N90-29024

PLANNING 3-D COLLISION-FREE PATHS USING SPHERES

Susan Bonner and Robert B. Kelley

Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

Abstract – A new scheme for the representation of objects, the Successive Spherical Approximation (SSA), facilitates the rapid planning of collision-free paths in a 3D, dynamic environment. The hierarchical nature of the SSA allows collision-free paths to be determined efficiently while still providing for the exact representation of dynamic objects. The concept of a freespace cell is introduced to allow human 3D conceptual knowledge to be used in facilitating satisfying choices for paths. Collisions can be detected at a rate better than 1 second per environment object per path. This speed enables the path planning process to apply a hierarchy of rules to create of a heuristically satisfying collision-free path.

I. INTRODUCTION

An important part of any robot application is the determination of a collision-free path through the environment for the manipulator. Typical space related applications which might require the planning of collision-free paths in 3D environments include the moving of payloads in the shuttle equipment bay, the construction of space structures, the maintenance and repair of satellites, and the navigation of both surface and winged vehicles. In this paper, we examine the use of a new object representation scheme, successive spherical approximations, which are particularly well-suited for collision detection in the planning of collision-free paths in a cluttered environments. The representation scheme is based on a hierarchy of bounding spheres and rectangular sectors of spheres which correspond to the faces of planar convex polyhedral objects. The path planning process uses an efficient generate-and-test philosophy which exploits human conceptual 3D knowledge to propose and test heuristically satisfying collision-free paths.

A satisfying path planner is one which arrives at a reasonably direct collision-free path with a minimal number of re-determinations of the path. The tradeoff is between the directness of the path and the number of iterations allowed to improve it. In addition, a good planner must be able to accommodate changes in position and orientation of objects in the environment. There are several methods which have been suggested to find such paths, but few have addressed the quality of the path found. A brief review of such methods is given next.

A popular method of planning gross motions has been to create a configuration space (C-space), use spatial occupancy enumeration to model the free space, and determine a path using explicit spatial planning. This works well in two-dimensions and with cartesian manipulators [Lozano-Perez 81], but has been found to be very computationally expensive with articulated manipulators [Gouzenes 84] and in three-dimensions [Brooks 84]. A special form of spatial occupancy enumeration using successive equivalent subdivision of space into octants, octrees, has been used in a similar manner with less computational expense in three dimensions [Faverjon 84, Hayward 86]. The major difficulty in any of these schemes, in addition to the price of computation, is the difficulty in modifying the configuration space in a dynamic environment. Goal directed incremental methods, which use obstacle sensing at run time to find collisions, have been proposed to provide safe paths [Khatib 85, Lumelsky 86]. These schemes, because they are driven by local information, cannot guarantee path directness. In the development of approach paths for grasping strategies, hypothesize-and-test methods have been used to avoid collisions with obstacles

encountered along the path toward the goal [Pickett 85]. Recently, a method has been proposed [Hasegawa and Terasaki 88] which divides a workspace into areas where orientation changes are restricted and where they are not to determine a collision-free path for the manipulator.

The major concern of this paper is to provide a method for path planning which creates heuristically satisfying paths in a dynamic 3D environment. The problems associated with dynamic environments and changing orientations using C-space techniques were judged too severe for a dynamic environment, therefore, the SSA representation is used to model both moving and stationary objects and collisions are handled rapidly in operational space. The efficiency of collision detection and the heuristic nature of operational space, allow paths to be determined and directness improved upon using hypothesize-and-test techniques without unreasonable computational expense. Rules for hypothesizing paths are aided by the concept of a freespace cell, which assigns properties to obstacles depending upon their position in the environment.

First, we provide a brief introduction to the SSA representation hierarchy and its use in collision detection, which is described in detail in [Bonner and Kelley 88]. Then, we introduce a generate-and-test path planner which determines heuristically satisfying paths. We provide a definition of heuristic satisfaction and introduce the concept of a freespace cell, which is fundamental to the efficient determination of paths. A hierarchy of rules for path determination is then discussed. Finally, issues which affect path planning in a dynamic 3D environment are considered.

II. THE SSA REPRESENTATION HIERARCHY

The SSA representation of an object is comprised of a series of successively detailed levels ranging from a sphere enclosing the object to the faces of the object itself. The hierarchical nature of these approximation levels allows for rapid and exact collision detection between 3D objects in a robot workspace with little additional cost incurred by changes in position and orientation of the objects.

