CSE-87-00005

TOWARDS BUILDING A TEAM OF INTELLIGENT ROBOTS (126-NAG-1-772-JFR) FINAL REPORT

Dr. M.R. Varanasi (PI)
Dr. R. Mehrotra (Co/PI)

TOWARDS BUILDING A TEAM OF INTELLIGENT ROBOTS

(126-NAG-1-772-JFR)

FINAL REPORT

1. *Contributing Faculty*

1. Dr. M.R. Varanasi (PI)
2. Dr. R. Mehrotra (Co/PI)

2. *Contributing Students*

1. Mr. R.A. Basta
2. Mr. J.P. Sowers
3. Mr. F.K. Kung

3. *Technical Aspects*

In this project, the research and development activities were mainly focused on the following topics:

a) Collision-Free Motion Planning of Multiple Robot Arms:

As a step towards solving the problem of planning collision-free motion of multiple robot arms in a common workspace, the two robot arms case was investigated. An efficient approach to planning collision-free motion of two robot arms in a common workspace was developed. The approach involves detection of collisions along the path and then modifying the path and/or motion characteristics of one or both of the robots to avoid the detected collisions. This technique is currently being extended to handle the

case of multiple arms motion planning problem. The details of the methods are reported in reports/papers in Appendix-1.

b) Object Recognition

Two new model-based approaches to 2-dimensional object recognition were developed.

One of these approaches involves representing a 2-D objects as an ordered set of meaningful components. Each component is described by a feature vector. To recognize an unknown object, a component of a given model is matched against the components of the unknown object representation. If a good match is found, the identity and the location of the object is hypothesized and verified.

The second approach is based on a data-driven hypotheses generartion scheme called SMITH (Shape Matching Utilizing Indexed Hypotheses Generartion and Testing) for 2-D model-based object recognition. It employs an efficient dynamic programming implementation of attributed string matching to compare a scene component with a model component.

The reports/papers given in Appendix-2 can be refered to for further details.

c) Pictorial Database

A database which enables users to store and share the representations of 3-dimensional objects was designed and

implemented. The database design is based on a relational feature based object representation scheme. This scheme is information preserving and efficient in terms of storage space and retrieval time requirements. A detailed report on this work is given in Appendix-3.

4. PUBLICATIONS

The results of research work on the above mentioned topics were reported in the following publications:

1. R.A. Basta, R. Mehrotra, and M.R. Varanasi, "Detecting and Avoiding Collisions Between Two Robot Arms in a Common Workspace," *IEE International Workshop on Robot Control: Theory and Applications*, Oxford, England, April 1988. REMOVED

2. R.A. Basta, R. Mehrotra, and M.R. Varanasi, "Collision Detection For Planning Collision-Free Motion Of Two Robot Arms," Submitted.

3. R.A. Basta, "Collision-Free Motion of Two Robot Arms In A Common Workspace," M.S. Thesis, Computer Science and Engineering Department, University of South Florida, Tampa, FL, November 1987.

4. R. Mehrotra, W.I. Grosky, and F.K. Kung, "Recognizing Two-Dimensional Objects", *IEEE International Conference on Systems Man Cybernetics,* Alexendria, VA, October 1987. REMOVED

5. R. Mehrotra and W.I. Grosky, "Shape Matching Utilizing Indexed Hypotheses Generation and Testing," *IEEE Journal of Robotics and Automation,* To appear. Removed

6. J.P. Sowers, "A Rudimentary Database For Three - Dimensional Objects Using Structural Representation," Technical Report, Computer Science and Engineering Department, University of South Florida, Tampa, FL.

S1-37

111677

P 67

APPENDIX-1

APP

CSE-87-00002

COLLISION-FREE MOTION OF TWO ROBOT ARMS IN A COMMON WORKSPACE

Robert A. Basta Rajiv Mehrotra
Murali R. Varanasi

TABLE OF CONTENTS

LIST OF FIGURES

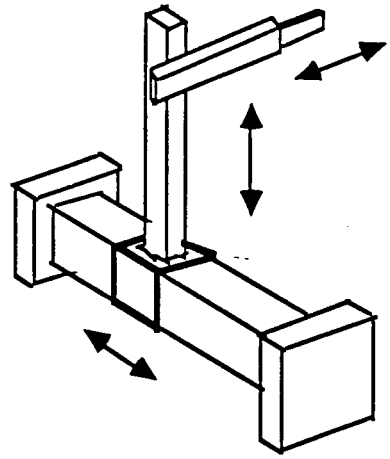
ABSTRACT

Collision-free motion of two robot arms in a common workspace is investigated in this report. A collision-free motion is obtained by detecting collisions along the preplanned trajectories using a sphere model for the wrist of each robot and then modifying the paths and/or trajectories of one or both robots to avoid the collision. Detecting and avoiding collisions are based on the premise that: 1) preplanned trajectories of the robots follow a straight line, 2) collisions are restricted to be between the wrists of the two robots (which correspond to the upper three links of PUMA manipulators), and 3) collisions never occur between the beginning points or end points on the straight line paths. In this report, the collision detection algorithm is described and some approaches to collision avoidance are discussed.

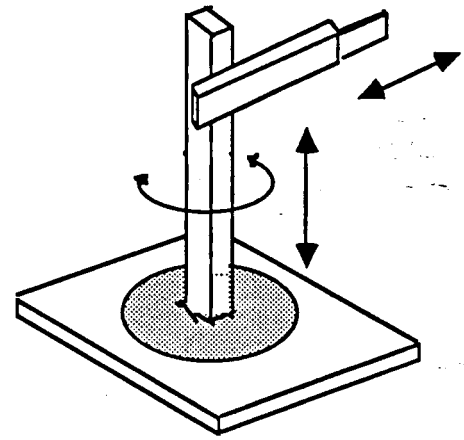
1. INTRODUCTION

Industrial robots have made significant contributions in automating the manufacturing process. They are general purpose manipulators consisting of a series of rigid links connected together by revolute or prismatic joints. Mechanically, a typical robot has a supporting base, an arm unit, a wrist unit, and an end effector or tool. Movement of the arm unit usually consists of three degrees of freedom in which a sequence of movements can position the wrist unit at some desired location in the workspace. The wrist unit, typically consisting of two or three rotary joints, orients the end effector in such a manner to perform the tool task. The wrist unit usually provides a mounting plate so that various types of end effectors, such as grippers, welding guns, or electro-magnets, can be attached. Typical robot configurations are shown in Fig. 1 and Fig. 2.

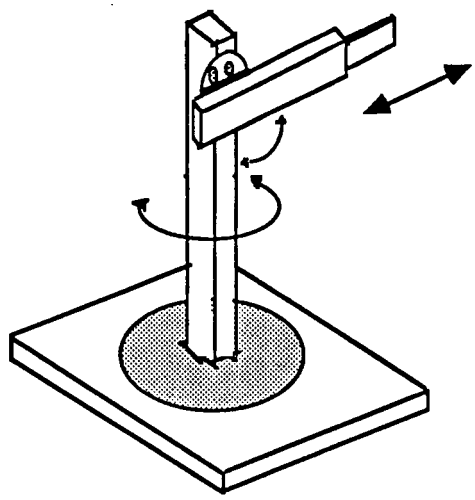
Presently, the number of robots being used in industrial and commercial applications is increasing at a rate of about 35 percent per year [1]. Manufacturers find that robots can increase productivity, reduce production costs, and improve product quality. However, the robots currently in use perform simple repetitive jobs such as pick-and-place tasks, machine loading and unloading, spray painting, and spot welding. Recent advances in such technologies as robot sensors and vision systems will allow more complex tasks to be performed. The development of intelligent robots is essential for



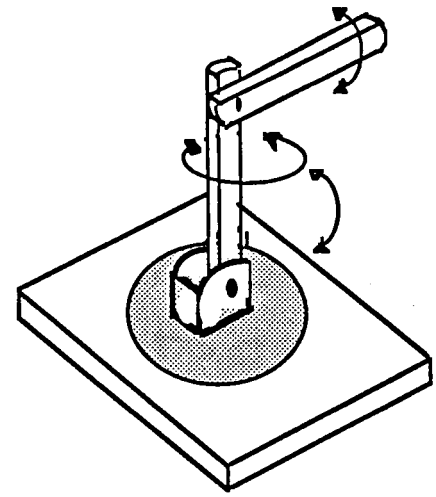
Cartesian Coordinates



Cylindrical Coordinates



Spherical Coordinates



Revolute Coordinates

Figure 1 Configurations of Robots

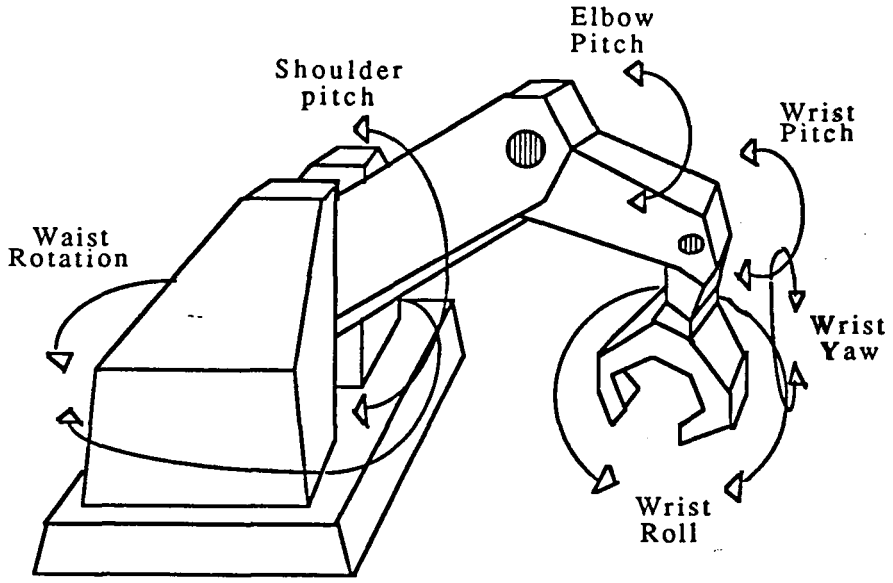


Figure 2 Six Degree Freedom Revolute Robot

increasing the industrial and commercial uses of robots [2], [3].

In a fully automated manufacturing environment, sophisticated robot systems should be able to handle nearly all manufacturing operations. In such an environment, robots must work together and perform their tasks in a coordinated manner to fully utilize the arms and the workspace. However, such robots could become obstacles to each other without proper strategy and, therefore, require collision-free paths between their end effectors. Techniques for controlling robot arms in a common workspace will demand trajectory planning schemes, collision detection algorithms, and collision avoidance strategies. These demands motivate the research that constitutes this report.

1.1. Multiple Robot Arms

Robots in the future are likely to possess human-like dexterity.

Robot arms will probably perform sophisticated tasks in the same manner as do human arms and hands. Presently, research in multiple arm coordination and control is just beginning. The performance of simple tasks, such as lifting an object by two robots or providing collision-free motions, is still difficult.

Since utilizing only one robot to operate in a workspace limits the amount of tasks that can be performed, the use of two or more robots is essential to improve the versatility of potential applications. To complete a task, several robots can be used with each performing its specific small subtask which allows for an increase in productivity. Along with an increase in productivity, multiple arms can perform complex tasks, such as lifting objects that are beyond the weight limits of a single arm and assembling sophisticated equipment, that cannot be performed by a single robot but require the use of two or more working together. In addition, certain applications require coordination between robots to save time in completing a task.

Roach [4] classifies coordinated actions of two robot arms into four categories:

1) Symbiotic Actions - A class of actions where one hand aids the other in a passive way. For example, a nail must be held upright while hammering it. Holding a tooth brush while applying tooth paste is another example.
2) Compliant Actions - A class of actions involving hands performing similar movements at different places, usually on the same object. For instance, two hands lifting a pan of water must move together to prevent spillover.

3) Co-operative Actions - A class of actions where hands are actively performing actions that are not the same but require coordination. Tying shoelaces is an example.

4) Countervailing Actions - A class of actions where hands assist each other by performing actions that are apparently opposed. For instance, glueing two pieces of tile together is performed by applying pressure to each one.

Cooperating robots in a common workspace must be coordinated in order to avoid collisions between them. The research reported in this paper is directed at coordination and control of two robots in a common workspace through collision-free motion planning.

1.2. Problem Formulation

The motivation for studying the problem of using multiple arms and controlling robots in a common workspace should now be clear. This investigation, without the loss of generality, deals with the case of only two robots. When two or more robots are used in a common workspace, they may become obstacles to each other and, therefore, motion planning must include detection and avoidance of collisions between them. A collision-free motion is obtained by detecting collisions along the straight line trajectories of the robots and then replanning the paths and/or trajectories of one or both of the robots to avoid the collision.

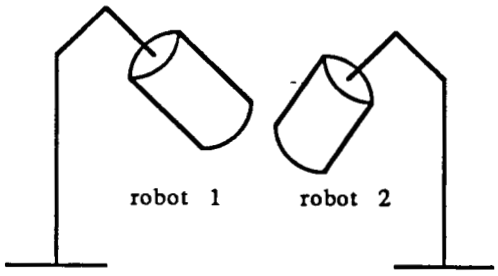
The straight line path and trajectory information of each robot is used to detect whether collisions exist. Collisions are restricted to be between the wrists of the two robots (which correspond to the upper three links of PUMA manipulators). A sphere model for the

wrist (including the tool and any grasped object) is used because it is rotationally invariant and computationally efficient compared to other geometric models such as the cone and cylinder models (Fig. 3). Collisions are assumed never to occur between the beginning points or end points on the straight line paths. This problem is handled by a higher level planner which guarantees that the operations of the robots are valid, and, therefore, they never attempt to access a specific location in the workspace at the same time.

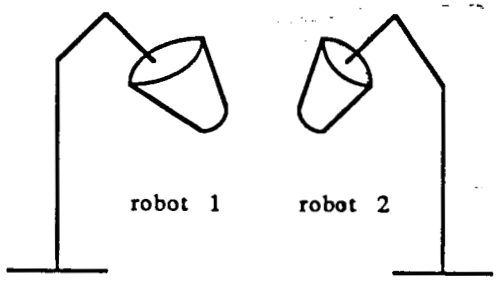
The collision detection method involves two steps: 1) obtaining the range of potential collisions along the straight line trajectories of the two arms without considering the motion characteristics, and 2) mapping the potential collision range information into the time domain to obtain the space-time collisions. Once the collision region in time and space is found, a collision-free motion is obtained by producing new paths and/or trajectories for the robots based on various collision avoidance techniques.

1.3. Background

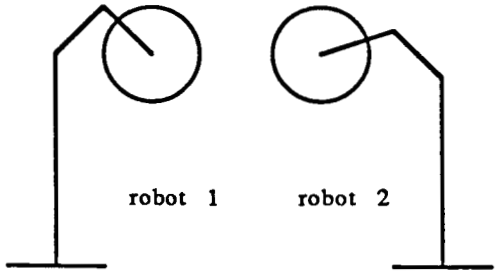
Methods for coordinating multiple arms that are currently being investigated by several researchers range from using low-level kinematics and dynamics to developing high-level motion planners. Various approaches in dual-arm control consider master/slave relationships between the robots using position and/or force feedback. Ishida's [5] force control method is based on a two arm transport system run in a master/slave mode. In this method, the fundamental movements are parallel transfer and rotational transfer, but more complex motions can be accomplished if



Cylindrical Model



Truncated Cone Model



Sphere Model

Figure 3 Commonly Used Wrist Models

combined. A wrist force sensor measures the interactive force between the two arms. The master arm is position controlled while the slave arm is entirely force controlled and free to move where necessary to follow the master. Alford and Belyen [6] use a hierarchical computer control structure to perform a type of master/slave coordination. The master is position controlled for a desired trajectory and the slave follows relative to the master's path. The slave arm's trajectory is modified in real time based only on the position of the master and not on any force/torque sensor information. Tarn, Bejczy, and Yun [7] have developed a control method to transfer an object along a desired trajectory using two robots. The robots work in a master/slave mode in which the slave robot follows the master by keeping a constant offset distance. Their control method uses a dynamic coordinator acting on relative position and velocity errors and/or on relative force/torque errors between the two arms. Zheng and Sias [8] use wrist force sensors to detect contact between two robot arms. The force information is used to adjust the relative positions and orientations of the arms to continue their assembly task. Hayati [9] uses hybrid position/force control for cooperation between two or more robots that are rigidly holding an object by allowing the arms to control the position and force on a designated point on the object.

The above approaches provide partial solutions, with various degrees of success, to the problem of robots physically operating simultaneously on a common object. However, they do not address the problem of coordinating robots that are working on separate tasks in a common workspace. In such an environment, collision-free

motions are required for the robots to safely perform their designated jobs.

However, most of the current collision avoidance schemes are directed toward avoiding contact between a robot arm and stationary objects in the work environment. Udupa [10] introduces the concept of transforming a robot into a point to find open paths. Brooks [11] represents free areas in forms of generalized cones through which the robot can travel. Lozano-Perez and Wesley [12] use a visibility graph to create a configuration space that represents corners of obstacles between which straight line travel of the manipulator is possible. In addition, Lozano-Perez [13], [14] provides approaches to the findpath and findspace problems using polyhedral representations of manipulator configurations that would produce collisions. Chien, Zhang, and Zhang [15] use the concepts of state space and rotation mapping to create a set of relationships between the positions and the corresponding collision-free orientations of a robot among obstacles. Finding collision-free paths for robots using this method is reduced to considering the connectivity of the graph represented by the set of relationships. A graphical simulation approach using configuration maps to plan manipulator paths in a two-dimensional workspace is given by Red and Truong-Cao [16]. Gilbert and Johnson [17] provide an approach to path planning by solving an optimal control problem with state constraints that ensure obstacle avoidance. The constraints are expressed in terms of the distance between potentially colliding parts of the robot and the obstacles. Kambhampati and Davis [18] use hierarchical representations based on quadtrees and staged path searching

methods for achieving collision-free paths. In a similar manner, Wong and Fu [19] use hierarchical path searching methods in three orthogonal two-dimensional projections of the three-dimensional environment to find a collision-free path. Oommen and Reichstein [20] provide algorithms for moving a manipulator represented by an ellipse among elliptical obstacles. A method for finding the minimum time motions for a manipulator between given end states with obstacle avoidance is given by Dubowsky, Norris, and Shiller [21]. A penalty function is used to account for the presence of obstacles and to provide constraints on the motion of manipulator joints. Recently, Singh and Wagh [22] have provided a path planning algorithm using intersecting convex shapes. A graph consisting of nodes representing all the largest rectangular free areas is created with intersecting areas being adjacent nodes. A cost function is utilized to find the path from the source node to the destination node in the graph. Rueb and Wong [23] structure free space into a set of overlapping convex regions which can be represented as a hypergraph. They introduce an approach to search the graph to obtain the characteristics of the robot's environment.

