning task into boom and forearm planning, and the maneuverable space into navspace and obstacles.

(6) Planning This thesis shows how the principles of hierarchial decomposition can be used to reduce the complexity of the trajectory planning problem. Different planning strategies are used for maneuvering far away from obstacles and for maneuvering close to obstacles. A formal characterization of large chunks of empty space makes maneuvering far away from obstacles very easy. A good characterization of obstacle configuration types simplifies planning of maneuvers close to obstacles.

(7) Planning at Execution Time A consequence of the investigations into the collision detection and avoidance problem has been the identification of execution-time strategies for terminal phase motion. Guidelines have been presented for incorporating proximity sensors into the manipulator system (see section 11.3.1).

## 11.3 SUGGESTIONS FOR FUTURE WORK

The characteristics of a general purpose manipulator system were presented in section 1.1. Section 1.1 indicated that no such general system exists and that research on various aspects of such systems is being done. I had indicated that considerable progress has been made on the problem of executing trajectories on real manipulators. Now the safe trajectory planning problem is solved. This section describes the next few steps that need to be taken in the march towards the design of a general purpose manipulator system.

## 11.3.1 Planning during Execution

I had indicated in section 1.1 that very little is known about modifying trajectories dynamically based on any sensory data that the system may acquire during execution. The discussion on terminal phase planning indicates how one might proceed in analyzing this aspect of manipulator systems. As a first try, proximity sensors could be put on the forearm and used in terminal phase planning. Later, one could experiment with force and tactile sensors and even visual feedback. The following paragraphs describe why it would be useful to have proximity sensors on the manipulator and how one might use them.

We know that boom planning is simpler than forearm planning. The simplicity arises from the use of charts boom planning. Within a chart the boom can be considered as a single point and so boom planning reduces to path planning for a single point. The notion of charts is not useful for forearm planning because the charts change as the boom tip changes its position. Without some equivalent of a chart, the simplest description of the forearm will always be a line in 3-space and never a point. Collision checking with a line segment being more expensive than collision checking with a point, forearm planning will always be the more expensive component of the trajectory planning problem.

Since there is no solution in software that will make forearm planning as simple as boom planning, we might look to see whether additional hardware support might help. Proximity sensors, mentioned in Chapter 9, are a possibility. They could be incorporated into the system as follows. We have the manipulator come out from the terminal position such that the boom tip is located at a safe point (for definition see section 2.3.6). This would be done during execution of the trajectory. The system then plans a safe trajectory from the safe point near the start to a safe point near the goal. This safe trajectory is executed. The hardware terminal phase controller then takes over to get the manipulator into the desired goal configuration. The hardware terminal phase controller implements the adjust and move motions described in Chapter 9.

The proximity sensors suggested in Chapter 9 are different from the ones investigated by Bejczy and Johnston (1974) in that I require the sensitive volume to be spread over a much larger solid angle.

### 11.3.2 Nature of Constraints

The characteristics of a general purpose programmable system presented in section 1.1 included a formal language for describing computational processes related to the manipulator. Using this language a user could specify how objects are to be manipulated and how the manipulator should maneuver around obstacles. Since the system has extensive internal models of the universe of discourse the user does not have to specify all his requirements explicitly. In particular the user need not worry about collision problems. In this dissertation we have solved the collision avoidance problem. The obvious next step is to incorporate this collision avoider into a larger system which analyzes more general constraints.

### 11.3.3 Multiple Manipulators

Considerable work on the use of two manipulators for assembly tasks has been done at the Stanford Artificial Intelligence Laboratory [Finkel et al (1974)]. The solution to the safe trajectory planning problem presented in this dissertation considered the manipulator to be the sole active agent in the environment. An interesting problem to tackle is one where two active manipulators are in the environment. Instead of checking for collisions with obstacles alone, collision with the trajectory envelope of the second manipulator also needs to be checked. Conceptually, now that we know how to handle one manipulator, handling two manipulators is easy. The complexity of any specific implementation of a system analyzing two or more manipulator trajectories will depend on the complexity of the interactions that are permitted between the manipulators.