The SSA representation is comprised of a series of approximations to an object referenced to a common center. Figure 1 illustrates the SSA approximations for a 2D object. Choice of the center is arbitrary; however, it must be contained within the object, which, in this paper, is limited to a convex polyhedron. Each level is composed of two bounds, an upper bound, which entirely contains the modeled portion of the object and a lower bound, which is entirely contained within the modeled portion. Uncertainty in the position of the object is modeled by adjusting these bounds by an uncertainty measure. Orientation uncertainty is handled by adjusting the angular bounds within the hierarchy.

The *bounding spheres* approximation, which provides the least accurate model of the object, is comprised of a pair of spheres positioned at the object center. The upper bounding sphere, which contains the entire object, has radius R. and the lower bounding sphere, which is entirely contained within the object, has radius r. Position uncertainty information is added at this level by simply increasing the upper bound radius and decreasing the lower bound radius by the determined measure.

The *bounding face* approximation provides an increase in the exactness of the model by dividing the spheres into rectangular sectors determined by the position of the object faces. Each sector is defined by a pair of angle ranges which locate the face with respect to the center. The upper bound for $face_i$, is a sector containing the face of radius R_i,where R_i is the maximum distance from the center to the face. The lower bound is a sector contained within the face of radius r_i, where r_i is the minimum distance from the center to the face. Position uncertainty is handled by increasing the upper bound radius and decreasing the lower bound radius, as with bounding sphere approximations. Orientation uncertainty is provided by increasing the angular ranges of each sector by the desired measure.

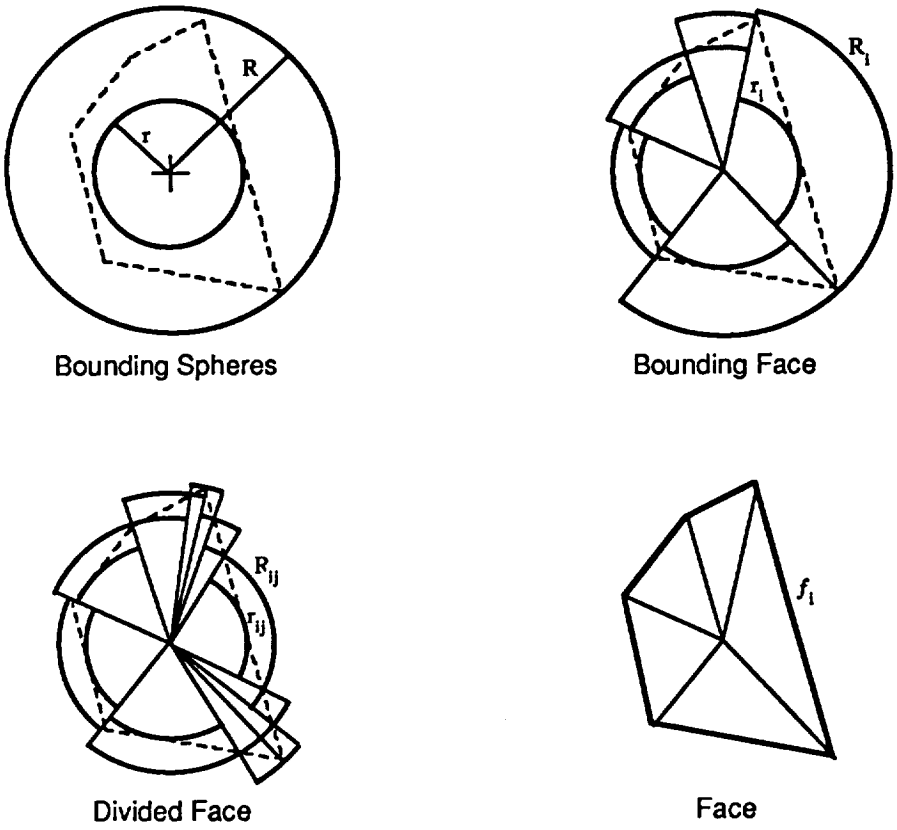

Figure 1. SSA Approximation Levels

The *divided face* approximation further divides each bounding face sector into a set of subsectors which provide an even closer approximation to the object face. Any method which provides a reasonable subdivision of the face into sectors may be used. We perform the sectoring by dividing the distance from the center to the face into even increments between r_i and R_i and assigning a subsector to each increment. The method is computationally straight forward and provides a consistent approximation to the face. Uncertainty for each divided face sector is modeled in a similar manner to a bounding face sector.