A literature review of all research directed at collision-free motion planning of multiple robots indicates that only a small amount of effort has been devoted to the problem of collision detection and avoidance. Gouzenes [24] proposes the use of graph-search techniques and petri-nets for collision avoidance between robots during an assembly operation. DuPourque, Guiot, and Ishacian [25] address distributed environments for multi-robot controllers using a hierarchical organization in which higher levels

perform the coordination, synchronization, and communication. Freund and Hoyer [26], [27], [28] provide an approach to collision avoidance by using a systematic design method for multi-robot systems. They propose a hierarchical structure for multi-robot systems using the dynamics of all the robots and a hierarchical coordinator for achieving collision-free paths. Canny [29] outlines a collision detection scheme for two moving robots represented as polyhedral objects. Tournassoud [30] uses the concept of separating hyperplanes to obtain avoidance between two manipulators. Fortune, Wilfong, and Yap [31] present a technique for motion planning for independent but synchronized motions of two robot arms each of which has two degrees of freedom movement. A joint feasible region is constructed that represents the set of placements of the arms that neither intersect the interior of an obstacle nor each other. Some of my past work [32] demonstrates the use of a concurrent processing environment to provide a collision-free coordination of independently controlled robots in a common workspace by using the techniques of concurrent programming and solving the problems of mutual exclusion, synchronization, and communication for a desired coordinated task. Roach [4], [33] discusses coordinating the collision-free motions of robot arms through the use of a robot operating system consisting of a task level planner, an execution monitor to govern execution, and low-level processes to control the robots.

Recently, Lee and Lee [34] have proposed an approach to collision-free motion planning of two robots by detecting and avoiding collisions using discrete time, straight line trajectories. As in

the detection and avoidance techniques presented in this report, they consider collisions to be between the wrists of each robot which are represented by sphere models. However, their discrete time approach must be performed off-line due to the time requirements in detecting collisions and calculating a collision region bounding box. Also, collisions that occur between discrete time instants cannot be accurately detected. In addition, their collision avoidance scheme is restricted to modify the path or trajectory of only one of the two robots.

This report investigates collision-free motion of two robots in a common workspace by detecting collisions along the straight line trajectories of the robots and then replanning the paths and/or trajectories of one or both of the robots to avoid the collision. The algorithm for collision detection is presented in Chapter 2. Various approaches to collision avoidance are presented in Chapter 3. Finally, Chapter 4 contains conclusions and future research that follows from this report.

2. COLLISION DETECTION

As discussed earlier, the sphere model for the wrist (Fig. 4) is used to detect and avoid collisions due to the fact that it is rotationally invariant and computationally efficient compared to other geometric models. The detection of a collision is accomplished by calculating the distance between the origins of two spheres. A collision is said to occur between the two wrists at any given time instant if the distance between the center of the two spheres is less than or equal to r1+r2.

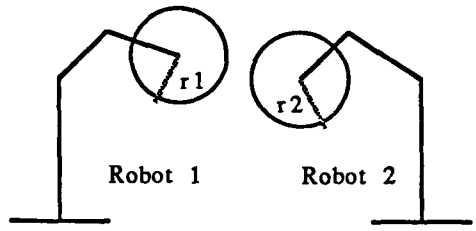


Figure 4 Sphere Model

One possible method of detecting collisions is to check at carefully chosen discrete time instants if the two spheres collide. As mentioned earlier, Lee and Lee [34] adopted this approach in which motion planning using such a collision detection technique had be done off-line.

The collision detection technique presented here involves

finding the segments on the straight line paths of the two robots where the possibility of collisions between the two wrists exists. Specifically, for each of the two straight line paths, a segment is found where, for each point of that segment, there exists at least one point on the other path which is less than or equal to r1+r2 distance apart. Such possible collisions, from now on, are referred to as potential collisions (i.e., collisions without considering the motion characteristics of the two robots). Since the ultimate objective is to avoid collisions, only the locations on each path where potential collisions begin and end are required to be determined. Once these points are known, trajectory information for each path can be used to determine if the potential collision is also a space-time collision (i.e., collisions taking motion characteristics into consideration). The overall approach to collision detection is explained in sections 2.1 and 2.2.

2.1. Detecting Potential Collisions

The position of a robot in three dimensional space (R^3) is defined by the center of the sphere representing the wrist with respect to a fixed global coordinate system (x,y,z). Any movement of the robot is represented by the path taken by the center of the sphere. Therefore, the straight line path of a robot, r, is specified by an initial point (x_{ri},y_{ri},z_{ri}) and a destination point (x_{rf},y_{rf},z_{rf}).

The detection of potential collisions involves finding the segments on the straight line paths of the robots where the possibility of collisions between the two wrists exists. Let the parametric equations of the straight lines representing the paths of

the two robots be

$$P_1 = P_{1i} + \lambda(P_{1f} - P_{1i})$$
$$P_2 = P_{2i} + \gamma(P_{2f} - P_{2i}) \qquad (1)$$

where $0 \le \lambda \le 1$ and $0 \le \gamma \le 1$. For a potential collision to occur,

$$\| P_1 - P_2 \| \le r1 + r2. \qquad (2)$$

Another way of computing the potential collisions is to obtain the intersections of a straight line representing one of the two paths with the locus of the surface of a sphere of radius r1+r2 whose center moves along the straight line representing the other path. This is equivalent to expanding the radius of the sphere of one robot by the radius of the other sphere while shrinking the other sphere to a point.

It is obvious in the case of straight line paths that there will be a continuous segment where the potential collisions exist. Fig. 5 shows potential collisions for two-dimensional paths through the use of a 2-D-Wrist-Potential-Collision Diagram (WPCD) in which each path contains its segment where potential collisions begin and end.

2.1.1. Generating Potential Collision Regions

In the standard cartesian coordinate system, a sphere of radius r=r1+r2 and center (x_c,y_c,z_c) is given by the equation

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = r^2. \qquad (3)$$

Letting the center of the sphere move along the path of robot 1, the straight line path of robot 1 can be parametrically defined as

$$x_c = x_{1i} + a_1\lambda$$
$$y_c = y_{1i} + b_1\lambda$$
$$z_c = z_{1i} + c_1\lambda \qquad 0 \le \lambda \le 1 \qquad (4)$$

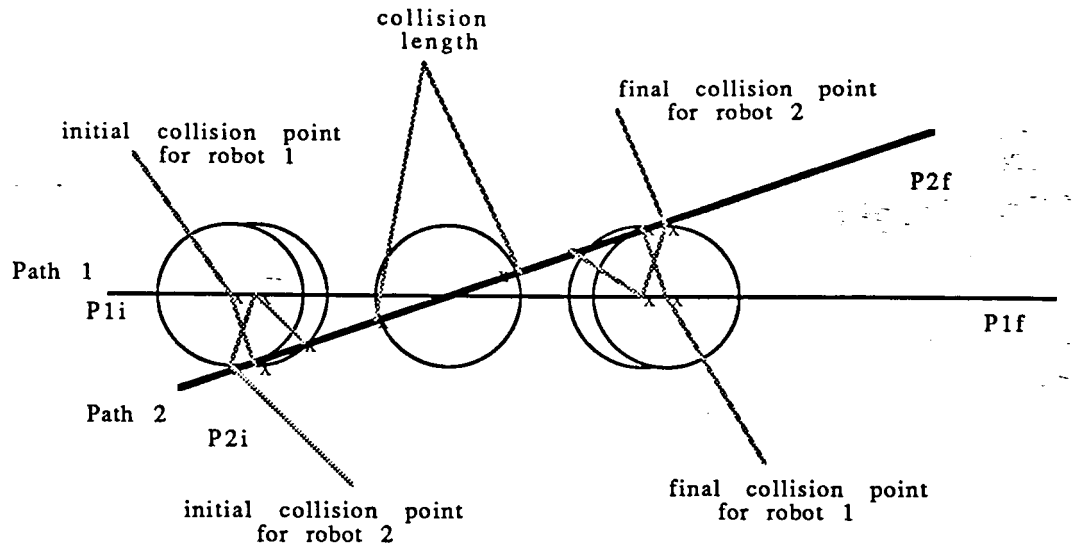


Figure 5 2D-Wrist-Potential-Collision Diagram (WPCD)

where

$$a_1 = x_{1f} - x_{1i}$$
$$b_1 = y_{1f} - y_{1i}$$
$$c_1 = z_{1f} - z_{1i} . \qquad (5)$$

The straight line path of robot 2 can be expressed as

$$x = x_{2i} + a_2\gamma$$
$$y = y_{2i} + b_2\gamma$$
$$z = z_{2i} + c_2\gamma \qquad 0 \le \gamma \le 1 \qquad (6)$$

where

$$a_2 = x_{2f} - x_{2i}$$
$$b_2 = y_{2f} - y_{2i}$$
$$c_2 = z_{2f} - z_{2i} . \qquad (7)$$

Thus, seven simultaneous equations with eight unknowns are formed. To generate the potential collision regions, equations (3), (4),

and (6) are combined to obtain

$$k_1\gamma^2 - 2k_2\gamma\lambda + k_3\lambda^2 + 2k_4\gamma - 2k_5\lambda + k_6 = 0$$

where

$$k_1 = a_2^2 + b_2^2 + c_2^2 \quad (9)$$

$$k_2 = a_1a_2 + b_1b_2 + c_1c_2 \quad (10)$$

$$k_3 = a_1^2 + b_1^2 + c_1^2 \quad (11)$$

$$k_4 = k_xa_2 + k_yb_2 + k_zc_2 \quad (12)$$

$$k_5 = k_xa_1 + k_yb_1 + k_zc_1 \quad (13)$$

$$k_6 = k_x^2 + k_y^2 + k_z^2 - r^2 \quad (14)$$

and

$$k_x = x_{2i} - x_{1i}$$
$$k_y = y_{2i} - y_{1i}$$
$$k_z = z_{2i} - z_{1i}. \quad (15)$$

Vectorially, equation (8) can be expressed as

$$\| A_2\gamma - A_1\lambda + K \| = r \quad (16)$$

where

$$A_1 = (a_1,b_1,c_1)$$
$$A_2 = (a_2,b_2,c_2)$$
$$K = (k_x,k_y,k_z) \quad (17)$$

which provides the positions along the straight line paths of robot 1 and robot 2 that are distance r apart.

For a given value of λ in equation (8), zero, one, or two values of γ can be found. This is equivalent to obtaining no intersections, one intersection, or two intersections with the other path for a given location of the sphere on its path. In case of two intersections, the distance on the straight line path between the two intersection points is called the potential collision length. Thus, equation (8)

represents the location of the intersection points with respect to λ (or vice versa with respect to γ) producing a parametric space potential collision region. A typical region is shown in Fig. 6 on a Parametric-Space-Potential-Collision-Region Diagram (PSPCRD). The ellipse in Fig. 6 is the most common case for parametric space potential collision regions and is discussed later. As stated earlier, only the segment on each path where potential collisions exist is required. The four extreme points of the ellipse are found by computing its horizontal and vertical tangents. These points, λ_{icp}, γ_{icp}, λ_{fcp}, and γ_{fcp}, represent the locations on each path where potential collisions begin and end.

2.1.2. Analyzing Potential Collision Regions

Equation (8) is a second-degree equation in two unknowns, λ and γ, where k_1, k_2, and k_3 cannot all be zero. The values of k_1 and k_3 are never zero due to the fact that the direction vector of a line must contain at least one non-zero element. Equation (8) defines a potential collision region in terms of parametric variables λ and γ. Unless equation (8) degenerates into straight lines, shrinks to a point, or is purely imaginary, it represents one of the following cases:

1)	a hyperbola	if	$k_2^2 - k_1k_3 > 0$	(18)
2)	a parabola	if	$k_2^2 - k_1k_3 = 0$	(19)
3)	an ellipse or a circle	if	$k_2^2 - k_1k_3 < 0$; a circle when $k_2 = 0$ and $k_1 = k_3$.	(20)

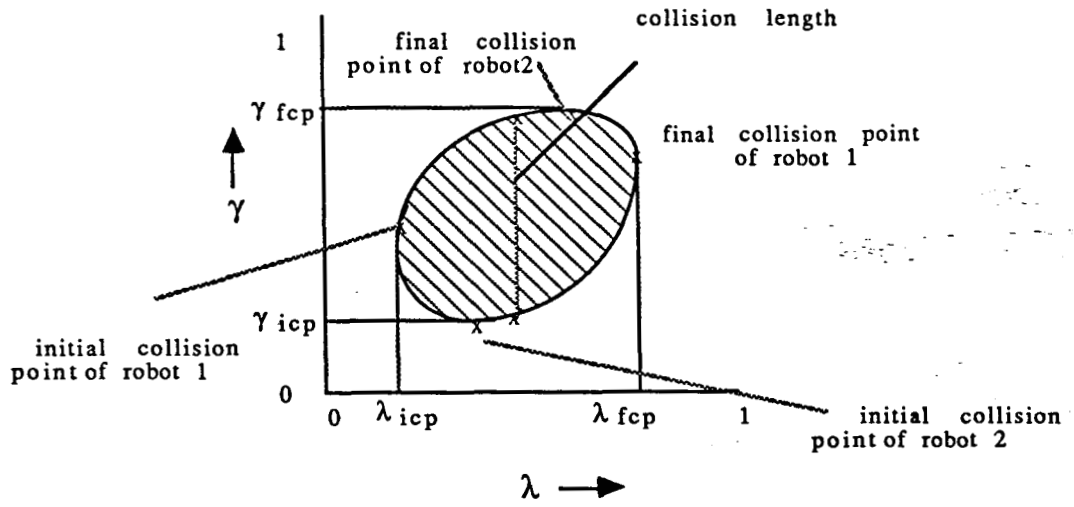


Figure 6 Parametric-Space-Potential-Collision-Region Diagram (PSPCRD)

A hyperbola will never be generated because application of the definitions from (17) to (18) produces the vector equation

$$A_1 \bullet A_2 > \| A_1 \| \| A_2 \| \tag{21}$$

which can never occur since the Cauchy-Schwarz Inequality shows that

$$|X \bullet Y| \leq \| X \| \| Y \| \tag{22}$$

where X and Y are vectors in R^3.

A parabola is generated when $k_2^2 = k_1 k_3$. This occurs when the paths of the robots are parallel. However, for this case, the parametric space potential collision region is the limiting form of a parabola which consists of a pair of parallel lines or a single line counted twice.

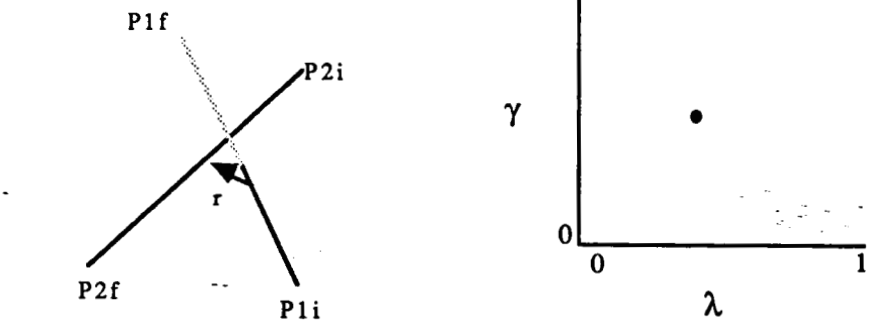
An ellipse is the most common case for the parametric space potential collision region. A circle is a limiting form of an ellipse and

occurs when $k_2=0$ and $k_1=k_3$. Coefficient k_2 equals zero when the direction vectors and, therefore, when the paths of the robots are perpendicular. When the sphere is cut by a perpendicular line, the distances of two new collision points from their corresponding previous collision points are equal producing a circular potential collision region. However, since the rate of change in λ and γ per unit path length may differ, perpendicular paths may still produce elliptical potential collision regions in λ and γ if the same scale is utilized. Therefore, the magnitudes of the paths must be equal ($k_1=k_3$) for a circle to be generated in the PSPCRD. When the paths of the robots are not perpendicular, an ellipse is generated because the distances of two new generated collision points from their corresponding previous collision points are unequal.

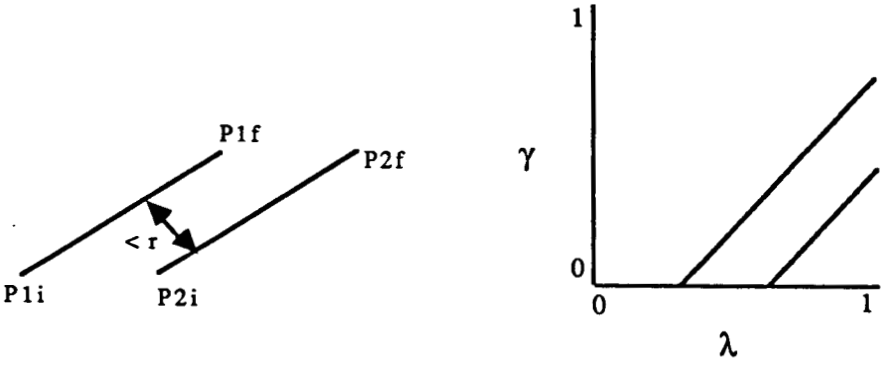
When the roots of equation (8) are imaginary, potential collisions and, therefore, space-time collisions do not exist. Equation (8) degenerates into straight lines when the paths of the robots are parallel and this represents the parabola case. A single root of equation (8) can exist representing a point collision. In summary, points, lines, and ellipses are the possible collision regions that can occur in the PSPCRD as depicted in Fig. 7.