### 11.3.4 Anthropomorphic Manipulators

For the purposes of this section let A-manipulator denote an anthropomorphic manipulator, and M-manipulator denote a mechanical manipulator. An example of an A-manipulator is shown in Figure 11.1(*). The last link is called the forearm, and is identical to the forearm on the M-manipulator. The first two links are called r-boom and f-boom. They accomplish the same purpose as the sliding joint of the M-manipulator. R-boom and f-boom have a hinge

--------------

* The original figure is from Winston(1974). I have added a coordinate frame and names of links and joints to the figure. The additions are obvious from the fonts.

joint, called psi, connecting the two. The boom tip is the front end of the f-boom link. The links r-boom and f-boom are of fixed length. Consequently, the r value of the boom tip, in the coordinate frame shown in the figure, has a one-to-one correspondence with the angle psi between the two links. The correspondence is expressed by

r = r-boom * cos(psi / 2) + f-boom * cos(psi / 2)

where r-boom and f-boom denote the lengths of the links r-boom and f-boom respectively. This observation is very crucial to the extension of the solution to the safe trajectory planning problem.

As for the M-manipulator, we can identify the hierarchy of problem spaces - real, primary and secondary - which permit simpler and simpler manipulator models. We use environmental models and trajectory primitives that are identical to the ones used by the M-manipulator solution. We have very similar free space models. We define A-navspace to be the set of all boom tip positions of the A-manipulator, for which both the r-boom and f-boom are free from collisions. Note that the definition of A-navspace is very similar to the definition of navspace, the difference is that the boom of the M-manipulator is replaced by the first two links of the A-manipulator. Again, since A-navspace comes in odd shapes, we approximate it by easily describable entities called A-charts. A-charts are made up of A-regions which are identical to the regions in the M-manipulator case.

Two complications arise in the case of the A-manipulator which prevent the direct application of the earlier solution. They are

1) In the case of the M-manipulator, each region has associated with it a single (rmin, rmax) pair of numbers designating the safe r-limits of the boom extension. In case of A-regions, we can have more than one of these. Thus there are different "pockets" of safe maneuverable volumes for the same solid angle, and these pockets may be inaccessible from one another.

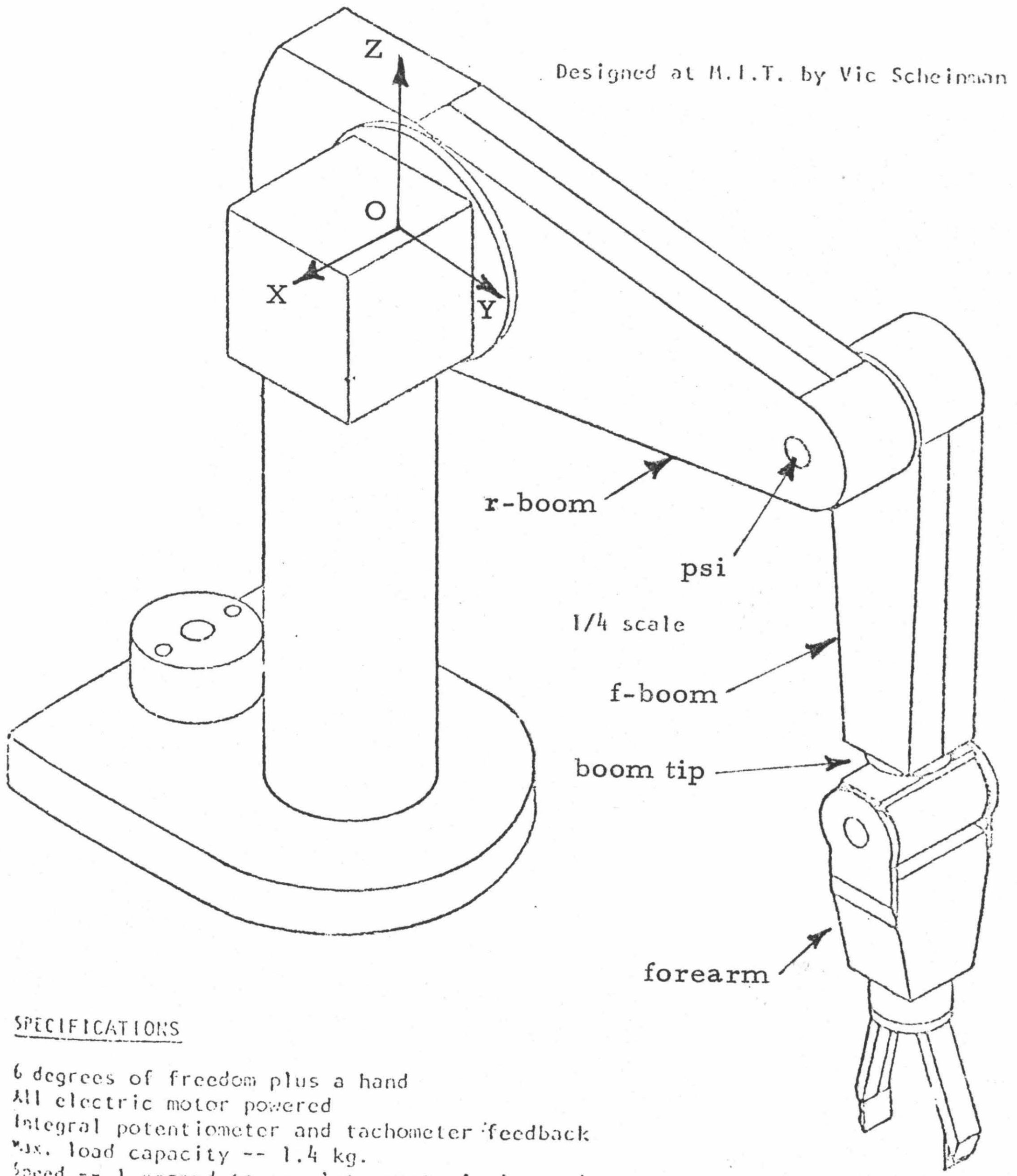
2) The determination of rmin and rmax is not as straightforward as it is in the case of the M-manipulator.