The *face* approximation is composed of the bounded planar surfaces which define the faces of the object. Upper and lower bounds are not required, as the representation of a convex polyhedron is exact at this level. Position uncertainty is handled in the definition of the faces by redefining the face vertices with reference to the object center to reflect the additional measure. Orientation uncertainty is not considered at this time.

III. Collision Detection along a Path

The SSA representation is used to determine collisions between an object, the payload, as it moves through the environment along a straight line. The hierarchical levels of the SSA representation are used to simplify the process by determining collisions at the more approximate levels of the hierarchy or, at least, identifying those portions of the objects, if any, which require the detailed process of swept volume collision detection. Each level of the representation may result in three possible outcomes: COLLISION when the objects are found to collide, NO COLLISION when the

objects do not collide, and UNKNOWN when the result is indeterminate and a more detailed approximation level must be used.

At the *bounding sphere* level, the length of the line corresponding to the closest approach of the payload center to the center of each object in the environment is used to determine the collision status. If the closest approach distance exceeds the sum of the radii of the upper bounding spheres for the payload and the object, then there is NO COLLISION for the entire path. If the distance is less than the sum of the radii of the lower bounding spheres, then there is a COLLISION with that object for that path. If the distance is between these two sums, the collision status is UNKNOWN and the next level must be consulted.

At the *bounding face* level, a cumulative sector for each object is found and compared in a similar manner to determine collisions. The first step in determining the parameters of the cumulative sector for an object is to find the swept angular range: i. e., the range of angles swept on each object as it moves relative to the other object. The upper bound radius of the cumulative sector is the maximum of the upper bound radii of all sectors which overlap the swept range. The lower bound radius of the cumulative sector is the minimum of the lower bound radii of all sectors which overlap the swept range.

The *divided face* approximation provides the next level in the hierarchy. The usefulness of this approximation for line collisions depends upon the degree of accuracy required by the path planning strategy and the computational expense required by collision detection at the face level. If collision detection to within a modeled accuracy is acceptable to the path planner, then the divided face level can be used as the final step in the collision detection process. This can result in considerable computational savings. If not, then there is a tradeoff between the additional time spent maintaining the divided face sectors and the computation saved in cases where the face level is avoided. In this paper, the path planner controls the use of the divided face approximation depending upon its requirements. Collisions at this level are determined in much the same manner as at the bounding face level. The divided face sectors which overlap the swept range are combined and compared to determine a collision.

At the *face* level, the detection of line collisions is complicated because of the difficulties involved in checking moving surfaces against each other. To guarantee a collision-free path, swept volume collision detection techniques are used. This results in considerable additional expense because each possibly colliding face of the payload must be swept along the path to create a volume and this volume examined to determine a collision.

The SSA representation is used to reduce such computational expense by eliminating faces which are not involved in a collision and, therefore, do not need to be swept. This is an inherent benefit of the hierarchical collision detection process. The SSA representation and the point collision process also provide an efficient way to model the volume swept by each face and test for collisions with obstacles.

A swept face volume (SFV) is created for each face of the payload involved in a potential collision. The endpoints of the sweep are limited to the points along the path where collisions between the upper bound radius of the cumulative sector for the obstacle and the upper bound radius of the bounding face sector of the payload could possibly occur. The SFV is formed by placing the payload at these two endpoints and creating a boundary representation of the prism formed between the two faces. This volume is then modeled using SSA techniques. The SFV is compared to the bounding face sectors of the obstacle which overlap the swept range to determine if a collision occurs. Since the SFV is stationary, the method for performing the comparison is slightly different. At the bounding sphere level, the outer and inner radii are compared to the distance between the object centers. At the bounding face level, the range of overlap between the swept object and the cumulative sector of the obstacle is found using the upper bounding radii. Sectors

which overlap the base range are determined and compared to find a collision. Both the radius and the angular position of the sectors are used to find collisions in the stationary case. If the Bounding Face Level fails to determine the collision status, the planes of the faces are compared directly to determine if they overlap within the bounds of the faces. The divided face level is not used because the computational complexity involved is too great compared to the time involved in direct comparison of the faces. If any of the SFVs are found to collide with any sectors of the obstacle, then there is a COLLISION. If none collide, then there is NO COLLISION.