2.1.3. Determining Potential Collision Segments

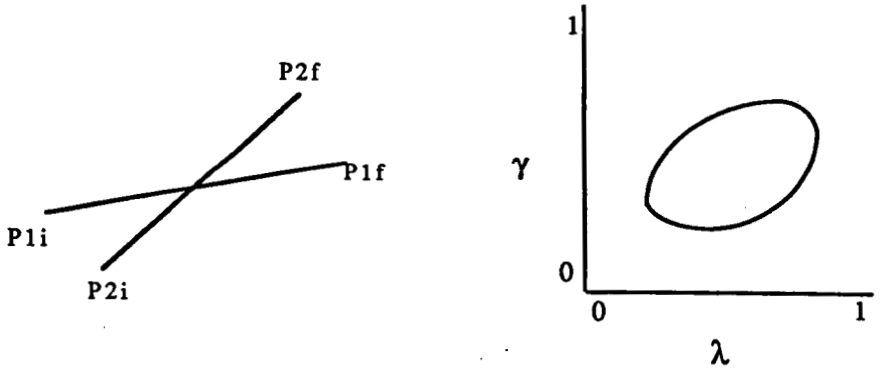
As shown in section 2.1.2, points, lines, and ellipses are the possible collision regions that can occur in the PSPCRD. The PSPCRD can be divided into nine sub-regions as shown in Fig. 8. Each sub-region specifies where on the paths of the two robots the



Point Collision Region



Line Collision Region



Ellipse Collision Region

Figure 7 Possible Potential Collision Regions

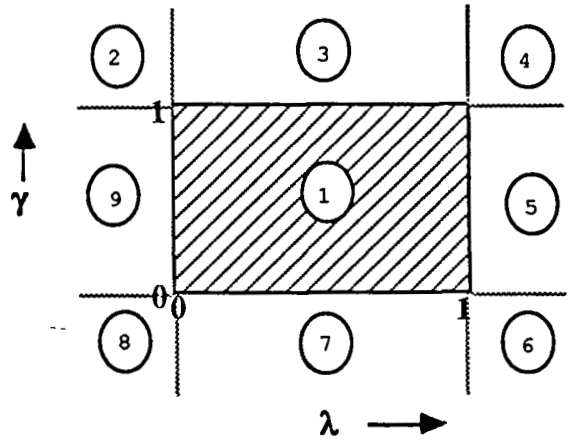


Figure 8 The Nine Sub-Regions of the PSPCRD

collision is occurring. Sub-regions with corresponding values of λ and γ that are less than zero or greater than one are indicating points that exist on the lines containing the robots' paths but are before the starting points and after the ending points of the paths, respectively. Sub-region 1 contains the potential collision information that occurs along the paths of the robots. Other sub-regions correspond to the invalid locations of potential collisions.

As discussed, only the four extreme points (smallest and largest values of λ and γ) in sub-region 1 are required. These values represent where collisions begin and end on each path. However, a single root (point collision) requires finding only the one value of λ and γ causing the collision. The point can exist in any of the nine sub-regions of which only sub-region 1 contains a valid collision. Otherwise, a potential collision and, therefore, a space-time collision does not exist. Detection of a point collision involves finding the minimum distance between the lines representing the paths of the robots. The minimum distance between the lines containing the

paths of the robots can be found by differentiating the equation

$$d^2 = k_1\gamma^2 - 2k_2\gamma\lambda + k_3\lambda^2 + 2k_4\gamma - 2k_5\lambda + k_7 \quad (23)$$

where

d is the distance between the two lines, k_1 to k_5 are the same as in equation (8), and $k_7 = k_x + k_y + k_z$

with respect to λ and γ and solving the results simultaneously to obtain

$$\gamma = (k_2k_5 - k_4k_3)/(k_1k_3 - k_2^2) \quad (24)$$

$$\lambda = (k_2\gamma + k_5)/k_3. \quad (25)$$

Equations (24) and (25) provide the condition for minimum distance between the lines since the second derivatives of equation (23) with respect to γ is k_1 and with respect to λ is k_3 which are always greater than zero. If the minimum distance is equal to r and is along the robot paths ($0\leq\lambda\leq1, 0\leq\gamma\leq1$), then a point collision occurs at those particular values of λ and γ. If the minimum distance lies outside the paths of the robots, then checking whether collisions exist at the starting and finishing points (end points) of the two paths will provide the necessary information since the paths are converging. A collision exists at an end point when the sphere placed at that end point produces a valid intersection ($0\leq\lambda\leq1$ or $0\leq\gamma\leq1$) with the other path. Placing the sphere at each of the four end points is accomplished by solving equation (8) for $\lambda=0$, $\lambda=1$, $\gamma=0$, and $\gamma=1$. A point collision occurs whenever: 1) one end point has only one intersection that is valid, or 2) two end points have only one valid

intersection of which each is at the end point of the other line. In either case, the collision point occurs at the end point of one path with either the end point of the other path or a location along the other path defined by the single intersection.

A similar procedure is applied to paths that are parallel with the requirement that four extreme values are needed. If the minimum distance between the parallel lines representing the robots' paths is greater than r or if no end points provide a valid intersection, then a potential collision and, therefore, a space-time collision does not occur. Each end point that is found to produce a potential collision is one of the extreme values. Any missing values are then determined from the valid intersection points. Such needed information can be obtained from sub-region 1 of the PSPCRD.

As mentioned earlier, the elliptical parametric space potential collision region is the general case. It is produced by all combinations of paths that result in potential collisions with the exception of those that are parallel or result in point collisions. The ellipse case represents a complete collision, one that begins and ends, along the lines that represent the paths of the robots as seen in the WPCD of Fig. 5. The four extreme points of the ellipse are found by calculating its horizontal and vertical tangents through implicit differentiation of equation (8) with respect to λ and γ. The equations producing the horizontal tangents (maximum and minimum values in γ) are

$$(k_1(k_3/k_2)^2-k_3)\lambda^2 + ((2k_3/k_2)(k_4-(k_5k_1)/k_2))\lambda + (k_1(k_5/k_2)^2-(2k_4k_5)/k_2+k_6) = 0 \quad (26)$$

$$\gamma = (k_3\lambda - k_5) / k_2 \quad (27)$$

and producing the vertical tangents (maximum and minimum values in λ) are

$$(k_3(k_1/k_2)^2-k_1)\gamma^2 + ((2k_1/k_2)((k_3k_4)/k_2-k_5))\gamma + (k_3(k_4/k_2)^2-(2k_4k_5)/k_2+k_6) = 0 \quad (28)$$

$$\lambda = (k_1\gamma + k_4) / k_2 . \quad (29)$$

The horizontal and vertical tangents provide the information for determining the four extreme values (smallest and largest values of λ and γ) that represent the locations on each path where collisions begin and end. If the tangents are imaginary, potential collisions and, therefore, space-time collisions do not exist. If they are valid (complete collision occurs along the path of both robots), an elliptical pararmetric space potential collision region is generated in sub-region 1 of the PSPCRD. Thus, the four extreme values are known from the four tangent values.

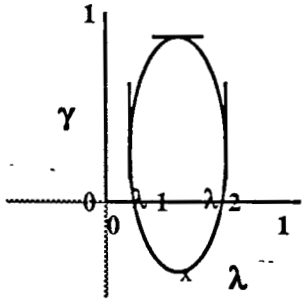
However, tangents are invalid when they are generated in any sub-region of the PSPCRD except sub-region 1. Each invalid tangent is produced by an extreme collision that occurs on the lines representing the paths, but not on the paths, of one or both of the robots. When three tangents are valid ($0\leq\lambda\leq1$ and $0\leq\gamma\leq1$), the fourth lies in section 3, 5, 7, or 9 of the PSPCRD. The missing extreme value is the end point (0 or 1) of one of the paths and can be found by: 1) placing the sphere on the end points of the path with the missing extreme and determining which end point has valid intersections, 2) using the known three extreme values to determine which end point must be the fourth, or 3) using the sign of the missing value from the invalid tangent to determine which end point of the path has

been surpassed.

A similar but more complicated procedure for determining the extreme intersection values is followed when zero, one, or two tangents are valid. The invalid tangents can lie in any of the nine sub-regions of the PSPCRD except sub-region 1. When zero, one, or two tangents are valid, each end point that is found to produce a collision is one of the extreme values. Any missing values are then determined from the valid intersections. Sub-region 1 of the PSPCRD contains all the required information. This procedure is similar to that when the paths are parallel. However, the sphere is only required to be placed at the end points that produce the invalid tangents. Each of these endpoints are determined using the sign of the missing extreme value from the invalid tangent. When all tangents are invalid and no end points produce an intersection, a potential collision and, therefore, a space-time collision does not occur. Fig. 9 provides examples of the ellipse case.

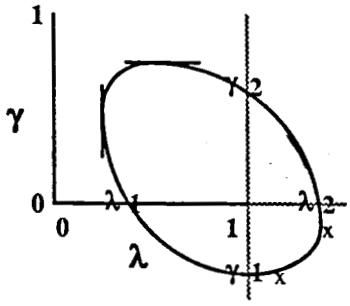
2.2. Detecting Space-Time Collisions

Once the locations on each path where potential collisions begin and end are known (λ_{icp},λ_{fcp},γ_{icp},γ_{fcp}), trajectory information can be used to determine whether a space-time collision exists. If a space-time collision is likely to occur, it has to happen within the potential collision segment on each robot's path. Using trajectory information (velocity, acceleration, and location of break points), the time range when the potential collisions along each path occur can be determined. Any overlap in the two time ranges suggests, but does not guarantee, the existence of a space-time collision. It is obvious



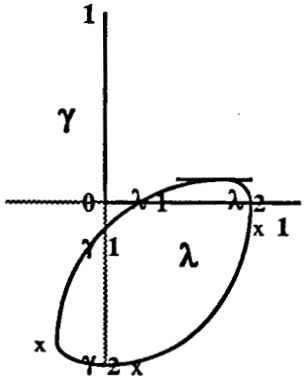
unknown extremes:
$\gamma = 0.$

Three Valid Tangents



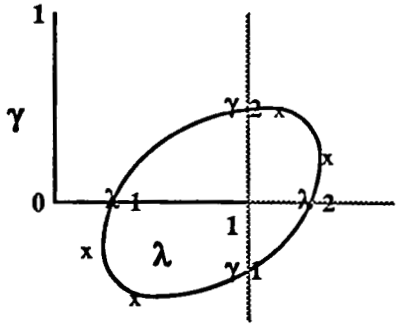
unknown extremes:
$\gamma = 0$
$\lambda = 1$

Two Valid Tangents



unknown extremes:
$\gamma = 0$
$\lambda = \lambda_1$
$\lambda = \lambda_2$

One Valid Tangent



unknown extremes:
$\gamma = 0 \quad \lambda = \lambda_1$
$\gamma = \gamma_2 \quad \lambda = 1$

Zero Valid Tangents

Figure 9 Examples of Ellipse Case

that the robots can simultaneously be within their potential collision segments and never collide in time. For example, one robot can be leaving its potential collision segment while the other robot is just entering its segment. If these locations are not within colliding distance, no space-time collision will occur. In other words, common time ranges relate information that each robot is colliding with every location on the potential collision segment of the other robot. This, of course, is incorrect. Therefore, an overlap in the time ranges when potential collisions occur is necessary, but not a sufficient condition, for determining if a space-time collision is going to happen.

Thus, the method of detecting space-time collisions involves two steps: 1) determination of an overlap in the time ranges when potential collisions along each path occur to assure the possibilty that a space-time collision can happen, and then 2) establishment of the existence of a space-time collision.

2.2.1. Determining Common Time Ranges

Using the trajectory information of a robot, the distance traveled along its straight line path per unit time can be calculated. Letting distance traveled be defined by a parametric position, the following equations can be defined:

$$\lambda = \mathbf{f}_{\lambda}(t) \tag{30}$$

$$\gamma = \mathbf{f}_{\gamma}(t) \tag{31}$$

and

$$t_{\lambda} = \mathbf{f}_{\lambda}^{-1}(\lambda) \tag{32}$$

$$t_{\gamma} = \mathbf{f}_{\gamma}^{-1}(\gamma). \tag{33}$$

Equations (30) and (31) give the parametric positions on each path where the robots are located in time, and equations (32) and (33)

perform the inverse which provides the time at which the robots reach specific locations along their paths.

Using the time equations (32) and (33), the time range when potential collisions along each path occur can be determined. If an overlap in time ranges does not occur, a space-time collision cannot exist since a collision is only possible within the potential collision segment on each path. As stated earlier, if the time ranges overlap, a space-time collision may or may not occur. A Space-Time-Collision-Region Diagram (STCRD) is shown in Fig. 10 which combines both path and trajectory information for a single break point case for each of the two trajectories. In this situation, an overlap in the time ranges does occur and a space-time collision region is formed. This region represents the positions along each path where a possiblity of a space-time collision exists.

The collision region in Fig. 10 is formed by an overlap in the time ranges such that the time values for the robots alternate. However, when an overlap occurs in which one robot's time range is contained totally within the time range of the other robot, a space-time collision is guaranteed to occur. This happens because one robot traverses its whole potential collision segment while the other robot, which was already within its potential collision segment, still remains in its potential collision segment. Thus, a space-time collision is assured to occur somewhere within the enclosed time range. This situation is further discussed in the next section.

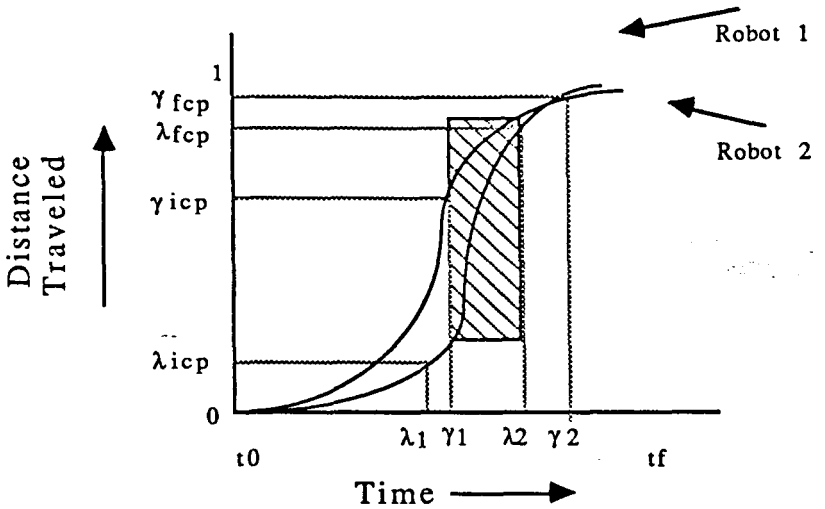


Figure 10 Space-Time-Collision-Region Diagram (STCRD)

2.2.2. Establishing Existence of Space-Time Collisions

The objective of collision detection is to establish whether or not a space-time collision occurs between two robots. The collision region in the STCRD represents the positions along each path where a space-time collision can exist due to the overlap in the time ranges when potential collisions along the paths of the robots occur. The region is delimited by the time of the initial potential collision of one of the two robots and by the time of the final potential collision of one of the two robots. The starting time and ending time of the region are denoted by t_α and t_β, respectively as shown in Fig. 11. This time range defines a curve in the STCRD for each robot. The curves represent the motion characteristics of the robots along their paths where space-time collisions are likely.

At a given time, the position of each robot can be determined by the position equations (30) and (31). Therefore, any time instant

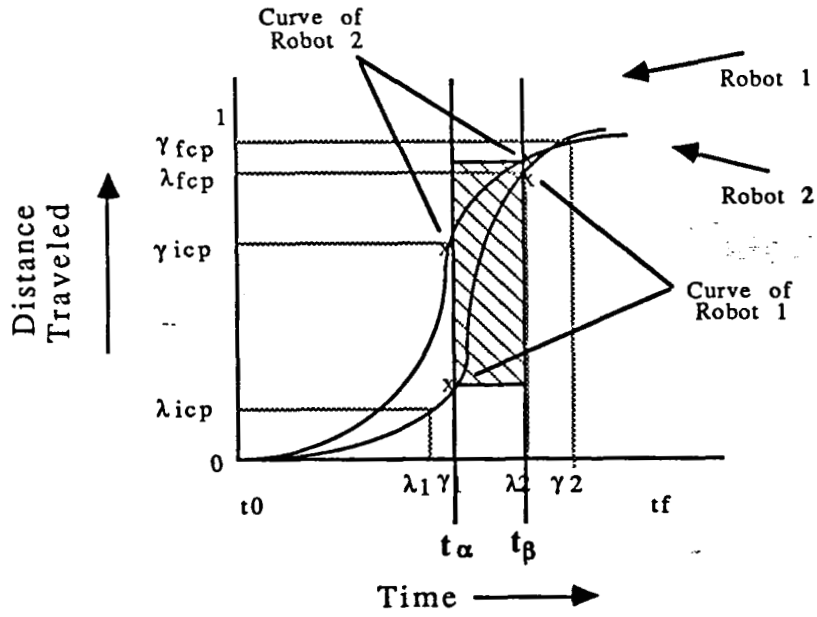


Figure 11 Time Range Curves in the STCRD

determines a (λ,γ) pair which can be transformed to another domain for analysis. This domain is shown in Fig. 12 as a Potential-Collision-Region-Motion Diagram (PCRMD). It is guaranteed that each (λ,γ) pair within t_α and t_β must be within the bounding box defining the potential collision region since t_α and t_β represent the common time range of the potential collision segments which form the bounding box. In other words, each value of λ and γ between t_α and t_β in the STCRD lies between λ_{icp} and λ_{fcp}, and γ_{icp} and γ_{fcp}, respectively in the PCRMD.

Thus, the two curves defined by t_α and t_β in the STCRD will result in one curve in the PCRMD. Therefore, the motion characteristics of each robot corresponding to possible space-time collisions can be analyzed with respect to the potential collision

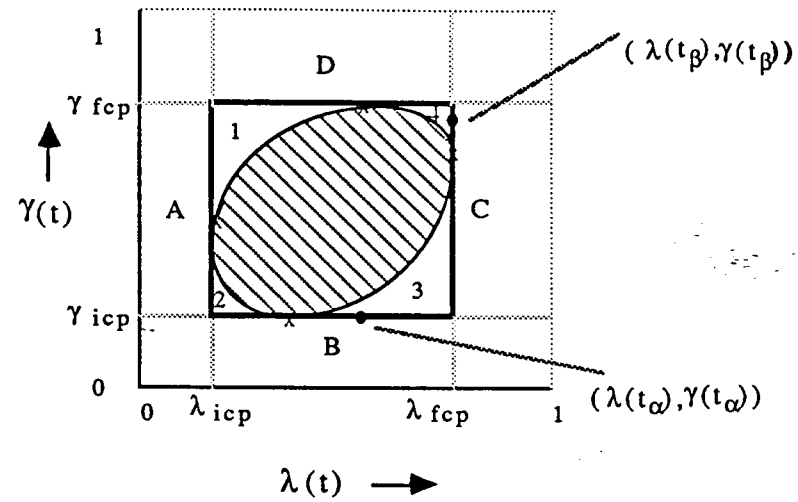


Figure 12 Potential-Collision-Region-Motion Diagram (PCRMD)

region. Let the starting point and the ending point of the curve in the PCRMD be defined as $C(\lambda(t_\alpha),\gamma(t_\alpha))$ and $C(\lambda(t_\beta),\gamma(t_\beta))$, respectively. The objective is to find whether or not this curve intersects the potential collision region. If the potential collision region is intersected by the curve, a space-time collision occurs since both robots are within colliding distance at some point in time defined by the (λ,γ) pair.