However, it is possible to compute the A-chart representation of A-navspace. Within the A-charts the three-link manipulator reduces to a single link. If A-charts of the secondary problem space are used, the entire manipulator is reduced to a single point. Planning may now proceed as in the case of the M-manipulator. The concepts of terminal phase planning and mid-section phase planning are valid for the A-manipulator too. The equivalent of boom planning is, however, more complex because of (1) above. Finally, since there are twice as many solutions to the position problem for the A-manipulator as for the M-manipulator, the planning programs have more alternatives to exhaust before they announce failure.

In short, we will have a solution to the safe trajectory planning problem for A-manipulators if, in addition to the solution developed for the M-manipulator, we have

1) inexpensive schemes for computing rmin and rmax, for the components of the A-charts, and

2) an algorithm for point path planning within A-charts.



SPECIFICATIONS

6 degrees of freedom plus a hand
All electric motor powered
Integral potentiometer and tachometer feedback
Max. load capacity -- 1.4 kg.
Speed -- 1 second to complete most simple motions
Resolution -- 1.5 mm.
Accuracy -- 3mm.
Workspace -- 20 cm. radius hemisphere

FIGURE 11.1

## REFERENCES

(1) Bejczy, A. K., "Machine Intelligence for Autonomous Manipulation", Proceedings of the First National Conference on Remotely Manned Systems, California Institute of Technology, Sep. 1972.

(2) Bejczy, A. K. and Johnston, A. R., "New Techniques for Terminal Phase Control of Manipulator Motion", JPL Technical Report 760-98, Feb. 1974.

(3) Brand, L., "Vector Analysis", John Wiley and Sons Inc., 1957.

(4) Dobrotin, B. M. and Scheinman, V. D., "Design of a Computer Controlled Manipulator for Robot Research", Third International Joint Conference on Artificial Intelligence, Stanford University, Aug. 1973.

(5) Finkel, R. et al., "AL, A Programming System for Automation", Stanford A.I. Memo 243, Nov. 1974.

(6) Gill, A., "Visual Feedback and Related Problems in Computer Controlled Hand Eye Coordination", Stanford A.I. Memo 178, Aug. 1972.

(7) Lewis, R.A., "Autonomous Manipulation in a Robot: Summary of Manipulator Software Functions", JPL Technical Report 33-679, Mar. 1974.

(8) Lewis, R. A. and Bejczy, A. K., "Planning Considerations for a Roving Robot with Arm", Third International Joint Conference on Artificail Intelligence, Stanford University, Aug. 1973.

(9) Newman, W. M. and Sproull, R. F., "Principles of Interactive Computer Graphics", McGraw Hill Inc., 1973.

(10) Olesten, N. O., "Numerical Control", John Wiley and Sons Inc., 1970.