IV. PATH PLANNING

The generate-and-test path planning process involves an iterative technique of proposing paths between two points and testing the paths for collisions. The relative collision detection speed provided by the SSA representation allows considerable insight to be incorporated into heuristic path determination rules. The ease with which paths can be checked for collisions allow a hypothesize-and-test solution to perform several iterations to find and improve a path between two points without consuming excessive amounts of time. This section introduces the concepts required to implement a hierarchy of rules for heuristic path determination.

A. Measures of Heuristic Satisfaction

The most fundamental question which guides hypothesize-and-test path determination is "What is a heuristically satisfying path?" The obvious answer is "A path which satisfies our heuristic conception of collision-free travel between two points." To quantify notions of heuristic satisfaction, five collision-free path quality measures are defined: subpath_number, path_length, path_wander, re-orientation_length, and constrainted_length.

Subpath_number: A segmented path is a set of joined straight line segments, subpaths, which form a route from a start point to a goal point. The subpath_number is the number of subpaths in the segmented path. It is heuristically desirable to minimize the number of subpaths.

Path_length: The path_length of a segmented path is the sum of the lengths of each subpath. It is heuristically desirable to minimize the length.

Path_wander: The path_wander of a segmented path is the sum of the absolute angular changes between successive subpaths as the path is traversed. It is heuristically desirable to minimize path_wander.

Re-orientation_length: The re-orientation_length of a segmented path is the length of path over which changes in the orientation of the payload can be made. It is desirable to have as much length of path for re-orientation as possible.

Constrained_length: The constrained_length portion of a segmented path is the length of path where careful manipulation of the payload around obstacles is required. It is heuristically desirable to minimize the length of path requiring careful fitting.

A heuristically satisfying path is a collision-free path which strikes a balance in the satisfaction of the above measures. For example, a heuristically satisfying path must be collision-free and provide a minimum of subpath_number, minimal path_length and minimal path_wander, while providing sufficient re-orientation_length and minimum constrained_length path segments.

B. The Freespace Cell Concept

The freespace cell concept provides the basis for much of the decision making in the path determination procedure. The freespace cell enables path determination rules to take advantage of the inherent structure of an environment by providing a standard direction for avoiding each obstacle in the environment. The basic concept is to regard the freespace as a six-sided box or cell and to associate each obstacle in the environment with one of these six sides. Each side of the freespace cell is defined by its base plane. The interior of the cell forms the allowable space where the payload and manipulator links may travel, with the exception of the space occupied by obstacles within the cell. The freespace cell side associated with an obstacle helps to determine the

path movement direction if a collision with an obstacle is detected. Motion away from the payload base plane should enable the payload to clear the obstacle and help give it a clear path through the workspace. This basic principle is utilized by many of the rules in the path determination procedure.

C. Path Determination Rules

The hypothesize-and-test method of path determination requires that paths be repeatedly proposed and tested until a collision-free path is determined. The path determination rules guide the process by providing an efficient decision-making hierarchy for altering paths in the event of a detected collision. These rules provide a means of creating paths which satisfy the criteria for heuristic satisfaction.

The rules are divided into four sets, each of which governs an increasingly tight obstacle avoidance situation. The Freespace Cell Rules are a basic exploitation of the freespace cell concept applied as a first stage in the path determination process. The Overlapping Influence Rules govern path determination when two obstacles from different freespace cell sides are interfering with the passage of the payload. The Iteration Rules are applied when the other rules fail to find a path in a tight situation. The Orientation Change Rules govern situations when it is discovered that a path cannot be found unless the orientation of the payload is changed. Each of the four sets of rules inputs a subpath with known collisions and divides it into a segmented path to be tested for collisions. A fifth set of rules, Violation Rules are applied when a path endpoint is found to be contained within an obstacle.

1) Freespace Cell Rules. The freespace cell rules use the association of obstacles with a freespace cell side to divide a colliding subpath. There are two steps in this process.

In the *grouping* step, the obstacles which collide are ordered along the subpath. Those which are adjacent and assigned to the same freespace cell side are grouped together. For each group, a cutoff plane, parallel to the base plane, is determined. The distance of the plane from the base is referred to as the height of the cutoff plane. Possible values for endpoints on the cutoff plane at each end of the group are determined and the best choice selected.