Since t_α always represents the time of an initial potential collision of one of the robots, the starting point of the curve can lie on either segment A or B in Fig. 12. Likewise, since t_β always represents the time of a final potential collision of one of the robots, the ending point of the curve can lie on either segment C or D. Since each of the functions in the STCRD is monotonically increasing, the (λ,γ) pairs that connect any two points on the curve in the PCRMD, such as the starting and ending points, must monotonically increase with respect to time. The valid area between any two points which the curve can travel is shown in Fig. 13. Fig. 14 shows a possible

situation in which the curve passes through the potential collision region. Since an intersection is found, a space-time collision does occur during the time range specified by the (λ,γ) pairs at the locations where the curve enters the region and leaves the region. Thus the time range for a space-time collision can be reduced from the overlapped time range to the actual time range if those locations can be found. Reducing the space-time collision region will be elaborated on further in a later section.

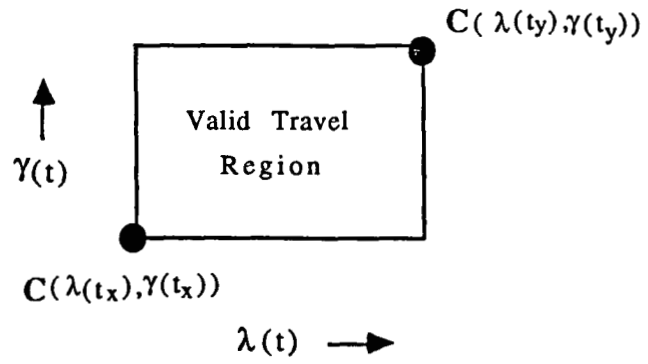


Figure 13 Valid Travel Region for the Monotonically Increasing Curve in the PCRMD

When the starting point of the curve lies on segment A or B in region 2, the curve must intersect the potential collision region to connect with its ending point. Therefore, a space-time collision does occur since at some point in time, the robots will be within colliding distance. Also notice that if the curve traverses either from segment A to C or from segment B to D, a space-time collision always occurs. This situation happens when an overlap in time ranges occurs such that one robot's time range is totally contained within the other as

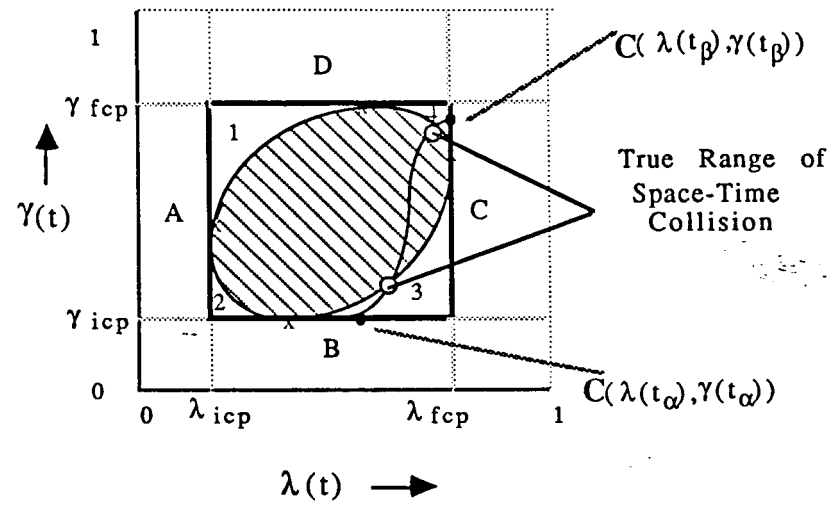


Figure 14 Motion Curve with True Collision Range

discussed in section 2.2.1. If the curve traverses from segment A, region 1 to segment D, region 4, a space time collision occurs. Similarly, if the starting point of the curve lies on segment B, region 3 and the ending point lies on segment D, region 4, a space-time collision occurs. However, if the curve travels from segment A, region 1 to segment D, region 1 or from segment B, region 3 to segment C, region 3, a space-time collision may or may not occur as shown in Fig. 15.

Determination of the existence of a space-time collision for these two cases is accomplished by a fast iterative algorithm. The algorithm assumes that if the curve comes within some threshold distance from the potential collision region, a space-time collision occurs. In other words, the potential collision region can be thought of as being expanded slightly.

The algorithm uses the fact that the (λ,γ) pairs defining the curve in the PCRMD monotonically increases with respect to time. The idea is to divide the curve into regions of bounding boxes such

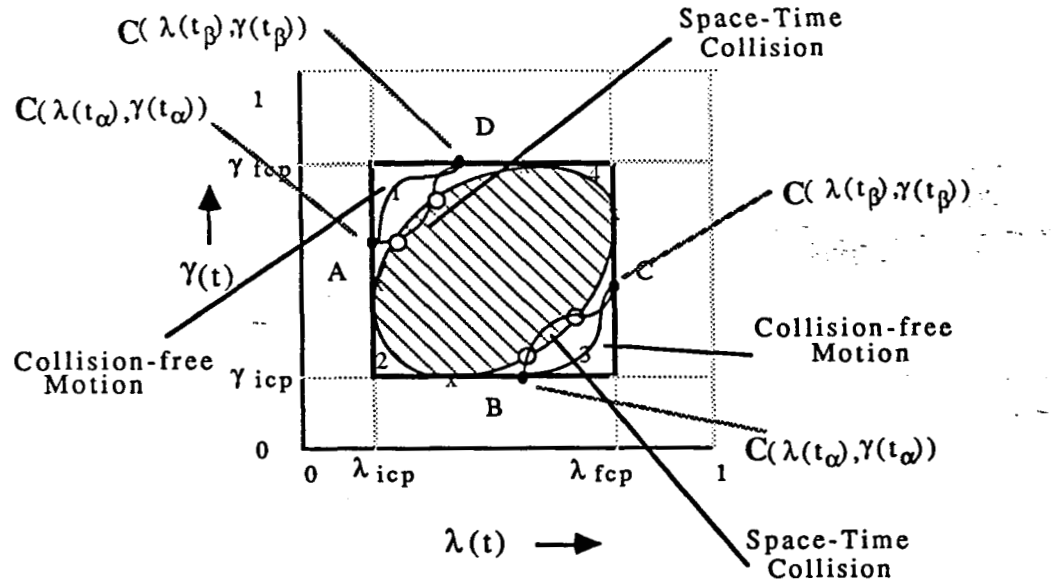
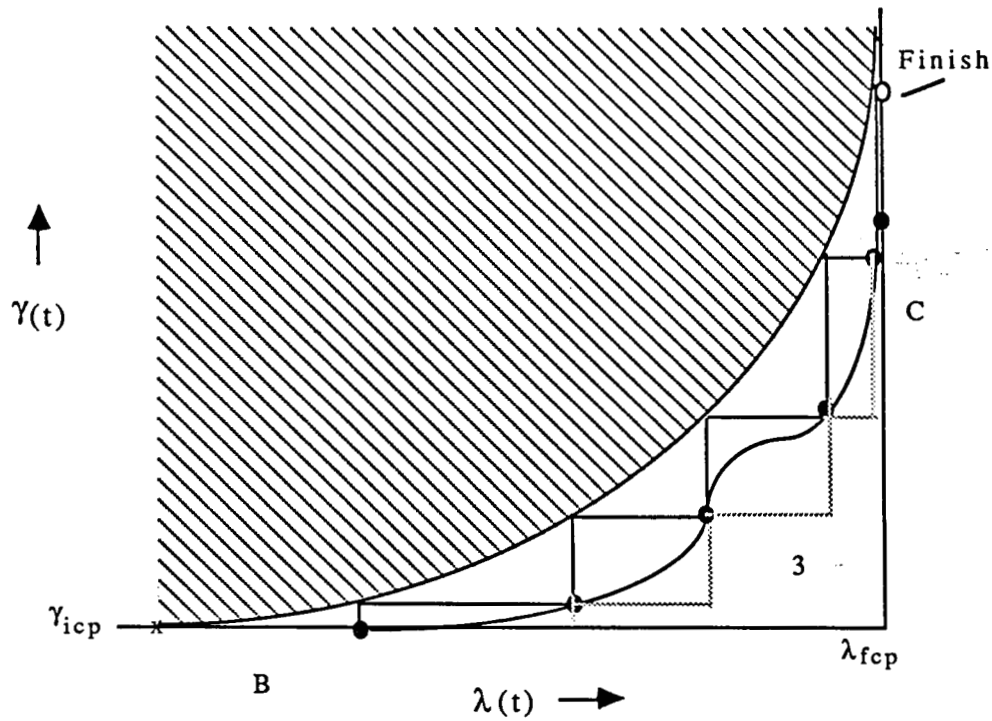
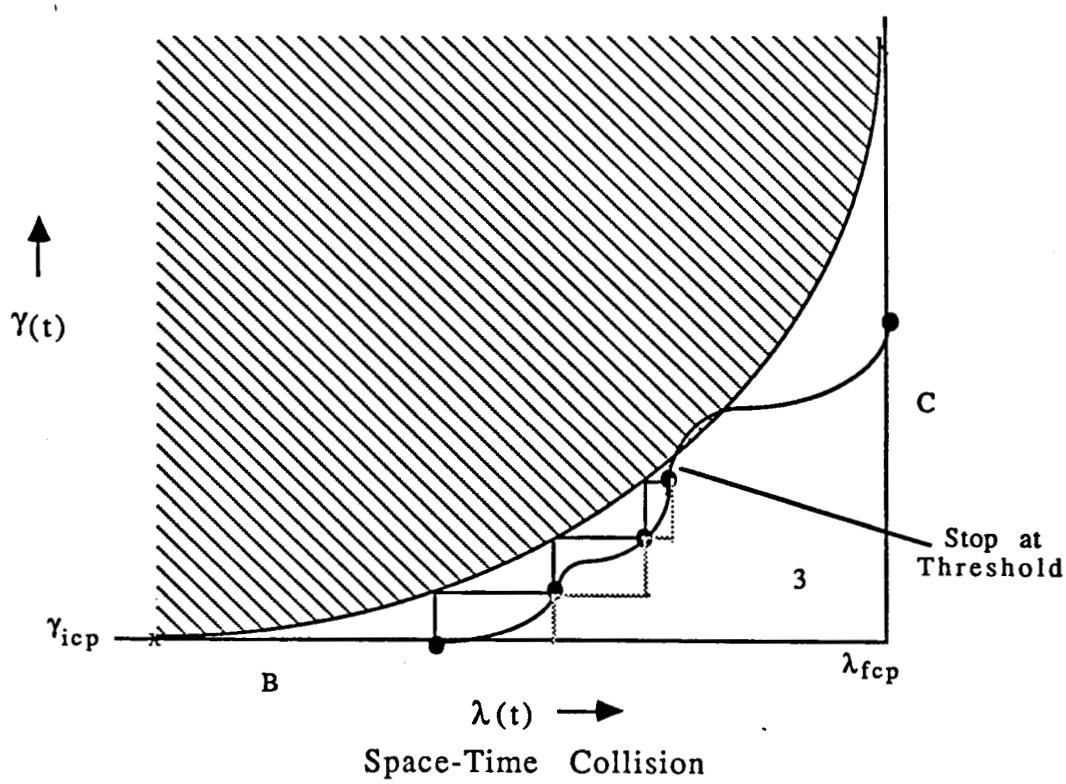


Figure 15 Exception Cases for Region One and Three

that the curve traversal within each bounding box is known. Such a bounding box is shown in Fig. 13 as the valid travel region for a monotonically increasing curve. Fig. 16 illustrates the algorithm for both cases where the curve does and does not intersect the potential collision region in region 3. Applying the approach to region 1 is straightforward. The reader should consult Fig. 16 to better appreciate the ensuing discussion. Basically, bounding boxes from the starting point to the ending point of the curve (left to right in λ and bottom to top in γ) are created. Each bounding box is formed by two points along the curve. From the starting point, the position of the potential collision region is determined by calculating the γ value using the λ value of the starting point. Of particular interest is the location on the curve at the γ value defined by the potential collision region position. Using the STCRD, the λ value of this location is found thus forming a (λ,γ) pair of a point on the curve and the second



Collision-Free Path



Space-Time Collision

Figure 16 The Iterative Algorithm

point defining the bounding box. Therefore, within this bounding box, the curve cannot intersect the potential collision region. The new (λ,γ) pair is the starting point for generating the next bounding box. This iteration continues until the γ value of the potential collision region position is larger than the γ value of the ending point of the curve. When this condition is satisfied, it ensures that the curve could never intersect the potential collision region and thus, no space-time collision occurs. Notice that at each iteration of generating a bounding box, only one new (λ,γ) point on the curve needs to be calculated since the previous (λ,γ) point is being used as the other point in the bounding box pair. Therefore, a progression towards the endpoint of the curve is being performed.

However, if the curve intersects the potential collision region, an infinite amount of bounding boxes are generated, each of which is decreasing in size as the curve approaches the potential collision region. Therefore, the iteration stops when the distance between potential collision region positions is within some specified threshold. This is similar to expanding the potential collision region. The threshold defines a distance the curve has to be from the potential collision region in order to state a space-time collision occurs. The maximum possible distance happens when the threshold value is produced by an equilateral triangle. Notice that if a curve traverses near the potential collision region, but never intersects the potential collision region, it is treated as a space-time collision although one never existed. The maximum number of iterations to determine whether a space-time collision exists can be expressed as

$$\text{Max Iterations} < L/\Delta T \tag{34}$$

where L is the length of the potential collision region segment and ΔT is the threshold value.

In the above examples, a full ellipse was used as the potential collision region in the PCRMD and in the iterative algorithm. The approach is also applicable for the other cases. Fig. 17 shows an example of a non-full ellipse case and Fig. 18 illustrates a parallel path situation. The point potential collision case is trivial. When one occurs, the λ and γ values of the point must produce the same time instant in the STCRD for a space-time collision to exist.

2.3. Role of Collision Detection in Avoidance

When a space-time collision is detected, the paths and/or trajectories of one or both of the robots must be modified to avoid the collision. The space-time collision region in the STCRD is formed by common time ranges when potential collisions along the paths occur. The objective is to eliminate the space-time collision region by producing non-overlapping time ranges. As presented previously, this time range is the extreme situation. The intersection points of the curve with the potential collision region in the PCRMD represent the actual time range of the space-time collision when mapped to the STCRD. Reducing the space-time collision region may help in some of the avoidance techniques that are discussed in Chapter 3.

Therefore, depending on the avoidance technique, it may be advantageous to reduce the space-time collision region whenever possible. When the curve in the PCRMD travels from a point on segment A, region 1 or from a point on segment B, region 3, the

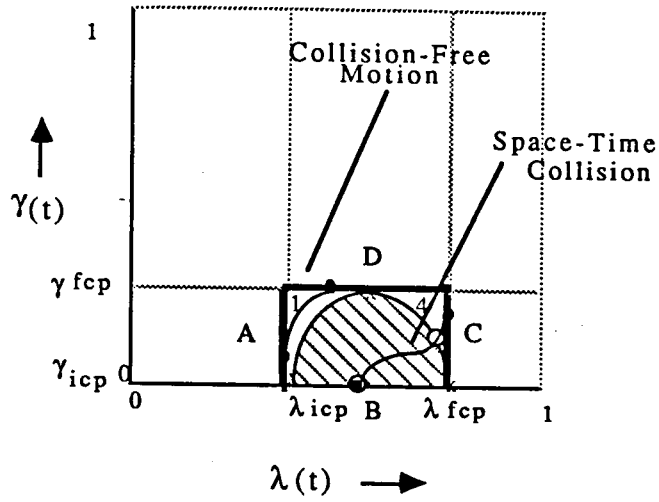


Figure 17 Non-Full Ellipse Case

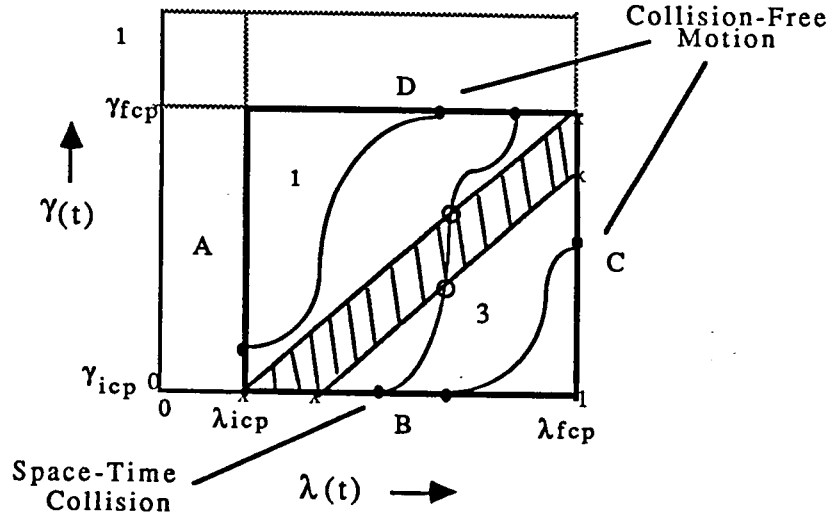


Figure 18 Parallel Path Situation

iterative algorithm approaches the actual collision points. The value produced when the iteration stops can be used as a new t_α value when mapped to the STCRD. In addition, when the above curves travel to point on segment D, region 1 and segment C, region 3, respectively, the algorithm can also be applied backward from the ending point. Therefore, when the iteration stops, a new t_β is formed when mapped to the STCRD. Thus, the space-time collision region is reduced. This identical procedure can be applied to parallel lines that slope from left to right in the PSPCRD. Basically, if the iterative algorithm is being used to determine the existence of space-time collisions, there is no penalty in using the acquired information to reduce the space-time collision region.

2.4. Summary of Collision Detection Algorithm

The collision detection algorithm is summarized in pseudo-code form as follows:

Procedure 1 -> Detect Potential Collisions (2.1)
- Step 1: Generate Potential Collision Regions (2.1.1)
- Step 2: Determine what type of region is generated (ellipse, line, or point) (2.1.2)
- Step 3: Determine Potential Collision Segments which are the points on each path where potential collisions begin and end (2.1.3)

Procedure 2 -> Detect Space-Time Collisions (2.2)
- Step 1: Determine if the time ranges overlap when potential collisions along each path occur (2.2.1)
- Step 2: Using the beginning and ending times of the overlapped time range, determine the existence of a space-time collision by mapping between time domain and position domain (2.2.2)

Step 3: If the mapping in Step 2 is inconclusive, determine the existence of a space-time collision by applying an iterative algorithm to the mapping between time and position domains (2.2.2)

Procedure 3 -> Reduce Space-Time Collision Region (2.3)

Reduce Space-Time Collision Region depending on avoidance technique by using information from the iterative algorithm (2.3)

3. COLLISION AVOIDANCE

To obtain collision-free motion, a collision in time and space is avoided by modifying the paths and/or trajectories of one or both of the robots. The method of achieving this collision-free motion depends on the avoidance requirements which modify various trajectory and path parameters. An overview of collision avoidance is presented in section 3.1. In section 3.2, some approaches to collision avoidance are discussed.