(11) Paul, R., "Modelling, Trajectory Calculation and Servoing of a Computer Controlled Arm", Stanford A.I. Memo 177, Nov. 1972.

(12) Pieper, D. L., "The Kinematics of Manipulators Under Computer Control", Stanford A.I. Memo 72, Oct. 1968.

(13) Scheinman, V. D., "Design of a Computer Controlled Manipulator", Stanford A.I. Memo 92, Jun. 1969.

(14) Shoenfield, J. R., "Mathematical Logic", Addison-Wesley, 1967.

(15) VanLehn, K. A., "SAIL User Manual", Stanford A.I. Memo 204, Jul. 1973.

(16) Weinstein, M., "A Framework for Robotic Design", Caltech Information Science TR-14, 1975.

(17) Weinstein, M., "Structured Robotics", Caltech Information Science TR-15, 1975.

(18) Whitney, W. M., "Hand-Eye System Design Book - Functional Requirements", Robotics Research Program, JPL, Oct. 1974.

(19) Widdoes, C., "A Heuristic Collision Avoider for the Stanford Robot Arm", Stanford C.S. Memo 227, Jun. 1974.

(20) Winston, P. H., "New Progress in Artificial Intelligence", MIT AI-TR-310, 1974.

# APPENDIX 1
# THE JPL ROBOT

The JPL robot research program aims at applying the methods of Artificial Intelligence and Robotics to the design of machines for planetary surface exploration. The machines are to perform without step-by-step human control on missions in which there is long round-trip communication time, limited communication channel capacity and conditions that may be largely unknown or unpredictable in detail [Whitney et al (1974)]. This requires that the machines be able to integrate sensory and motor functions in the autonomous performance of activities in response to high level commands issued by a human. The immediate goals of the project are to design and implement an integrated system that has vision (TV, laser etc.), manipulation and locomotion capabilities.

The breadboard robot hardware (Figure A1.1) consists of a mobile vehicle, a six degree-of-freedom manipulator (the Scheinman arm), two vidicon TV cameras for stereo vision, a GaAs pulsed laser range finder, navigation and guidance sensors, tactile and proximity sensors, and a local minicomputer connected to remote computers, graphic displays, and operator consoles.

The manipulator is a modified version of the arm used at the Stanford A.I. Laboratory [Dobrotin and Scheinman(1973), Scheinman(1969)]. It has six degrees of freedom, allowing any desired hand position and orientation in an open or slightly obscured workspace. The maneuverable space is within a radius of 1.30 meters measured from the center of the base of the manipulator. The six joints connecting the links from the base to the hand are in the following sequence (see Figures 3.4-3.5) : two rotary joints (providing shoulder azimuth and elevation action), a linear joint (providing in and out reach action), and three rotary joints (providing the wrist action) [Lewis and Bejczy (1973)]. The hand is presently a simple parallel jaw mechanism. The joints are driven by permanent magnet DC torque motors geared directly to the corresponding links. Depending on the relative position of the links, the arm can handle loads of up to 5-8 pounds Earth weight. The arm servo control utilizes analog position measurements from the joint outputs and analog velocity measurements from the motor shafts. Holding torque at each joint is provided by electromagnetic brakes. The arm's structural stiffness and tight servo control can provide accuracy within a few tenths of an inch.

The software system architecture is hierarchial with the robot executive (REX) controlling and monitoring the various software subsystems. The various software subsystems are largely independent of each other. The issues related to the design of integrated robotic systems are very complex and are discussed elsewhere [Weinstein (1975)].