In the *combination* step, the segments are combined into a segmented path. A simple algorithm might connect the subpath start point to the first segment, connect adjacent segments, and connect the final segment to the subpath goal point. However, the information provided by the freespace cell can be used to create a more efficient segmented path. If the freespace cell sides of adjacent segments are perpendicular to each other, joining the segments end to end would result in less efficiency than joining the start point of the first segment and the final point of the second segment. The nature of the freespace cell makes this "shortcut" a reasonable alternative to a direct connection. This concept can be extended to three sides by eliminating the middle segment entirely.

An iterative process is used to alter the cutoff plane, if the initial choice results in a collision with opposing freespace cell sides. Successively "lower" heights are chosen according to the following criteria. For the first iteration, the first cutoff plane is at the height which allows clearance and free orientation change of the payload for all obstacles in the group. The second iteration assigns a separate cutoff plane to a segment endpoint and uses only the closest obstacle to that endpoint to determine clearance and orientation changes. The third iteration assigns a height based on the clearance of a single endpoint obstacle with the additional sacrifice of orientation changes to the payload. The process provides clearance of obstacles on the opposite freespace cell side and sets up the most efficient path for further processing. Note that lower heights may result in collisions with obstacles on the freespace cell side of the original segment; these are handled by reapplying freespace cell rules for the path section which collides.

2) Overlapping Influence Rules. The overlapping influence rules identify portions of a subpath which are involved in a collision with more than one obstacle from different freespace cell sides and propose a segmented path which fits the payload between the obstacles. There are three steps in the process.

The *segmentation* of a subpath is achieved by ordering the points at which the colliding obstacles collision may first influence the subpath. The obstacles are then examined for influence areas which overlap with other obstacles. Subpath portions with overlapping influence are combined into segments, the endpoints of which are determined by the part of the subpath influenced by both obstacles. Portions of the subpath influenced by no obstacles, a single obstacle or more than one obstacle from the same freespace cell side are assigned segments using freespace cell rules.

The *combination* step combines the segments into subpaths. The combination is based on the ordering of the first points of influence of the obstacles along the path. The final point of each segment is joined to the starting point of the next segment. An exception to this rule occurs when more than one overlapping influence segment is found to influence the same portion of the path. In this instance, the start point of the first segment is joined to the start point of the second segment. This provides a means of handling more than two obstacles which influence the same portion of the subpath by allowing the direction needed to navigate the first two obstacles to be established and modified by the direction needed to navigate the second two.

In the *fitting* step, the subpaths having overlapping influence are re-segmented to fit between the obstacles. The basis of the re-segmentation is the determination of a segment, perpendicular to the line between the two obstacles and passing through its midpoint. Segments parallel to the side of the obstacle closest to each subpath endpoint are chosen to connect the endpoints to the fitting segment. These segments are combined with the fitting segment create a segmented path for the subpath.

3) Iteration Rules. If a subpath proposed by the overlapping influence rules is found to contain a collision, the incremental collision detection method is applied in the form of iteration rules to find a collision-free path. The iteration rules modify the subpath by finding a collision with an obstacle and following the boundary of the obstacle until the goal point of the subpath can be reached.

The first step in using this set of rules to modify a subpath is to find the point at which the payload collides with an obstacle. This is done by placing the payload at successive intervals along the subpath and checking each point until a collision is found. The direction of the subpath and the freespace cell side of the obstacle are used to determine a direction parallel to the obstacle face involved in the collision. This direction is followed using iterative techniques until the face is cleared (in the manner of [Lumelsky, 88]). As a final step, this last point is joined to the endpoint of the subpath.

4) Orientation Change Rules. If the iteration rules cannot find a collision-free segmented path, then the orientation change rules are applied. These rules determine a new orientation for the payload which optimizes its travel parallel to an obstacle face. The orientation rules also determine areas in which the required orientation can be established.

The face of the obstacle being navigated by the payload has already been established by the iteration rules. The orientation of the payload is chosen so as to minimize its width perpendicular to the line of travel. This is done by finding a line of minimal length which passes through the payload and aligning it perpendicular to the line of travel.

Finding a place to change orientation is accomplished by finding a portion along the subpath over which an orientation change can be made. If this fails, the preceding subpaths are considered in succession until a safe place to change orientation is determined. All subpaths which are considered

must be rechecked for collisions with the new orientation and modified if necessary. A similar process is used to find a place to reestablish orientation after the tight spot has been navigated.

