Guidance Algorithms for a Free-Flying Space Robot

A.F. Brindle, H.E.M. Viggh, J.H. Albert
Boeing Aerospace
P.O. Box 3999 MS 82-58
Seattle, WA 98124

Abstract

Robotics is a promising technology for assembly, servicing, and maintenance of platforms in space. This project is investigating several aspects of planning and guidance for telesupervised and fully autonomous robotic servicers. This paper describes ongoing work on guidance algorithms for proximity operations of a free flyer. The general approach combines numeric trajectory optimisation with artificial intelligence based obstacle avoidance. An initial algorithm and results of simulating it on a platform servicing scenario are discussed. A second algorithm experiment is then proposed.

Keywords: Autonomous robotics, spacecraft servicing, artificial intelligence, trajectory optimisation, obstacle avoidance.

1 Introduction

Robotics is a promising technology for assembly, servicing, and maintenance of platforms in space. Robots will reduce expensive and risky astronaut extra-vehicular activity (EVA) around manned systems and permit servicing of remote platforms. Several robots, for example, have been proposed for the Space Station (see the reports of the NASA Advanced Technology Advisory Committee [1] and subsequent studies, including the Phase B Automation and Robotics Plan for Work Package 1 [2], which has an excellent bibliography.) Proposals for spacecraft servicers include teleoperated, telesupervised, and autonomous robots. Teleoperations will be the first step in deploying on-orbit robots, but long term goals include telesupervision and perhaps full autonomy for servicers.

This project is investigating several aspects of planning and guidance for a telesupervised and autonomous robotic servicers. This paper describes work on guidance algorithms for proximity operations of a free flyer. First, the selection of a servicing scenario is described and the functional focus of the project defined. Then the general project approach to developing guidance algorithms is presented. The strategy is to combine numeric trajectory optimization with artificial intelligence (AI) based obstacle avoidance to provide autonomous motion between specified start and goal points. Such guidance would be essential for autonomous operations and would form a component of a telesupervised system. The first algorithm and results of simulating it on the scenario are discussed. Finally, a proposal is given for a second algorithm experiment.

2 Robotic Spacecraft Servicer Scenario Definition

The domain of spacecraft servicing scenarios is large and varied. We considered these requirements for the project scenario:

- The scenario should address the needs of the space community roughly 20 years hence.
- The scenario should drive advances in AI and guidance technology.
- The scenario should not impose requirements on other entities in the environment to facilitate robot operation.

A note is needed about the last requirement. In most scenarios, it is possible to restrict or control other entities and thus ease requirements on the servicing robot. For example, it may be mandated that whenever a robot exits its orbiting home base, there will be no other entities active around the base, or no entities around the base at all. Any platform to be serviced may be required to have predefined corridors of safe approach for a servicer. Restrictions such as these will undoubtably be imposed on other entities in the future; however, if they are not assumed at this point, this research will aid in determining which restrictions are needed.

The next subsection defines a scenario in terms of the degree of autonomy assumed for the robot, the location of servicing, the type of target platform, and the complexity of the surrounding environment. The following subsection then describes the project focus within the selected scenario.

2.1 Scenario

A robotic servicer located in a bay of an orbiting home base is instructed to refuel and repair a remote platform. It is given a service order, such as a human maintenance engineer would receive, a description of the platform, including shape and current orbit (and probably where to obtain updates on the orbit), and a deadline for completing the work. It must obtain necessary supplies, exit the home base, perform orbit transfer, maneuver into proximity to the platform and inspect it from several points of view, dock, and service the platform.

The platform is a complex shape composed of trusses, solar panels, and antennae. Maneuvering around such a platform will require more sophisticated guidance than maneuvering around a small compact satellite. The home base is also a complex shape, and it is busy. Other robots and astronauts are moving around the base, and obstacles exist within its vicinity. The robot has to avoid multiple, moving, actively propelled objects.

2.2 Project Focus

This project focuses on guidance algorithms for the proximity operations performed by the robot near the home base and the platform. Proximity is defined as within 1000 feet. For these operations, the robot has a cold-gas thruster system similar to that of NASA's Manned Maneuvering Unit (MMU). To reflect this focus, the scenario is further defined to be an inspection task by a maintenance robot at the Space Station. Thus the home base and target platform are the same, and no orbit transfers are required.

There are four major functional components involved in proximity guidance.

1. Identification and tracking of platform and obstacles.
2. Generation of feasible goal positions and times.
3. Generation of robot trajectories.
4. Execution of trajectories, including reactive planning to handle contingencies.