3.1. Overview of Collision Avoidance

A space-time collision region in the STCRD is depicted by common time ranges when potential collisions along the paths occur (or a reduced range when possible). The objective of collision avoidance is to eliminate the space-time collision region by producing non-overlapping time ranges. This is accomplished by modifying the paths and/or trajectories of one or both of the robots.

3.1.1. Trajectory Modification

Trajectory modification involves alteration of the motion characteristics of a robot along its path. Several parameters define the motion characteristics of a robot. These include number of break points, position of break points, chosen constant acceleration, and the starting time of motion. Any combination of the above parameters

on one or both of the robots can be modified to achieve collision-free motion. Specifically, the new motion characteristics of the robots are such that the time ranges when potential collisions occur along each path (or reduced time ranges) do not coincide.

Each parameter influences the motion characteristics differently. To begin with, break points contribute significantly to the motion characteristics of a robot. The point at which a robot arm decelerates after previously accelerating, or accelerates after previously decelerating, is called the break point. Fig. 19 shows a preplanned straight line trajectory with one break point at λ_B. The robot is accelerating along the path between the starting position and the break point and it is decelerating along the path between the break point and the final destination. For collision avoidance purposes, the break point can be moved to position λ_a or λ_b. The effects of repositioning the break points on one or both of the robots may result in a collision-free motion as seen in Fig. 20 when the break point of each robot is modified.

Repositioning the break points alone may not always produce a desirable outcome. Another option is to modify one or both of the preplanned trajectories by providing additional break points. Choosing the proper number and position of the break points can result in collision-free motion. Fig. 21 shows the effects of adding break points γ_{B1} and γ_{B2} to the preplanned trajectory of robot 2. In this situation, robot 2 accelerates along the path between the starting position and γ_{B1}, decelerates between γ_{B1} and γ_{B2}, accelerates between γ_{B2} and the original break point, and finally decelerates

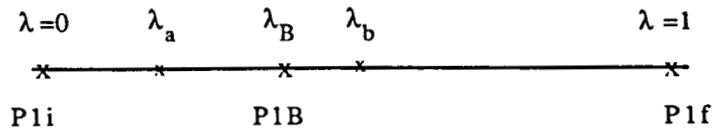


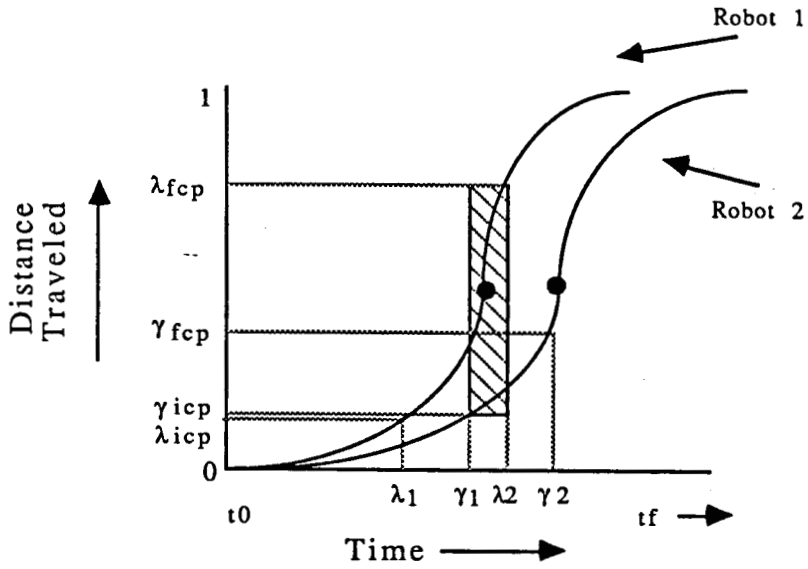
Figure 19 Preplanned Trajectory with One Break Point

between the original break point and the final destination. The two new break points cause a separation of the overlapped time ranges and thus provide collision-free motion.

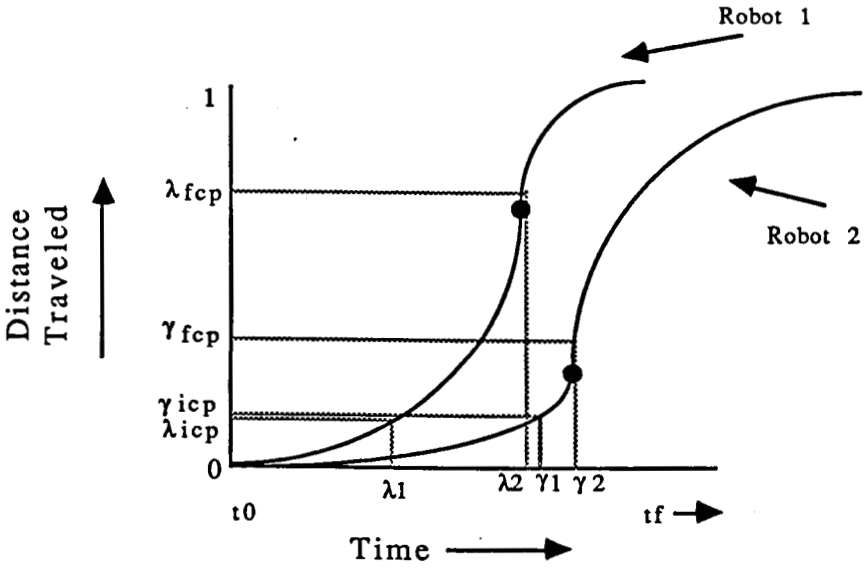
In addition to break point modification, which may sometimes require acceleration reduction, the acceleration itself along the path of a robot can be manipulated. Assuming the preplanned trajectory of a robot uses maximum constant acceleration for minimum time travel, smaller acceleration values can be chosen for collision avoidance purposes. In other words, a reduction in the speed of a robot, without causing substantial delay, can be used to avoid a collision.

A final parameter affecting the motion characteristics of a robot is the time at which motion begins. Assuming a robot cannot begin its motion earlier in time, postponing the starting time of motion, within a reasonable delay, can provide collision-free motion. Fig. 22 shows an example of time postponement. If the motion of robot 2 begins λ_2-γ_1 later in time, the common time ranges separate and therefore, collision-free motion is obtained.

In summary, collision-free motion may be achieved by altering any combination of the above trajectory parameters that contribute to the motion characteristics of one or both of the robots.

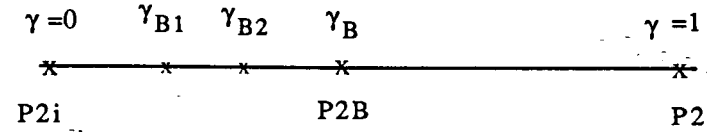


Before Break Point Modification

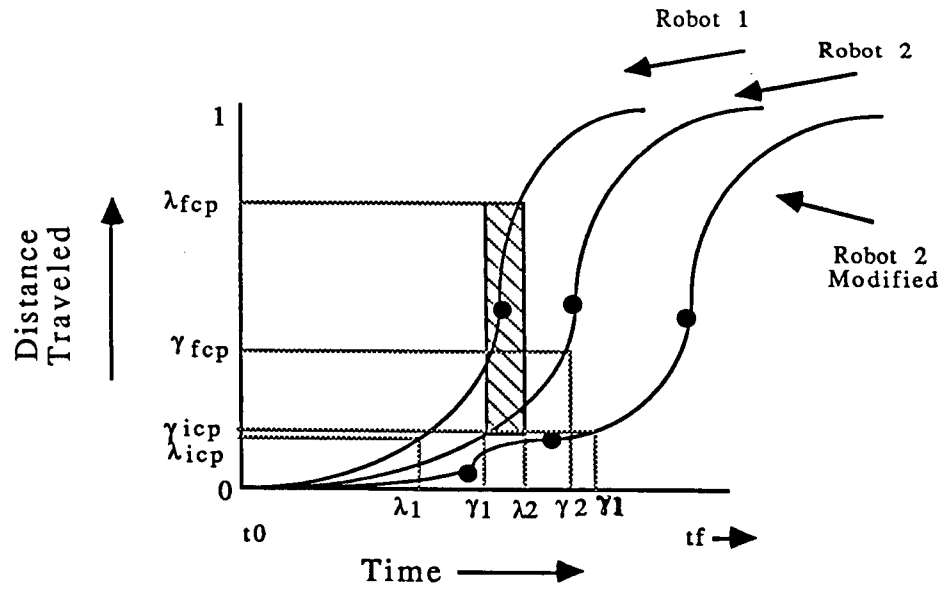


After Break Point Modification

Figure 20 Collision-Free Motion by Repositioning the Break Points



Trajectory with Two Additional Break Points



STCRD with Break Point Modification

Figure 21 Collision-Free Motion by Providing Additional Break Points

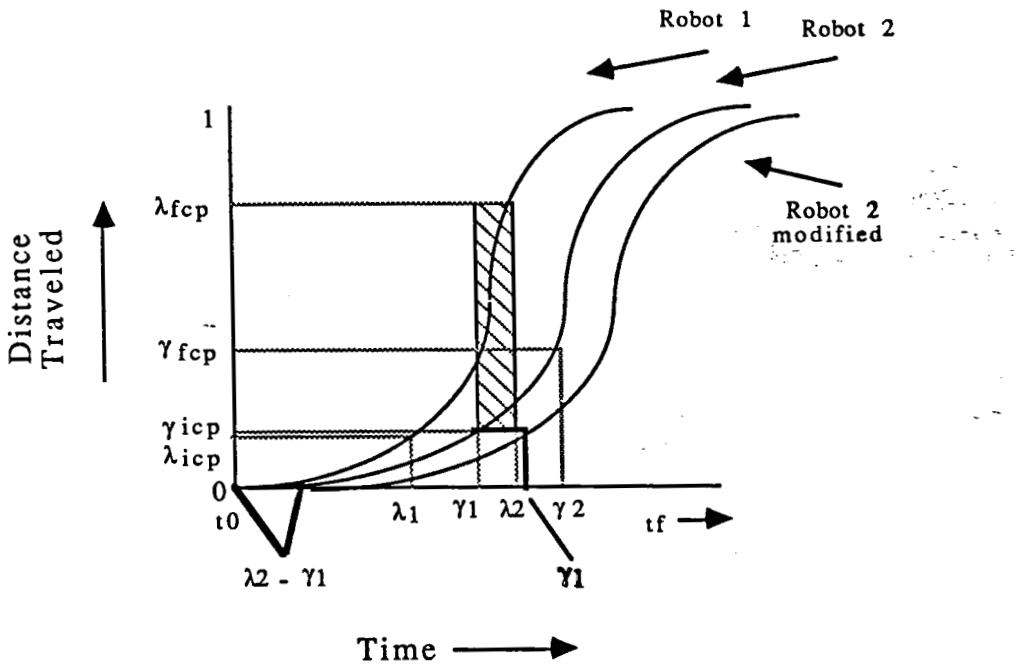


Figure 22 Collision-Free Motion by Time Postponement

3.1.2. Path Modification

Path modification involves alteration of the path of a robot. Since the initial and destination points of a path remain the same, the straight line path of a robot is modified to two or more connected straight line paths which avoid the collision as shown in Fig. 23. This is consistent with the initial assumption requiring straight line paths. Various parameters describing a new travel route are the number of straight line path segments, the average deviation from the original path, and the new travel distance.

Many circumstances require the use of path modification. For instance, if the potential collision region extended across the entire PSPCRD in either λ or γ, altering the motion characteristics will not avoid the collision. Specifically, the whole path of one robot is a

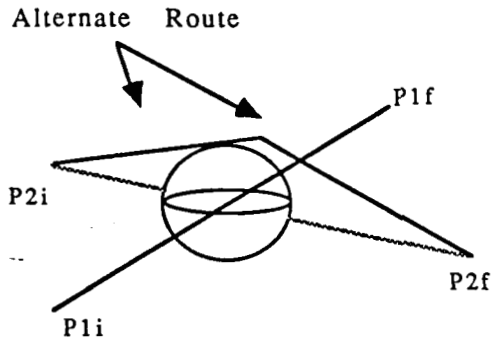


Figure 23 Path Modification

potential collision segment. Fig. 24 shows a case with such a situation. Another example requiring path modification to obtain collision-free motion is paths that are parallel with the robots traveling toward each other. In summary, path modification avoids a space-time collision by eliminating the possibility of potential collisions by altering the paths of one or both of the robots.

3.1.3. Avoidance Requirements

A collision-free motion is achieved based on the avoidance requirements which modify the trajectory and path parameters discussed in the previous sections. The avoidance requirements for two robots are classified into three categories:

Requirement 1) The final arrival times of one or both of the robots can be modified, but both robots must adhere to their original paths.

Requirement 2) The paths of one or both of the robots can be modified, but both must adhere to their final

arrival times.

Requirement 3) The paths and final arrival times of one or both of the robots can be modified.

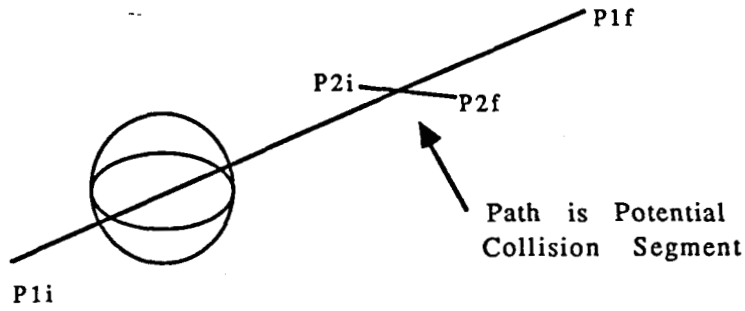


Figure 24 Situation Requiring Path Modification

In avoidance requirement one, the trajectory parameters are manipulated. Specifically, the motion characteristics of one or both of the robots are altered to avoid a collision. In avoidance requirement three, three cases exist: 1) modify one or both of the paths, 2) modify one or both of the final arrival times, and 3) modify one of the paths and one of the final arrival times. In case 1, only one of the paths should be modified since it is expensive in time to modify both paths. This case also handles the situation of a stationary robot. Case 2 is identical to avoidance requirement one and thus is solved by modifying the motion characteristics of one or both of the robots. In case 3, the motion characteristics of one robot are altered while the path of the other robot is modified. Since modifying the path of a robot requires a greater travel distance, a solution to avoidance requirement two may not exist.

3.2. Parameter Modification

The number of possible variations in trajectory parameters or robot paths to achieve collision-free motion is very high. An obvious way to optimize these modifications is the use of some criterion to obtain a best solution. One approach currently being investigated formulates collision avoidance as an optimization problem. For instance, a collision-free motion can be obtained by minimizing the delay in the final arrival times of both robots based on modifying the trajectory parameters of one or both of the robots under avoidance requirement one. In this case, a penalty can be given to each robot for the delay in their new final arrival times due to placement of break points, acceleration reduction, and/or time postponement. In a similar manner, collision-free motion can be obtained by minimizing the deviation of one or both of the robots from their original paths based on distance traveled. In this case, a penalty is given to the robots for an increase in travel distance due to path modification. Combinining the above two cases into one optimization problem can be performed to achieve collision avoidance under avoidance requirement three. In this situation, the optimization is performed by minimizing the delay in the final arrival times of both of the robots based on trajectory modification and/or minimizing the deviation from their original paths based on distance traveled. The optimization function is subject to various constraints on acceleration values, location of break points, etc.

Therefore, a collision-free motion is determined using the various methods of modifying the trajectory and path parameters that best fit an optimization function. In the above situations,

heuristics should be defined in order to limit the amount of search for a solution. Notice in this approach that a reduced space-time collision region is helpful in finding possible break point positions so that a separation in the overlapped time ranges will occur.

Another approach to collision avoidance is to fit the trajectory curve of a robot through a specified point to force a separation in the common time ranges. Fig. 25 shows the position of the point where the trajectory curve of robot 2 must pass in order to achieve collision-free motion. This method can incorporate break point modification, acceleration adjustment, and if necessary, time postponement. This approach can utilize a reduced space-time collision region and represents avoidance requirement one because only the trajectory parameters are being manipulated.

A final approach to collision avoidance is illustrated in Fig. 26. In this method, the PCRMD is utilized to produce collision-free motions. The idea is to restrict the values of λ and γ to be within certain ranges such that a motion curve can never pass through the potential collision region. One possibility is to modify the curve within the bounding box of the potential collision region. Therefore, modifications to the trajectroy curves are altered in the STCRD between the common time ranges. A second possibility is to redefine the whole motion of the robots such that the motion curve in the PCRMD bypasses the potential collision region. In this case, both robots will finish at the same time. In either of these two situations, the size of the space-time collision region is not of great significance since once a collision is detected, collision-free motion is achieved based only on avoiding the potential collision region. Altering the

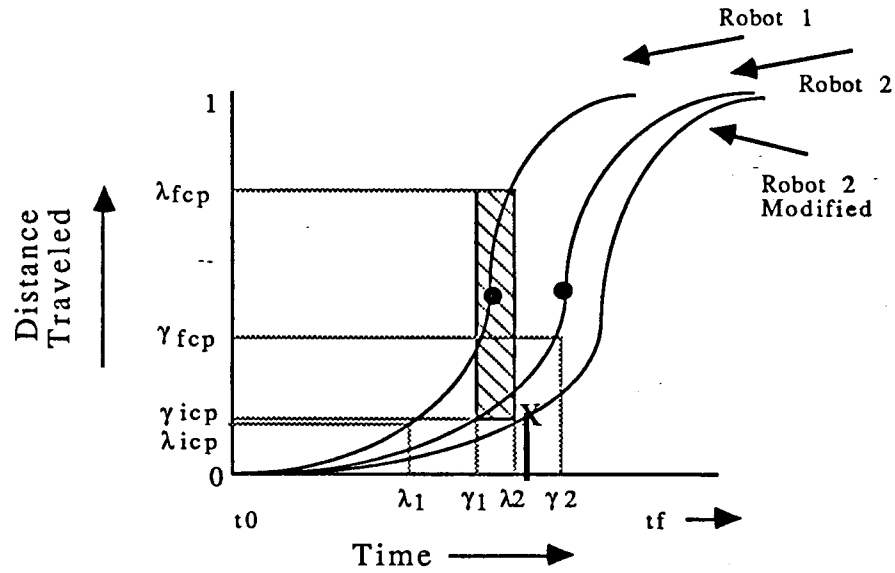


Figure 25 Fitting the Trajectory Curve

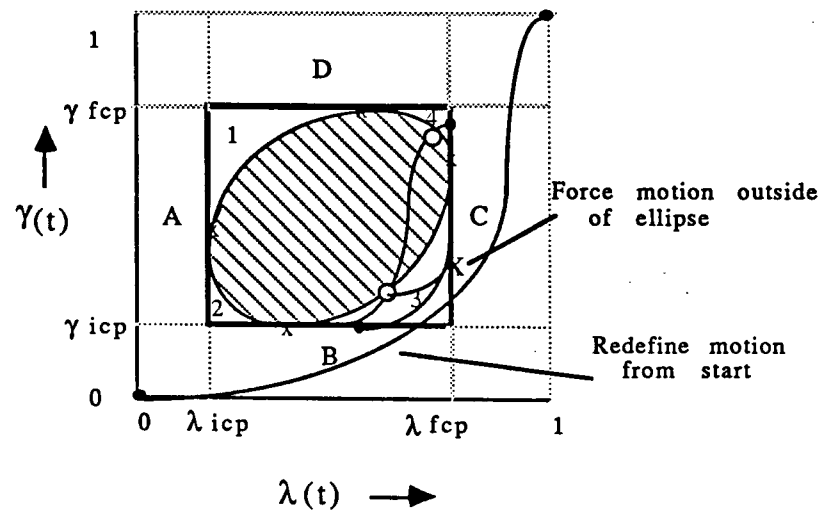


Figure 26 Motion Restriction

motion curve in the PCRMD has the effect of modifying the trajectory of both of the robots and thus provides collision avoidance under avoidance requirement one.