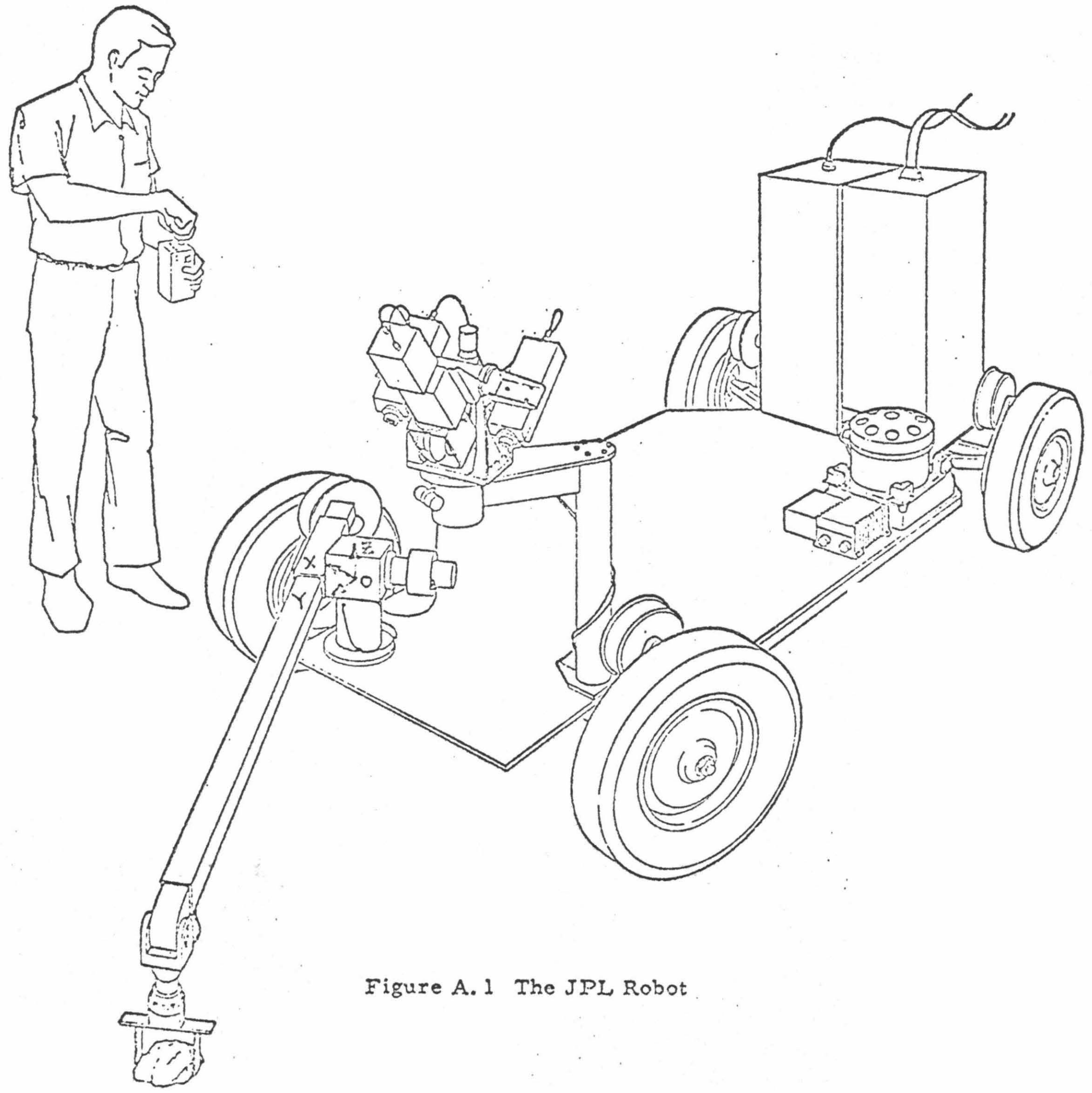


Figure A.1 The JPL Robot

# APPENDIX 2

## CIRCULAR ORDERING

Let x be such that -pi leq x leq +pi and x + 2 * pi = x and x - 2 * pi = x. Let y satisfy relations similar to x. Then the following function written in SAIL defines a predicate grt(x, y). Intuitively, the predicate is true if y can be brought to coincide with x by rotating it anticlockwise and by less than pi radians.

```
SIMPLE BOOLEAN PROCEDURE GRT(REAL X, Y);
  IF (X GEQ 0 AND Y GEQ 0) OR (X LEQ 0 AND Y LEQ 0) THEN
    RETURN(X > Y)
  ELSE IF X GEQ 0 THEN RETURN(IF X = Y + PI THEN FALSE
                                           ELSE Y > X - PI)
    ELSE BEGIN
      X SWAP Y;
      RETURN(IF X = Y + PI THEN FALSE
              ELSE Y < X - PI)
      END;
```

where geq is greater than or equal, leq is less than or equal.

## APPENDIX 3

## PROGRAM LISTINGS

The program listings of the 2D and 3D systems are available on microfilm with

Len Friedman,

Robotics Research Program,

114-122 J. P. L.,

Pasadena, Ca 91103.