These experiments are focused initially on the third function, trajectory generation. Therefore, a goal position and time have already been determined as input to the guidance module. The guidance system must plan a trajectory to this goal state which minimizes fuel and avoids foreseen obstacles.

It is assumed that this planning activity occurs prior to execution of the trajectory, while the robot is in a safe, stable position and able to sense the scene. Some path planning researchers argue that such *predictive* planning is of little utility for terrain vehicles and should be replaced by local, or *reactive*, responses to the immediate environment. This stems from the fact that robotic sensing and modeling of an uncertain and changing world is incomplete and inaccurate, leading to poor plans.

Certainly space robots will need reactive planning capability to respond to active obstacles, unforeseen obstacles, and other contingencies. However, there are two arguments in favor of having predictive planning as well. First, the environment is almost entirely synthetic, has a small number of understandable objects, and does not change rapidly. Accurate modeling of obstacles is quite conceivable. Second, there is a strong need for global optimization of trajectories. All space vehicles are fuel limited, so *a priori* identification of fuel efficient routes is highly desirable.

It is also assumed that object sensing and modeling can predict passive motions on orbit and can recognize regular motions of articulated parts on the target platform, with the help of the platform models supplied to it. Therefore, the scenario leads to the following requirements on trajectory planning by the guidance system:

1. The system generates a trajectory, or route plan, from a given start a given goal position.
2. The trajectory avoids all passive, foreseen obstacles, which may be in motion.
3. The trajectory avoids the platform of interest, which is a composite of different shapes.

3 Approach

The guidance algorithms must include 1) trajectory optimization to minimize fuel consumption, with orbital mechanics taken into account, and 2) obstacle avoidance. The general approach is to combine optimization techniques from the domain of spacecraft guidance with path planning and obstacle avoidance techniques from artificial intelligence (AI). Numerical optimization techniques cannot cope with the constraint explosion which occurs when more than one or two simple obstacles are modeled. AI has several techniques which handle multiple obstacles under various limiting assumptions, but these techniques have not been applied to on-orbit problems. By prototyping and

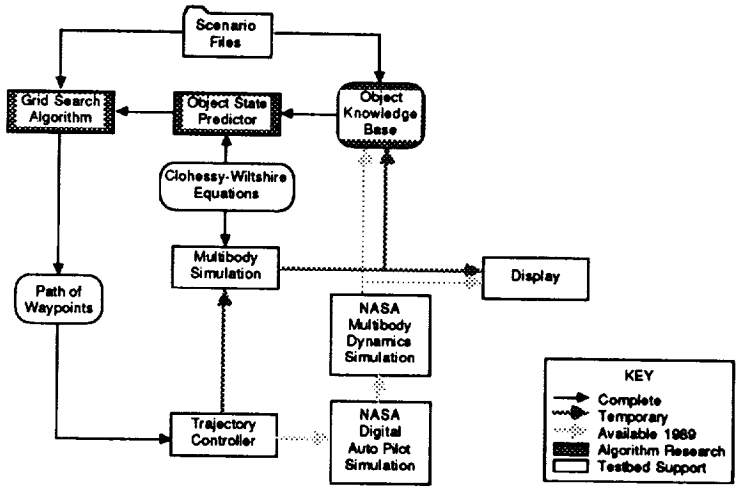

Figure 1: Software Simulation Testbed for Guidance Algorithm Experiments

evaluating algorithms which borrow techniques from both domains, we hope to arrive at a guidance system for safe, efficient on-orbit maneuvering.

The prototyping experiments are being performed on a Symbolics workstation. Several tools have been developed or obtained to form a simulation testbed for the work. The testbed is illustrated in Figure 1. The environment of platform and other objects is currently simulated on the Symbolics using Clohessy-Wiltshire equations for the orbital mechanics, but future experiments will use a multibody dynamics simulation acquired from NASA Johnson Space Center. This simulation has been installed on a VAX/UNIX system communicating with the Symbolics via Ethernet. The code includes a digital autopilot simulation for one of the bodies, which has been configured to match the MMU and will simulate the robot control and actuation systems for future closed loop experiments.

The simulations generate obstacle state information which is placed in the object knowledge base. Object shape information is loaded into the base from scenario files. The base controls concurrent access, so that simulation and planning can occur in separate processes.

The interface for algorithm demonstrations uses the Symbolics S-Geometry package for three dimensional, wire-frame graphical display. While displays have been generated on other, more graphics-oriented devices, hosting a 3-D display on the development machine was found to be indispensable for algorithm development and evaluation.