4. CONCLUSIONS AND FUTURE RESEARCH

A critical area of interest in robotics research involves coordination of multiple arms which is essential for the "intelligent robot" of the future. Independently controlled robots in a common area should be coordinated in order to avoid collisions between them. Therefore, motion planning must include detection and avoidance of collisions between two or more robots performing tasks in a common workspace.

Collision-free motion of two robot arms in a common workspace is investigated in this report. The collision-free motion is obtained by detecting collisions along straight line trajectories of each robot by using a sphere model for the wrists and then modifying the paths and/or trajectories of one or both robots to avoid the collision. The collision detection algorithm is described and suggested approaches to collision avoidance are outlined for future research.

The collision detection method obtains a range of potential collisions along the straight line trajectories of the two arms without considering the motion characteristics by producing a Parametric-Space-Potential-Collision-Region Diagram (PSPCRD). The potential collision range is then mapped into the time domain to obtain the space-time collisions by producing a Space-Time-Collision-Region Diagram (STCRD) and using an analysis domain consisting of a Potential-Collision-Region-Motion Diagram (PCRMD). Once the

collision region in time and space is found, a collision free motion is obtained by producing new paths and/or trajectories for the robots based on the avoidance requirement and various avoidance techniques.

In conclusion, this report presents a novel approach to collision-free motion planning of two robots operating in a common workspace. The efficiency of the collision detection algorithm allows for an on-line motion planner. Thus, this work provides a significant contribution towards multiple arm coordination.

In order to provide a complete solution to the multiple robot arm coordination problem, further development and investigation is needed. Future research direction will consider the following goals:

1) Develop the formal theory for the collision avoidance techniques using the sphere model for two robots.
2) Extend the techniques of detection and avoidance for other types of wrist models (cylinder, cone, polyhedra, etc.) for two arms.
3) Generalize the techniques of detection and avoidance of collisions to handle more than two robots in a common workspace.

REFERENCES

[1] William B. Gevarter, "Robotics: An Overview," *Computers in Mechanical Engineering,* August 1982, pp. 43-49.

[2] David Nitzan, "Development of Intelligent Robots: Achievements and Issues," *IEEE Journal of Robotics and Automation,* Vol. RA-1, No. 1, March 1985, pp. 3-13.

[3] Aaron Cohen and Jon. D. Erickson, "Future Uses of Machine Intelligence and Robotics for the Space Station and Implications for the U.S. Economy," *IEEE Journal of Robotics and Automation,* Vol. RA-1, No. 3, September 1985, pp. 117-123.

[4] John Roach, "Coordinating Multiple Robot Arms - Planning and Execution," *Proceedings of the International Conference on Advanced Automation - 1983,* December 1983, pp. 167-175.

[5] Tatsuzo Ishida, "Force Control in Coordination of Two Arms," *Proceedings of the 5th International Joint Conference on Artificial Intelligence,* August 1977, pp. 717-722.

[6] Cecil Alford and Stanley Belyen, "Coordinated Control of Two Robot Arms," *Proceedings of the International Conference on Robotics - 1984,* March 1984, pp. 468-473.

[7] T. J. Tarn, A. K. Bejczy, and X. Yun, "Coordinated Control of Two Robot Arms," *Proceedings of the IEEE International Conference on Robotics and Automation,* April 1986, pp. 1193-1202.

[8] Yuan F. Zheng and Fred R. Sias, "Two Robot Arms in Assembly," *Proceedings of the IEEE International Conference on Robotics and Automation,* April 1986, pp. 1230-1235.

[9] Samad Hayati, "Hybrid Position/Force Control of Multi-Arm Cooperating Robots," *Proceedings of the IEEE International Conference on Robotics and Automation,* April 1986, pp. 82-89.

[10] Shriram M. Udupa, "Collision Detection and Avoidance in Computer Controlled Manipulators," *Proceedings of the 5th International Joint Conference on Artificial Intelligence,* August 1977, pp. 737-748.

[11] Rodney A. Brooks, "Solving the Find-Path Problem by Good Representation of Free Space," *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. SMC-13, No. 3, March/April 1983, pp. 190-197.

[12] Tomas Lozano-Perez and Michael A. Wesley, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Communications of the ACM,* Vol. 22, No. 10, October 1979, pp. 560-570.

[13] Tomas Lozano-Perez, "Automatic Planning of Manipulator Transfer Movements," *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. SMC-11, No. 10, October 1981, pp. 681-698.

[14] Tomas Lozano-Perez, "Spatial Planning: A Configuration Space Approach," *IEEE Transactions on Computers,* Vol. C-32, No. 2, February 1983, pp. 108-120.

[15] R. T. Chien, Ling Zhang, and Bo Zhang, "Planning Collision-Free Paths for Robotic Arm Among Obstacles," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-6, No. 1, January 1984, pp. 91-96.

[16] W. E. Red and Hung-Viet Truong-Cao, "Configuration Maps for Robot Path Planning in Two Dimensions," *Journal of Dynamic Systems, Measurement, and Control,* Vol. 107, December 1985, pp. 292-298.

[17] Elmer G. Gilbert and Daniel W. Johnson, "Distance Functions and Their Application to Robot Path Planning in the Presence of Obstacles," *IEEE Journal of Robotics and Automation,* Vol.

RA-1, No. 1, March 1985, pp. 21-30.

[18] Subbarao Kambhampati and Larry S. Davis, "Multiresolution Path Planning for Mobile Robots," *IEEE Journal of Robotics and Automation,* Vol. RA-2, No. 3, September 1986, pp. 135-145.

[19] E. K. Wong and K. S. Fu, "A Hierarchical Orthogonal Space Approach to Three-Dimensional Path Planning," *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, March 1986, pp. 42-53.

[20] B. John Oommen and Irwin Reichstein, "On Translating Ellipses Amidst Elliptic Obstacles," *Proceedings of the IEEE International Conference on Robotics and Automation,* April 1986, pp. 1755-1760.

[21] S. Dubowsky, M. A. Norris, and Z. Shiller, "Time Optimal Trajectory Planning for Robotic Manipulators with Obstacle Avoidance: A CAD Approach," *Proceedings of the IEEE International Conference on Robotics and Automation,* April 1986, pp. 1906-1912.

[22] J. Sanjiv Singh and Meghanad D. Wagh, "Robot Path Planning using Intersecting Convex Shapes: Analysis and Simulation," *IEEE Journal of Robotics and Automation,* Vol. RA-3, No. 2, April 1987, pp. 101-108.

[23] Kurt D. Rueb and Andrew K. C. Wong, "Structuring Free Space as a Hypergraph for Roving Robot Path Planning and Navigation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-9, No. 2, March 1987, pp. 263-273.

[24] Laurent Gouzenes, "Collision Avoidance for Robots in an Experimental Flexible Assembly Cell," *Proceedings of the IEEE International Conference on Robotics and Automation,* March 1984, pp. 474-476.

[25] V. DuPourque, H. Guiot, and O. Ishacian, "Towards Multi-Processor and Multi-Robot Controllers," *Proceedings of the IEEE International Conference on Robotics and Automation,*

April 1986, pp. 864-870.

[26] E. Freund and H. Hoyer, "Hierarchical control of Guided Collision Avoidance for Robots in Automatic Assembly," *Proceedings of the 4th International Conference on Assembly Automation,* 1983, pp. 91-103.

[27] E. Freund, "On the Design of Multi-Robot Systems," *Proceedings of the IEEE International Conference on Robotics and Automation,* March 1984, pp. 477-490.

[28] E. Freund and H. Hoyer, "Pathfinding in Multi-Robot Systems: Solutions and Applications," *Proceedings of the IEEE International Conference on Robotics and Automation,* April 1986, pp. 103-111.

[29] John Canny, "Collision Detection for Moving Polyhedra," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-8, No. 2, March 1986, pp. 200-209.

[30] Pierre Tournassoud, "A Strategy for Obstacle Avoidance and Its Application to Multi-Robot Systems," *Proceedings of the IEEE International Conference on Robotics and Automation,* April 1986, pp. 1224-1229.

[31] Steven Fortune, Gordon Wilfong, and Chee Yap, "Coordinated Motion of Two Robot Arms," *Proceedings of the IEEE International Conference on Robotics and Automation,* April 1986, pp. 1216-1223.

[32] R. A. Basta, "Multiple Arm Coordination Using Concurrent Processing," *Proceedings: IEEE Southeastcon'87,* April 1987, pp. 328-331.

[33] J. Roach and M. Boaz, "Coordinating the Motions of Robot Arms in a Common Workspace," *Proceedings of the IEEE International Conference on Robotics and Automation,* March 1985, pp. 494-499.

[34] B. H. Lee and C. S. G. Lee, "Collision-Free Motion Planning of Two Robots," *IEEE Transactions on Systems, Man, and*

Cybernetics, Vol. SMC-17, No. 1, January/February 1987, pp. 21-32.

REMOVED

APPENDIX-2

APPENDIX-3

CSE-87-00004

A RUDIMENTARY DATABASE FOR THREE-DIMENSIONAL OBJECTS USING STRUCTURAL REPRESENTATION

James P. Sowers

A RUDIMENTARY DATABASE FOR THREE-DIMENSIONAL OBJECTS USING STRUCTURAL REPRESENTATION*

James P. Sowers

Computer Science and Engineering Department
University of South Florida
Tampa, Florida

* This project was supported by NASA-Langley grants NAG-1-772 and NAG-1-632.

I. INTRODUCTION

A database which enables users to store and share the description of three-dimensional objects in a research environment is presented. The main objective of the design is to make it a compact structure that holds sufficient information to reconstruct the object. The database design is based on an object representaion scheme which is information preserving, reasonably efficient and yet economical in terms of the storage requirement. The determination of the needed data for the reconstruction process is guided by the belief that it is faster to do simple computations to generate needed data/information for the construction than to retrieve everything from memory.

The next section discusses some recent techniques of three-dimensional representation that influenced the design of the database. Section III gives the schema for the database and the structural definition used to define an object. Section IV contains the user manual for the software developed to create and maintain the contents of the database.

II. BACKGROUND

Most of the three-dimensional object representation schemes [1-4] can be classified into three major categories: surface or boundary, sweep, and volumetric, where a representation scheme is a formal system for descibing shape or some aspect of shape along with rules that specify how that scheme is to be applied to a shape. The description resulting from applying the scheme to a given shape is the representation in that scheme. When deciding on a representation scheme for modeling objects the following properties should be considered [3,5]:

domain - a clearly defined descriptive power for the scheme,

validity - the representation for an object is in the range of

representations for the scheme,

unambiguity - the representation corresponds to a single object in the range of valid representations,

uniqueness - the ability to easily assess the equality of two objects in that for a given object a single representation is formed,

consistency - the same representation is always produced for a given object and scheme,

conciseness - the size, verbosity, or redundancy of the representation,

ease of creation - the ease with which valid representations may be created with the modeling system, and

efficacy for application - the representation is condusive to good, efficient algorithms for computing useful functions.

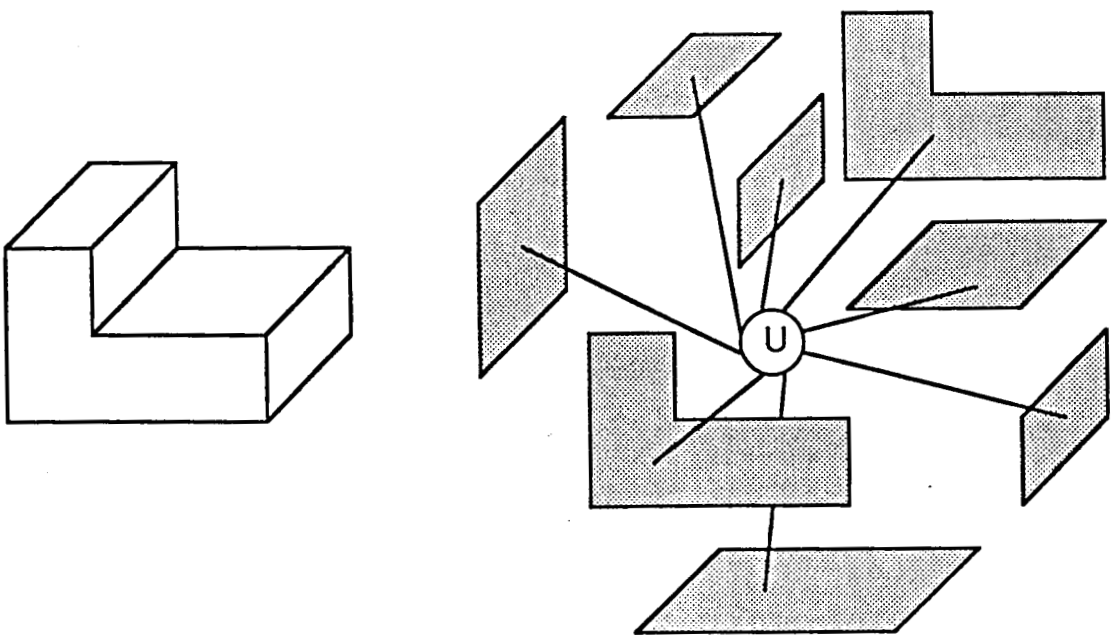


Figure 1. Surface Representation

Surface or boundary representation of an object [6,7] is represented by a set of "faces" or "patches" that are bound

together by a set of rules to make up the object [see figure 1]. Some current approaches include Coons patches [8], bicubic surface patches, Bezier methods [9], and B-splines. Even though boundary representation is unambiguous, it is not unique and the validity is not guaranteed.

Sweep representation of an object is represented by a two-dimensional set that is translated along a line. Two common methods are generalized cylinders [10,11] and symmetric axis transform, also known as the medial axis transform, which was introduced by Blum [12]. Nackman and Pizer present a theory to expand the symmetric axis transform to three dimensions in [13]. The definition of the two-dimensional symmetric axis transform also applies in three dimensions, except that maximal disks become maximal spheres and the symmetric axis becomes the symmetric surface, see figure 2. Generalized cylinders (generalized cones), is analogous to symmetric axis transform in three dimensions. A generalized cylinder is a solid whose axis is a three-dimensional space curve, and its cross sections are orthogonal to its axis, see figure 3. Also, the two-dimensional set defining the generalized cylinder may be allowed to rotate about the axis, while it is translated along the axis.

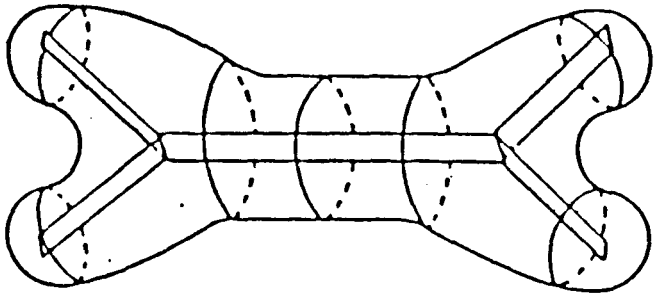


Figure 2. Example of 3-D Symmetric Surface

Sweep representation works well with manmade objects that have an axis of symmetry. It is concise, but in general, it is not

unique.

Volumetric representation of objects [3,14-17] is accomplished by representing an object in terms of more primitive solids. The three representations are: 1) spatial occupancy - values are represented as a three-dimensional array of cells which may be marked as filled or not with matter [15], 2) cell decomposition - cells are more complex in shape but still do not share volumes, so the only combining operation is "glue" [16,17], and 3) constructive solid geometry - complex solids are represented as various ordered operations of simpler objects (primitives), by means of psuedo-Boolean set operations. The primitives used could be simple geometric solids such as prisms, cylinders, ellipsoids, and boxels [18] to more complicated primitives such as superquadrics [19].

Volumetric representation is adequate to comprise most conventional, unsculptured objects and is unambiguous but is not unique.

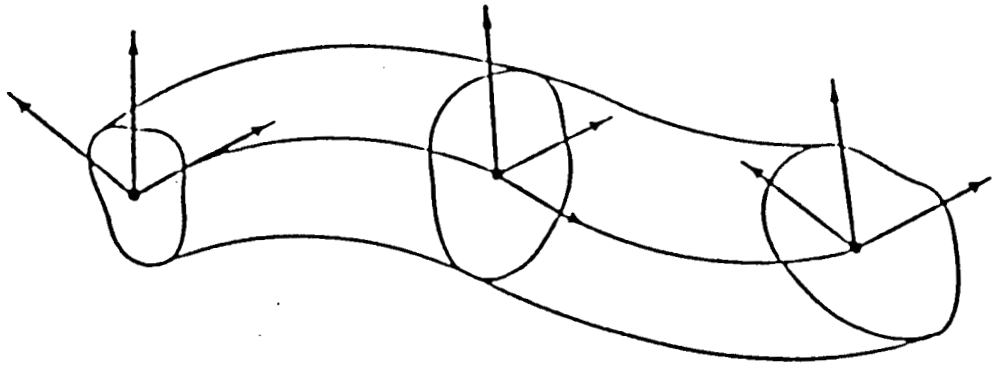


Figure 3. A Generalized Cylinder with some Cross-Sectional Coordinate Systems

III. DATABASE DESIGN

A general model, similar to the one described in [20], is used for the object description in the database, since it allows for the

ability to save properties about the object, such as global attributes and parameters. An object is defined as a 6-tuple O={C, N, A, P, R, PA}, where C is the class of the object, which defines a set of similar objects, N is the name of the object, A is the set of attributes for the object, P is the set of primitives used to make up the object, R is the set of relationship functions used to describe the associations between primitives in set P, and PA is the set of parameters. Parameters in this structure are basic characteristics about the object, such as material composition.