5) Violation Rules. It is possible that a collision may occur when the payload is placed at an endpoint of a proposed subpath. This situation, which prevents the path determination rules from functioning properly, is referred to as a violation and is handled by violation rules. The rules attempt to find an alternative endpoint which does not result in a collision. Two techniques are applied in an attempt to do this, one which uses freespace cell information and a second which uses the line between the obstacle center and the endpoint.

When the first technique is applied, the endpoint is moved to a height "above" the offending obstacle in the direction indicated by the obstacle freespace cell side. Successively "lower" heights are chosen if a collision is found in a similar manner to choosing cutoff planes in *1) Freespace Cell Rules* above. If a collision still occurs, the technique is applied again using the freespace cell associated with the obstacle now involved in the collision. This technique is allowed to be applied only a limited number of times, as it is likely to cycle in a very tight situation.

If the first technique fails to find a new endpoint, a second technique is attempted. In this method, the line joining the obstacle center with the payload is determined and the endpoint is pushed out along that line until the payload no longer collides with the obstacle. If a collision occurs with another obstacle, this technique is applied with the new obstacle. This method is also only applied a limited number of times. If it fails to determine an endpoint, the environment is too cluttered for the path planner to handle.

V. Path Planning Issues

There are several issues involved in path planning which are not addressed in the above path planning algorithm. These issues are vital to the design of a successful path planner in a structured environment. The first issue is the optimization of the generated path to improve its heuristic satisfaction. The second is the ability to provide a change in orientation of the payload as it travels over the path. The third is the ability to find paths in the presence of varying degrees of uncertainty. And the fourth is the affect of the manipulator links as it moves the payload along the planned path.

A. Optimization

Once a collision-free path has been found, a final set of rules, designed to provide additional optimization based on the path_length and path_wander measures of heuristic satisfaction, is applied. These rules attempt to shorten the total path length and make the path more direct by joining together adjacent subpaths and eliminating paths which are not in general agreement with the overall path direction. If the new subpath is collision-free, it replaces the two old ones.

B. Orientation Changes

The path planner has chosen a simple means of effecting orientation changes to the payload which allows maximum flexibility in the determination of where to make the change. The basic approach is to find two paths for the payload, one using the initial orientation and one using the final orientation and to make an reasonable choice of where to make an orientation change based on these two paths. Some simple rules govern this choice.

Orientation changes are only allowed over subpaths which are marked as capable of sustaining one. This marker is established as the path determination rules are applied. Longer subpaths are better suited to changes in orientation because they allow a longer time for the manipulator to effect the change. It is pointless to make a desired orientation change prior to a fit situation requiring a

specific orientation. Therefore, the path planner eliminates such subpaths from consideration prior to these changes and gives special consideration to establishing the desired orientation along the subpath indicated for the re-establishment of the original orientation. If there is no subpath available for orientation changes, the longer subpaths are examined to determine if any portion can sustain an orientation change of the payload. If so, the subpath is divided to supply a choice for orientation changes.

First, the orientation change algorithm creates the two segmented paths and searches the first path for a subpath over which to make the orientation change, using the rules described above. It then creates a subpath, over which to make the change. Next it checks the subpath for collisions. If collisions are found, the path planner uses the basic path planning algorithm to find a collision-free path. The final result is a collision-free path which accomplishes a desired orientation change between the start and goal points.

C. Uncertainty

The nature of the SSA representation enables the path planner to generate paths using different degrees of uncertainty. This allows the system executing the path to rely on interactive sensing to optimize paths and enables path determination in tight situations. In this paper, three levels of uncertainty are established: maximum uncertainty, nominal uncertainty and no uncertainty. The maximum uncertainty guarantees a collision-free path. The nominal uncertainty should provide a collision-free path, but sensing is recommended in tight situations. No uncertainty provides a path in the ideal case and requires the use of sensing to execute. These levels of uncertainty are easily built into the SSA representation of the objects in the environment. Once paths have been established for each level, the path planner examines the relative merits of each, chooses the best, and stores those remaining for possible use in error recovery.