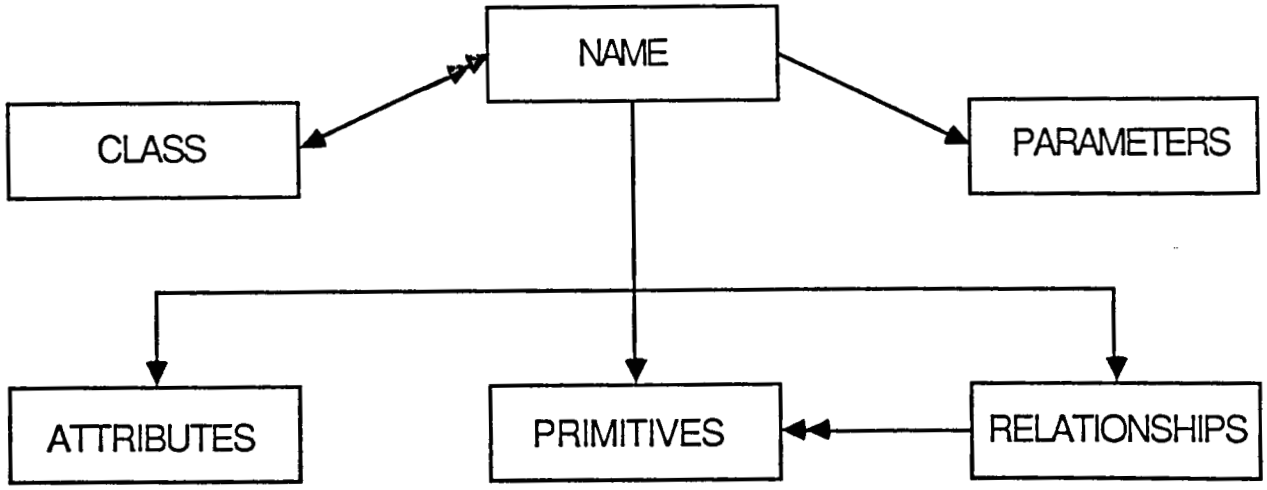


Figure 4. Database Schema

Figure 4 shows the database schema, with the notation borrowed from Martin [21]. The database file structure consists of a main file and a set of attribute, primitive, and relationship files for the objects in the database. For every object in the database a record that contains the name, class, parameters, and pointers to the attribute, primitive, and relationship files for the object is contained in the main file. The attribute file for an object consists of records that contain the attribute name and quantity.

The primitive file for an object consists of records that contain id tag, primitive id type, and three parameters, discussed further in Section III.A. The relationship file for an object consists of records that contain the operator, the two primitive id tags, and six parameters defining connection points and orientation, discussed further in Section III.B.

III.A PRIMITIVES

The present set of volumetric primitives consist of a four-sided prism, a right-wedge, a left-wedge, an ellipsoid, a cylinder, and a cone, which is shown in the appendix. The primitives as shown in the appendix give the "home" position for each primitive, and the format for the type id and the three parameters. The three parameters simply give the overall width, depth, and height of the primitive.

A record in the primitive file contains the id tag, primitive id type, and the three parameters. This information is used to determine the configuration of the primitive. The id tag is an unique integer value to distinguish the primitive from the others in the set. The primitive id type is used to classify the primary shape of the primitive and the three parameter values give the dimension for the final shape of the primitive. For example, if the id type is a cylinder and the values of the parameters are X=5, Z=1, and D=3, the shape of the primitive would be as shown in figure 5.

III.B RELATIONSHIP FUNCTIONS

The relationship describes how the primitives relate to each other so that the object can be reconstructed. The relational operations used are analogous to the ones employed by constructive solid geometry [22], hence the same limitations as mentioned under volumetric representation apply. A record in the relationship file

contains an operator, two ids, two connection points, and two rotational values. The general format for the relationship is as follows:

$$(OP, ID_i, ID_j, X_i, Y_i, Z_i, X_j, Y_j, Z_j, Rx_i, Ry_i, Rz_i, Rx_j, Ry_j, Rz_j)$$

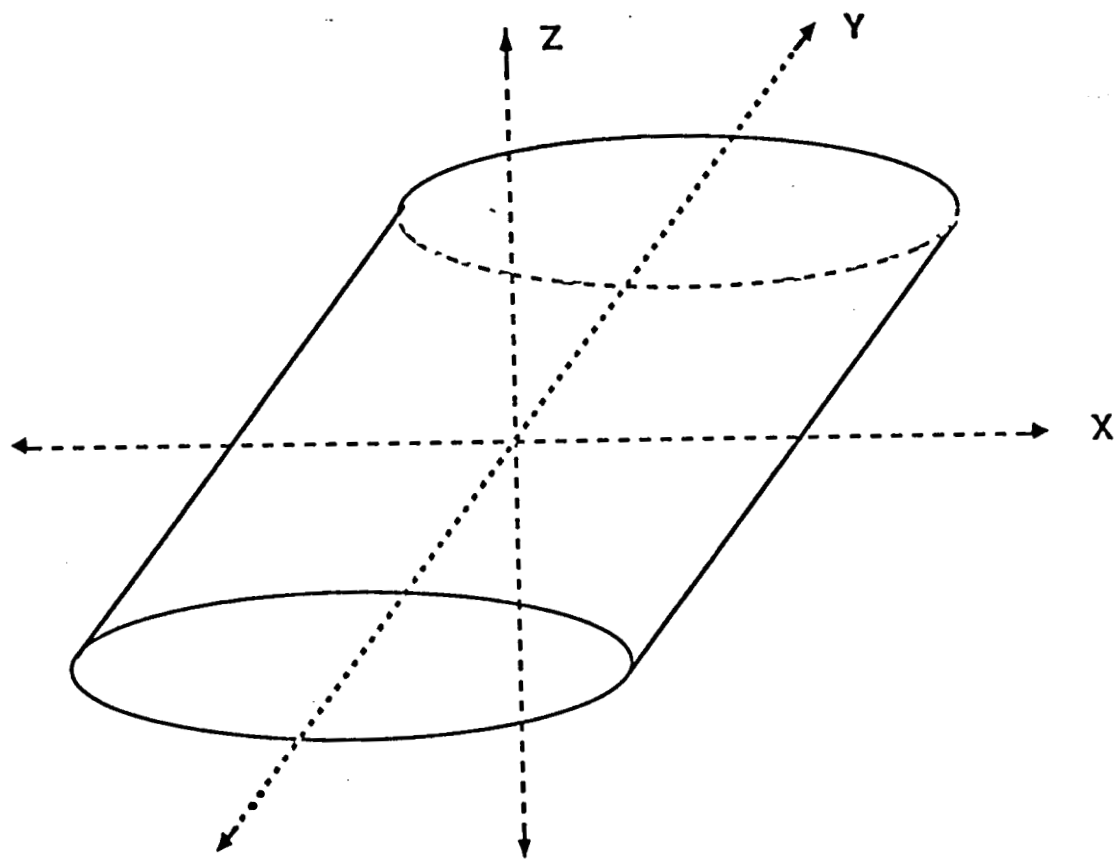


Figure 5. Cylinder with X=5, Z=1, and D=3 in home position

Where ID_i and ID_j are set to id tag values to identify which primitives, from the set of primitives, are to be used for the operation. The (X_i, Y_i, Z_i) and (X_j, Y_j, Z_j) points are values to describe where the two primitives will be connected. The valid range of values, relative to the "home" position and specific primitive type are defind as $-P_i/2 <= V_i <= P_i/2$; P_i is the

parameter value for the primitive in the i^{th} position, where i=1,2, or 3, and V_i is the value for the i^{th} position. (Rx_i, Ry_i, Rz_i) and (Rx_j, Ry_j, Rz_j) are the angles that the primitives are rotated after being connected; the rotation is about the connection point.

Union, intersection, and subtraction are the three operators employed to operate on the primitives. The operations, with their present restrictions, are defined below.

union operation:

$(+, ID_i, ID_j, X_i, Y_i, Z_i, X_j, Y_j, Z_j, 0, 0, 0, Rx_j, Ry_j, Rz_j)$

This is the union operation which connects the surface point (X_j, Y_j, Z_j) of primitive j to the surface point (X_i, Y_i, Z_i) of primitive i, with primitive j being rotated about its point of connection by (Rx_j, Ry_j, Rz_j) and the rotation angles for primitive i are set to zero.

intersection operation:

$(*, ID_i, ID_j, X_i, Y_i, Z_i, 0, 0, 0, 0, 0, 0, Rx_j, Ry_j, Rz_j)$

This is the intersection operation which places primitive j in primitive i, with the origin point of primitive j being located in primitive i at point (X_i, Y_i, Z_i) and primitive j being rotated about its connection point (origin) by (Rx_j, Ry_j, Rz_j). The rotation angles for primitive i are set to zero.

subtraction operation:

$(-, ID_i, ID_j, X_i, Y_i, Z_i, 0, 0, 0, 0, 0, 0, Rx_j, Ry_j, Rz_j)$

This is the subtraction operation which is identical to the intersection operation except primitive j is removed from primitive i.

The next section contains the user manual for the database that has just been described.

IV OPERATIONS ON THE DATABASE

IV.A Entering an object

Name:

After the main menu appears and selection of the CREATE OBJECT option, the following prompt will appear:

Please enter object name:

Enter any string up to 80 characters in length, but only the first 15 significant characters are kept, and the string must contain at least one non-blank character. Leading and trailing blanks are stript, however embedded blanks are not stript from the string. If an invalid name is entered the user will be prompted again for another name. Also if object already appears in the database the user will be asked for another name. The following examples show valid and invalid names.

Example 1.

VALID ENTRY	
NAME	COMMENT
1	one non-blank
A	one non-blank
A1C$	any character legal
bbbALPHAbbb	leading and trailing blanks ignored
ALPHA	recognized same as above
Alpha	different by case of lettering
abcd1234efgh5678ijk	only a through 7 is considered
good name	embedded blank left in name
	INVALID ENTRY
<cr>	null string
bbbbbbb	non-blank rule broke

Class:

The following prompt will appear for entry of object class:

Please enter object class:

The criteria for a valid class name is the same as for a valid object name. If an invalid class is entered the user will be prompted again for entry of class name.

Parameters:

Currently the parameters for an object have not been fully defined and the only parameter presently being looked for is the object's material composition. The following prompt will appear:

Please enter primary material composition of object:

The criteria for a valid material entry is the same as for a valid object name. If an invalid material entry is made the user will be prompted again for entry of material.

Attributes:

The present structure for object attributes is for name and quantity of attribute. One or more attributes can be entered fo any one object. The following prompt will appear:

Please enter attribute:

The criteria for a valid attribute entry is the same as for a valid object name. If an invalid attribute entry is made the user will be prompted again for entry of an attribute. After valid attribute is entered, the quantity (integer value) will be asked for and the following prompt will appear:

Enter quantity of this attribute:

No error correction is offered. After entry of quantity the user will be prompted, by the following, for continuation:

Enter another attribute (y/n)?

To continue entering attributes, enter either y or Y. To sto entering attributes, enter either n or N, if any other character is entered, the default is to stop.

Primitives:

There are presently six primitives available for composing an object as seen in appendix A. The following menu will appea for entering primitives comprising the object:

	PRIMITIVES	QTY
1)	Four-sided prism	#
2)	Right-wedge	#
3)	Left-wedge	#
4)	Ellipsoid	#
5)	Cylinder	#
6)	Cone	#
0)	Quit	

Choice:

The QTY column reminds the user of the number of primitives entered for each type. Depending on which primitive is selected the user will be prompted to enter the following dimensions fo that primitives: enter the X-axis diameter or length, the Z-axis diameter or height, and the Y-axis diameter or depth of the primitive. The values expected for each is a real number, n error correction is offered. The 0 option is used to qui entering primitives.

Relationship:

The relationships describe how the primitives entered above relate to each other so that the object can be reconstructed. The format for the relationship is as follows:

$$(OP, PRIM_i, PRIM_j, X_i, Y_i, Z_i, X_j, Y_j, Z_j, Rx_i, Ry_i, Rz_i, Rx_j, Ry_j, Rz_j)$$

union operator:
$(+,PRIM_i,PRIM_j,X_i,Y_i,Z_i,X_j,Y_j,Z_j,0,0,0,Rx_j,Ry_j,Rz_j)$

This is the union operator which connects the surface point (X_j,Y_j,Z_j) of primitive j to the surface point (X_i,Y_i,Z_i) of primitive i, with primitive j being rotated about its point of connection by (Rx_j,Ry_j,Rz_j).

intersection operator:
$(*,PRIM_i,PRIM_j,X_i,Y_i,Z_i,0,0,0,0,0,0,Rx_j,Ry_j,Rz_j)$

This is the intersection operator which places primitive j in primitive i, with the origin point of primitive j being located in primitive i at point (X_i,Y_i,Z_i) and primitive j being rotated about its origin point by (Rx_j,Ry_j,Rz_j).

subtraction operator:
$(-,PRIM_i,PRIM_j,X_i,Y_i,Z_i,0,0,0,0,0,0,Rx_j,Ry_j,Rz_j)$

This is the subtraction operator which is identical to the intersection operator except primitive j is removed from primitive i.

For entry of relationship the following prompt will appear:

Number of operations entered: #

Enter first primitive tag id:

The first prompt reminds the user of the number of relationship operations entered so far. The second one is looking for the tag id (integer value) for the first primitive involved. No error correction is offered. The next prompt to appear is:

Enter second primitive tag id:

Here enter the tag id (integer value) for the second primitive involved. No error correction is offered. The next prompt to appear is:

Enter desired operator (+,-,*):

Depending on the operator entered, the user will be prompted to enter the appropriate (real) values for the connection points and orientations. If the user does not want to enter a relationship, enter a character other than the operators and the current operation will be ignored. After entry of an operation the following prompt will appear:

Enter another operation (y/n)?

To continue entering relationships, enter either y or Y. To stop entering relationships, enter either n or N, if any other character is entered, the default is to stop.

IV.B Deleting an object

After selecting the DELETE OBJECT option in the main menu, the following prompt will appear:

Please enter object name:

Enter any string up to 80 characters in length, but only the first 15 significant characters are kept, and the string must contain at least one non-blank character. Leading and trailing blanks are stript, however embedded blanks are not stript from the string. If an invalid name is entered the user will be prompted again for another name.

If name entered exist it will be removed from the database along with all other files or information relating to it, otherwise an error message will appear stating no object with that name presently exist in the database.

IV.C Viewing the database

LIST OBJECTS - This option for viewing the database list the objects by name, a page at a time. To stop viewing the database, enter either n or N to the inquiry about continuing.

LIST CLASSES - This option for viewing the database list the classes and the objects associated with each class, a page at a time. To stop viewing the database, enter either n or N to the inquiry about continuing.

IV.D Changing an object

After selecting the UPDATE OBJECT option in the main menu the following prompt will appear:

Please enter object name:

Enter any string up to 80 characters in length, but only the first 15 significant characters are kept, and the string must contain at least one non-blank character. Leading and trailing blanks are stript, however embedded blanks are not stript from the string. If an invalid name is entered the user will be prompted again for another name.

Name change:
If the object exist in the database the following prompts will appear:

Object's name: <Hopefully the one the user entered>

Change object's name (y/n)?

To change name, enter either y or Y, then the following prompt will appear:

Please enter object name:

The criteria for name is the same as for entry of name for change. If an invalid name is entered the user will be prompted again for another name.

Class change:
For changing the class, the following prompt will appear:

Object's class: <class for object>

Change object's class (y/n)?

To change the class of the object, enter either y or Y, then the following prompt will appear:

Please enter object class:

The criteria for a valid class entry is the same as for a valid object name. If an invalid class is entered the user will be prompted again for a class.

Parameter change:
The present format for parameters is undefined and the only thing contained in this structure is the primary material composition of the object. The following prompt will appear for changing the material:

Object's material: <material of object>

Change object's material (y/n)?

To change the material, enter either y or Y, then the following prompt will appear:

Please enter primary material composition of object:

The criteria for a valid material entry is the same as for a valid object name. If an invalid material is entered the user will be prompted again for a material.

Attribute change:
For changing the attributes of the object, the following prompt will appear:

Change object's attributes (y/n)?

To change attributes, enter either y or Y, then the following prompt will appear:

Attribute is as follows:
name: <attribute>
quantity: #

Do you want to C)hange
D)elete
or get N)ext attribute

Change:
To change this attribute of the object, enter either c or C, the following prompt will appear:

Please enter attribute:

The criteria for an attribute is the same as for a valid object name. If an invalid attribute is entered the user will be prompted again for another attribute. Then the user will be asked for the quantity of this attribute by the following prompt:

Enter quantity of this attribute:

An integer value is being looked for and no error checking is offered. If only the quantity is desired to be changed, select the C)hange option and reenter the attribute then when asked for quantity enter the change.

Delete:
To delete the currently displayed attribute, enter either d or D, no prompt will appear for this option.

Next:
If no action is desired for the currently displayed attribute, to retrieve the next attribute, enter either n or N. No prompt will appear for this option. This allows a way to review the attributes associated with an object, without changing them.

Add:
After current attribute list is reviewed then you have the option to append more attributes to the list. The following prompt will appear for additions:

Add new attribute (y/n)?

To add another attribute, enter either y or Y, and the user will be prompted for attribute and quantity in the same manner as for changing attribute. Entering either n or N will stop changes to attribute list, also if any other character is entered the process will stop.

Primitive change:
For changing the primitives of the object, the following prompt will appear:

Change object's primitives (y/n)?

To change primitives, enter either y or Y, then the following prompt will appear:

```
Primitive is as follows:
                    Tag: #
                     Id: <primitive code>
          Length/X-axis: #
           Depth/Y-axis: #
          Height/Z-axis: #

Do you want to D)elete
        or get N)ext primitive
```

Delete:
To delete the currently displayed primitive, enter either d or D, no prompt will appear for this option.

Next:
If no action is desired for the currently displayed primitive, to retrieve the next primitive, enter either n or N. No prompt will appear for this option. This allows a way to review the primitives associated with an object, without changing them.

Add:
After current primitive list is reviewed then you have the option to append more primitives to the list. The following prompt will appear for additions:

```
Add new primitive (y/n)?
```

To add another primitive, enter either y or Y, and the user will be prompted as follows:

```
          PRIMITIVES
          ~~~~~~~~~~

      1) Four-sided prism
      2) Right-wedge
      3) Left-wedge
      4) Ellipsoid
      5) Cylinder
      6) Cone
      0) No creation of primitive


Choice:
```

Depending on which primitive is selected, the user will be prompted to enter the following dimensions for that primitives: enter the X-axis diameter or length, the Z-axis diameter or height, and the Y-axis diameter or depth of the primitive. The values expected for each is a real number, no error correction is offered. The 0 option is used to quit entering primitives.