D. Manipulator Restrictions

The physical construction of the manipulator which moves an object through the environment puts restraints on the path the object can take. The payload is comprised of the object being moved, the gripping mechanism which holds it, and any links of the manipulator which affect orientation changes. To find collision-free paths which provide areas for orientation change, the entire payload must be modeled by a single SSA representation and used to find paths through the environment.

Once a path for the payload has been established, possible collisions of the positioning links must be examined. This is done by examining the paths dictated by the motion of the payload on each positioning link. To check the paths of the links for collisions, each link is modelled using the SSA representation and checked for collisions with the environment obstacles as it travels along its dictated path. Each manipulator link has a work envelope out of which it cannot travel. Environment obstacles which do not fall within the work envelope of a link need not be considered for collisions with that link. Since a straight line path for the payload does not map into straight line paths for the links, the incremental collision detection method is used to check for collisions at regular intervals along the path. If a collision is found, two alternatives are tried to eliminate the problem.

The first alternative is to chose an alternate kinematic configuration to find another path for the link. In an articulated arm such as the PUMA there are two sets of kinematic configurations: left_elbow and right_elbow, and elbow_up and elbow_down. The configuration used for initial path determination depends upon the position and function of the manipulator in the environment. Alternative configurations are checked according to the freespace cell side of the obstacles involved in the collision and the elbow configuration chosen. If the configuration changes fail to establish a collision-free path, changes to the path of the payload are attempted.

The second alternative attempts to alter subpaths of the payload path to eliminate link collisions. The only subpath types which are considered alterable are those formed using freespace cell rules. Subpaths which require fitting between obstacles are assumed to be too finely tuned to enable alterations substantial enough to create enough change in link position to make any difference. The freespace cell rules for altering the height of the subpath are applied in a manner which might eliminate the collision problem. If the alterations result in a collision-free path, then the links are rechecked for collisions.

VI. CONCLUSION

In this paper, we have introduced a method for the efficient determination of collision-free paths in a 3D environment. The method is based on a novel spherical representation of environment objects. The paper defines goals for creating heuristically satisfying paths and introduces the concept of a freespace cell, which exploits the structure of the environment to facilitate satisfying choices for paths. Collisions can be detected at a rate better than 1 second per environment object per path. This speed enables the path planning process to accommodate uncertainty, object re-orientation and a dynamic environment while creating a heuristically satisfying collision-free path.

REFERENCES

Bonner, Susan and Kelley, Robert, B., "A Representation Scheme for Rapid 3-D Collision Detection," Third International Symposium on Intelligent Control, Aug. 1988 (in press).

Brooks, Rodney A., "Planning Collision-free Paths for Pick and Place Operations," First International Symposium on Robotics Research, Aug./Sept. 1984, pp. 5-37.

Faverjon, Bernard, "Obstacle Avoidance using an Octree in the Configuration Space of a Manipulator," IEEE International Conference on Robotics, Atlanta, GA, Mar. 1984, pp. 504-512.

Gouzenes, Laurent, "Strategies for Solving Collision-free Trajectories Problems for Mobile and Manipulator Robots," *The International Journal of Robotics Research,* v3 n4, Winter 1984, pp. 51-65.

Hasegawa, Tsutomu, and Terasaki, Hajime, "Collision Avoidance: Divide-and Conquer Approach by Space Characterization and Intermediate Goals," *IEEE Transactions on Systems, Man and Cybernetics*, v18 n3, May/June 1988, pp. 337-347.

Hayward, Vincent, "Fast Collision Detection Scheme by Recursive Decomposition of a Manipulator Workspace," IEEE International Conference on Robotics and Automation, San Francisco, CA, Apr. 1986, pp. 1044-1049.

Khatib, Oussama, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," IEEE International Conference on Robotics and Automation, St. Louis, MO, Mar. 1985, pp. 500-505.

Lozano-Perez, Tomas, "Automatic Planning of Manipulator Transfer Movements," *IEEE Transactions on Systems, Man and Cybernetics,* v11 n10, Oct. 1981, pp. 681-698.

Lumelsky, Vladimir J., "Continuous Motion Planning in Unknown Environment for a 3D Cartesian Robot Arm," IEEE International Conference on Robotics and Automation, San Francisco, CA, Apr. 1986, pp. 1050-1055.

Pickett, E. E. and Jha, R., "Geometric Reasoning in Task Planning," Intelligent Robots and Computer Vision (SPIE v579), Cambridge, MA, Sept. 1985, pp. 121-130.

NAVIGATION