Relationship change:
For changing the relationships of the object, the following prompt will appear:

Change object's relationships (y/n)?

To change relationships, enter either y or Y, then the following prompt will appear:

```
Relationship is as follows:
                    First primitive id: <tag #>
                   Second primitive id: <tag #>
                              Operator: <*,-,+>
 Connection point for first id(x,y,z): #,#,#

          Rotation of first id(x,y,z): #,#,#

Connection point for second id(x,y,z): #,#,#

         Rotation of second id(x,y,z): #,#,#

Do you want to D)elete
       or get N)ext relationship
```

Delete:
To delete the currently displayed relationship, enter either d or D, no prompt will appear for this option.

Next:
If no action is desired for the currently displayed relationship, to retrieve the next relationship, enter either n or N. No prompt will appear for this option. This allows a way to review the relationships associated with an object, without changing them.

Add:
After current relationship list is reviewed then you have the option to append more relationships to the list. The following prompt will appear for additions:

Add new relationship (y/n)?

To add another relationship, enter either y or Y, and the user will be prompted as follows:

Enter first primitive id:

Enter the tag id (integer value) for the first primitive involved. No error correction is offered. The next prompt to appear is:

Enter second primitive id:

Here enter the tag id (integer value) for the second primitive involved. No error correction is offered. The next prompt to appear is:

Enter desired operator (+,-,*):

Depending on the operator entered, the user will be prompted to enter the appropriate (real) values for the connection points and orientations. If the user does not want to enter a relationship, enter a character other than the operators and the current operation will be ignored.

IV.E Main menu

The following is the group of operations on the database, which is described in section IV.

1) Create object
2) Delete object
3) List objects
4) List classes
5) Update object
0) Quit

Choice:

ACKNOWLEDGEMENTS

Support for this project, from NASA-Langley Research Center grants NAG-1-772 and NAG-1-632, is gratefully acknowledged. Numerous discussions with and encouragement from the research group (COVIRT*) members at USF, Mike Goode and Karin Cornils of NASA-Langley Research Center is also acknowledged.

* COVIRT - COmputer Vision and Intelligent Robotics research Team.

REFERENCES

[1] J.K. Aggarwal, L.S. Davis, W.N. Martin, and J.W. Roach, "Survey: Representation Methods for Three-dimensional Objects," in *Progress in Pattern Recognition*, L.K. Kanal and A. Rosenfeld, Eds. North-Holland, 1981, pp. 377-391.

[2] N. Badler and R. Bajcsy, "Three-dimensional Representations for Computer Graphics and Computer Vision," *Computer Graphics*, vol. 12, pp. 153-160, August 1978.

[3] A.A.G. Requicha, "Representations for Rigid Solids: Theory, Methods, and Systems," *Computing Surveys*, vol. 12, pp. 437-464, December 1980.

[4] T.C. Henderson, "Efficient 3-D Object Representations for Industrial Vision Systems," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. PAMI-5, pp. 609-618, November 1983.

[5] C.M. Brown, "Some Mathematical and Representational Aspects of Solid Modeling," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol PAMI-3, pp. 444-453, July 1981.

[6] A.R. Forrest, "On Coons and Other Methods for the Representation of Curved Surfaces," *Comput. Graphics Image Processing*, vol. 1, pp. 341-359, 1972.

[7] R.E. Barnhill and R.F. Risenfeld, "Surface Representation for Computer Aided Design," in *Data Structures, Computer Graphics and Pattern Recognition*, A. Klinger, K.S. Fu, and T.L. Kunii, Eds. New York: Academic, 1977.

[8] S. Coons, "Surfaces for Computer-aided Design of Space Forms," M.I.T. Project MAC, MAC-TR-41, 1967.

[9] P. Bezier, "Mathematical and Practical Possibilities of UNISURF," in *Computer Aided Geometric Design*, R. Barnhill and R. Riesenfeld, Eds., New York: Academic Press, 1974.

[10] T. Binford, "Visual Perception by Computer," Invited paper, *IEEE Systems Science and Cybernetics Conference*, Miami, December 1971.

[11] G. Agin, "Representation and Description of Curved Objects," Ph.D. Thesis, Stanford A.I. Memo, AIM-173, October 1972.

[12] H. Blum, "A Transformation for Extracting New Descriptors of Shape," in *Models for the Perception of Speech and Visual Form*, W. Wathen-Dunn, Ed., Cambridge, MA: MIT Press, 1967, pp. 362-380.

[13] L.R. Nackman and S.M. Pizer, "Three-dimensional Shape Description Using the Symmetric Axis Transform I: Theory," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. PAMI-7, pp. 187-202, March 1985.

[14] A.A.G. Requicha and H.B. Voelcker, "Solid Modeling: A Historical Summary and Contemporary Assessment," *IEEE Comput. Graphics Applications*, pp. 9-24, March 1982.

[15] L. March and P. Steadman, *The Geometry of Environment*, Cambridge, MA: MIT Press, 1974.

[16] C.L. Jackins and S.L. Tanimato, "Oct-trees and Their Use in Representing Three-dimensional Objects," *Comput. Graphics Image Processing*, vol. 14, pp. 249-270, Nov. 1980.

[17] J.L. Bentley, "Multidimensional Search Trees Used for Associative Searching," *Commun. Assoc. Comput. Mach.*, vol. 18, pp. 509-517, September 1975.

[18] E.P. Krotkov, "Thesis Proposal: Active Visual Perception for Determining Spatial Layout," Thesis, Dept. Computer and Information Science, Univ. of Penn., Spring 1987.

[19] R. Bajcsy and F. Solina, "Three-dimensional Object Representation Revisited," Tech. Report MS-CIS-87-19, Dept. Computer and Information Science, Univ. of Penn., March 1987.

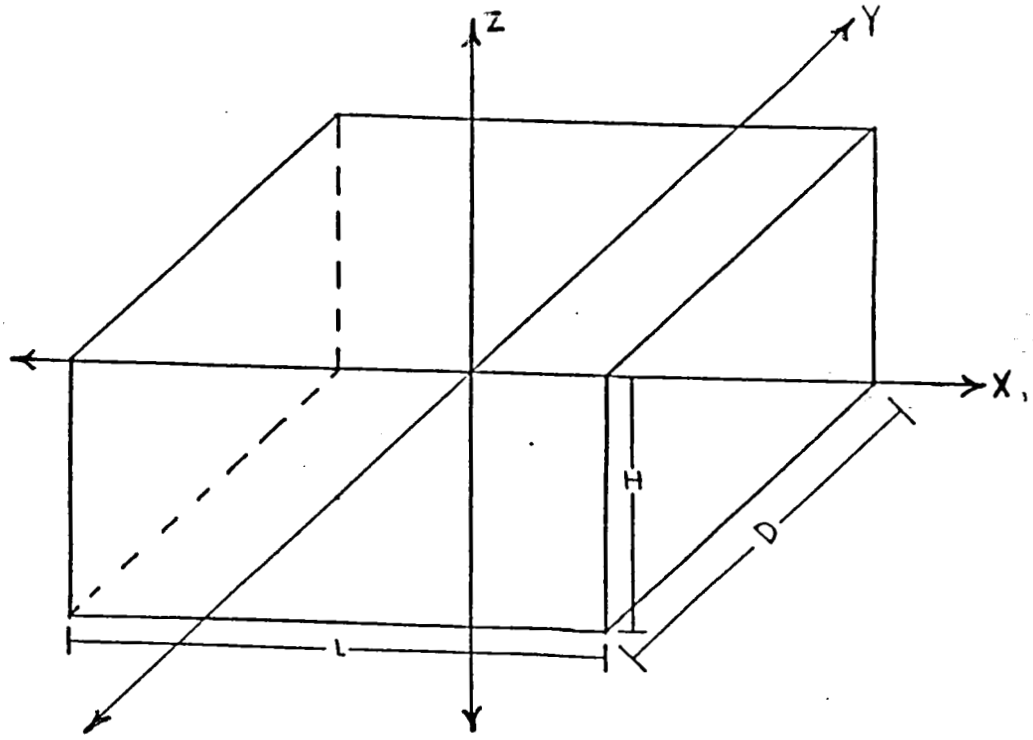
[20] L.G. Shapiro, "A Structural Model of Shape," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. PAMI-2, pp. 111-126, March 1980.

[21] J. Martin, "Computer Data-Base Organization," Series in Automatic Computation, Prentice-Hall, 1975.

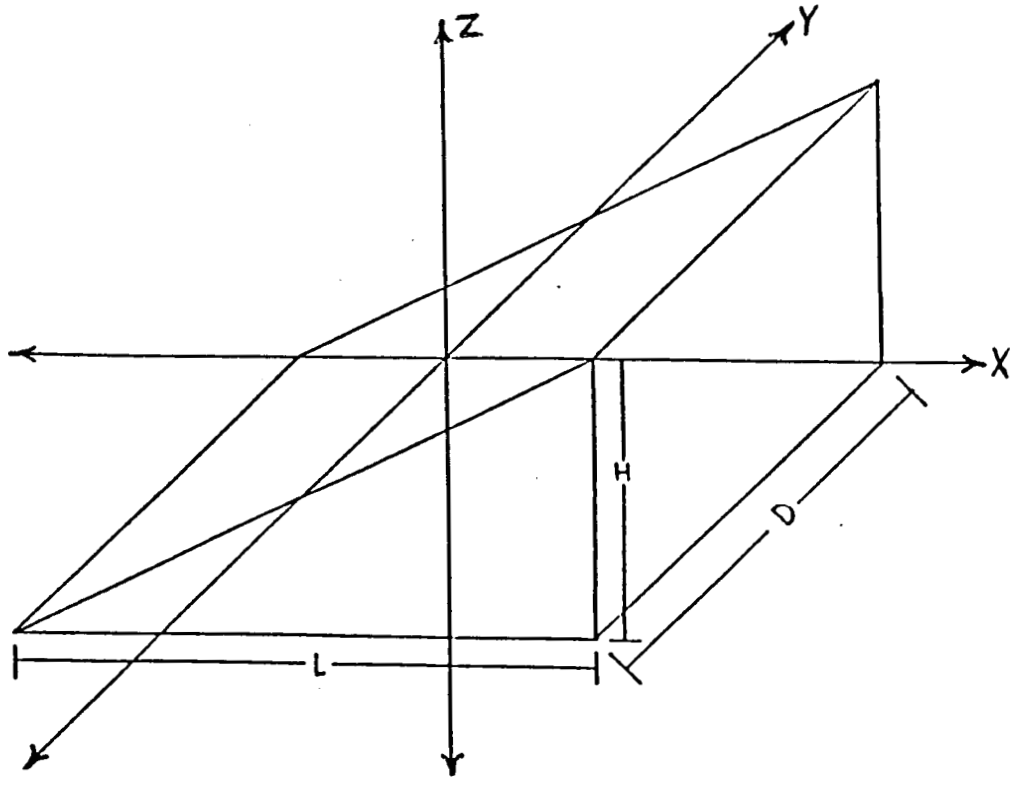
[22] A.A.G. Requicha and H.B. Voelcker, "Constructive Solid Geometry," Tech. Memo 25, Production Automation Project, Univ. Rochester, Rochester, N.Y., November 1977.

C-2

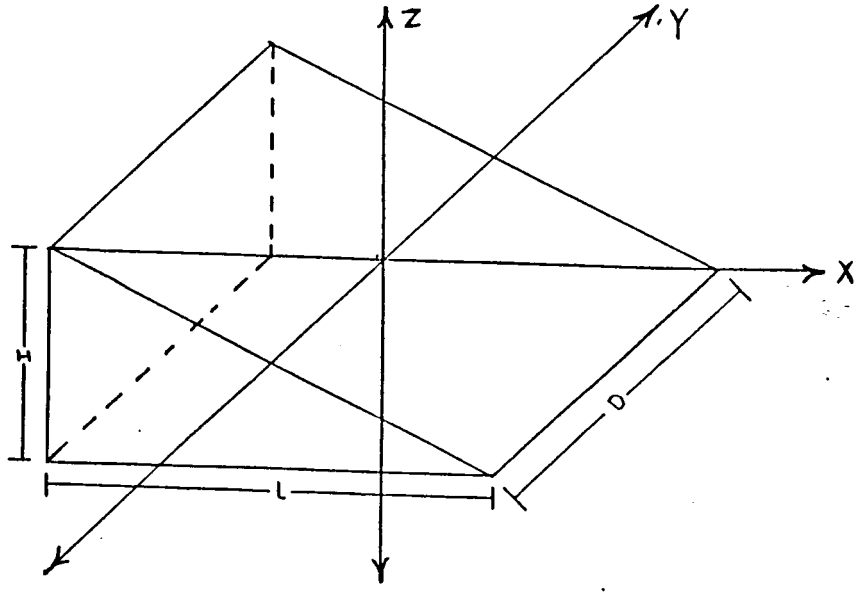
APPENDIX



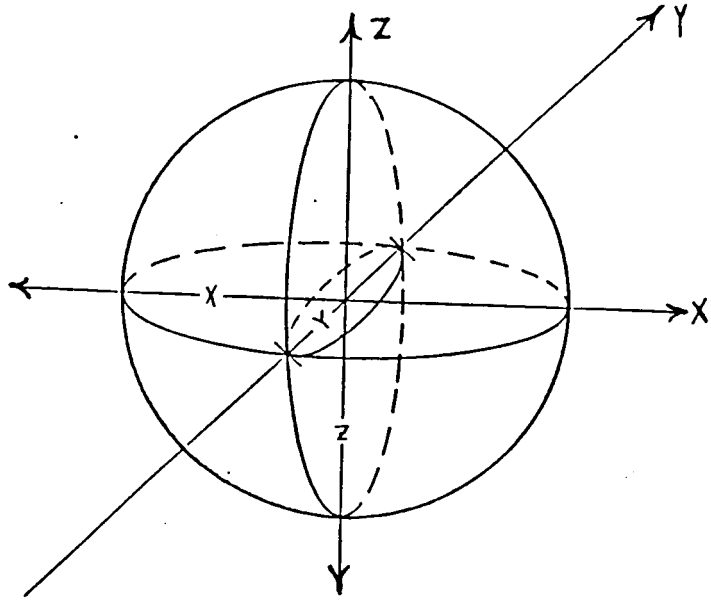
Primitive: Four-Sided Prism
Type Id: PR
Parameters: L D H



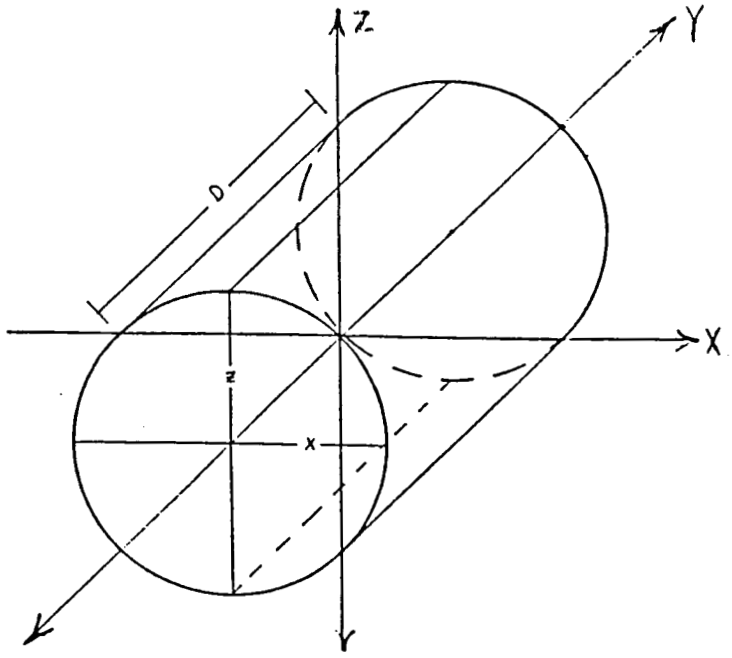
Primitive: Right-Wedge
Type Id: RW
Parameters: L D H



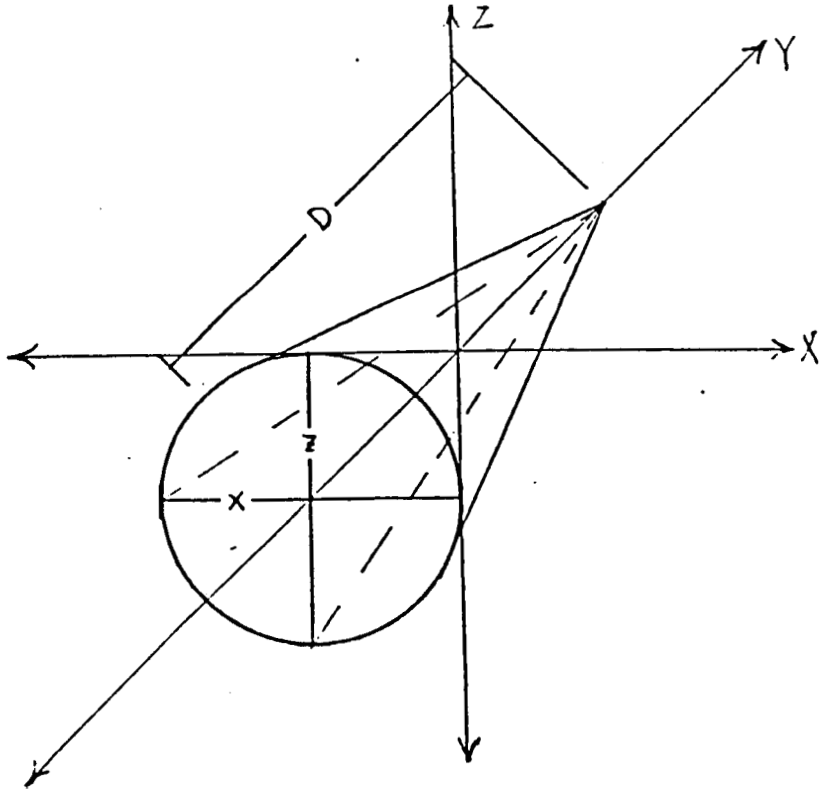
Primitive: Left-Wedge
Type Id: LW
Parameters: L D H



Primitive: Ellipsoid
Type Id: EL
Parameters: X Y Z



Primitive: Cylinder
Type Id: CY
Parameters: X D Z



Primitive: Cone
Type Id: CO
Parameters: X D Z