4 Grid Search Algorithm

The first experiment involved modifying a standard AI path planning technique to optimize fuel on-orbit, rather than the usual optimization of distance in two dimensions. Several AI techniques may extend to higher dimensionality, such as polyhedral blocks [7], or octrees [4,6] (see [3] for a survey). A* search [9] on a uniform grid, also called grid search [5,8], was selected for the initial implementation. This method is one of many which generate a space of points between the start and goal points. The space is searched for a set of waypoints which, when connected, form a safe, nearly optimal path from start to goal.

Table 1: Steps in the A* Search Algorithm

	Form a queue to hold partial paths from the start to other endpoints.
	Add the null path from start to start to the queue.
loop	If the queue is empty, return. failure to find path.
	Remove path P from the front of the queue.
	If P reaches the goal, return P as the solution.
	Form new paths by extending P toward the neighbors of P's endpoint.
	Add the new paths to the queue.
	Sort the queue by total cost, keeping lower cost paths in the front.
	If several paths in the queue have the same endpoint, discard all but the one with least cost.
	Repeat from loop.

A* search is based on discrete dynamic programming, which accomplishes breadth-first searching of a cost-weighted graph to find a shortest path. A* search differs from dynamic programming in that a heuristic factor is added to the cost function. A partial path's total cost is the sum of the cost of the path plus a heuristic estimate of the cost remaining from the endpoint of the path to the goal. The heuristic may render the solution less than optimal, but its use reduces search time. The algorithm is described in Table 1.

Adapting the grid search to space required new definitions of the cost function, the search space, the cost heuristic factor, and the grid with its notion of neighbor.

- The goal of minimizing fuel was approximated by using a cost function of magnitude of delta-v. The robot is assumed to fire its thrusters instantaneously, and only at the grid points. Moving from point A to point B is modeled as a single burn (thruster firing) at point A, and the magnitude of the change in velocity at point A is the cost of moving from A to B.

  There may be goals imposed on the robot in addition to minimizing fuel. To permit experiments which minimize time or distance as well, the cost function for moving from A to B was implemented as the weighted sum of 1) magnitude of delta-v, 2) straight line distance, and 3) time duration. The weights are specified as part of the scenario.

- The choice of cost function and the presence of moving obstacles necessitated the use of a grid in seven dimensional space. Each point is a partial robot body state, including three dimensions for position, three dimensions for incoming velocity, and one dimension for time. Another way to view this is that in space, the advantage of being at a certain position depends upon when you are there, since hazards are in motion, and what your velocity is, since turning and braking in space are expensive.

  A reference frame also had to be chosen for the search space. The NASA simulation employs an earth-centered inertial frame and has full frame transformation capabilities. AI path planning is usually terrain based or airborne and uses a local, non-inertial frame. For this work, a local vertical, local horizontal (LVLH) frame, on orbit and centered on the platform of interest, was selected. Orbital effects are easily modeled in this frame, and position and velocity vectors for obstacles and robot are of a scale which the grid search can manage.

- When spatial distance is the grid search cost function, the heuristic cost factor is usually the straight line distance from path endpoint to the goal. For the on-orbit algorithm, the

heuristic cost is computed by solving a two burn problem, one at the path endpoint and one at the goal, to reach the goal in a direct (but not necessarily straight line) trajectory.

- A uniform grid and a definition of the neighbors to each point was needed for the seven dimensional search space. In two spatial dimensions it is sufficient to specify a resolution distance d to achieve a uniform grid. All points (id, jd) for integer i and j are on the grid. A point's neighbors are usually declared to be any point on the grid less than 2d distance from the point (8 neighbors) or 1d distance away (4 neighbors). This is easily extended to three spatial dimensions. If it is extended to the fourth dimension of time by the declaration of a time resolution t, however, so that neighbors are, say, 1d distance away and 1t time away, the effect is to fix the speed of robot to the one value d/t. This is undesirable for an on-orbit robot.

  Many schemes are possible which result in a uniform grid in seven dimensions, a small number of neighbors for each point, and a variable speed robot. For the first experiment, a uniform grid was defined in four dimensions in terms of distance resolution d and time resolution t. A point was defined to be on a grid in seven dimensions if it was on the space/time grid and its velocity was one of the set of velocities achievable by arriving from a neighboring point. A point's neighbors were defined to be all those grid points in two sets:

  - those time t away and any distance away, subject to limits on maximum robot speed and maximum instantaneous delta-v magnitude, and
  - those points distance less than 2d away and any time away, subject to limits on minimum robot speed and maximum instantaneous delta-v magnitude.

  For example, assume t and d are 1, the robot is stopped at point (0, 0, 0) at time 0, and robot speed may vary from .4 to 2.5. Ignore limits on delta-v magnitude. Then the robot may move to (1, 0, 0) at times 1 or 2 (with speed 1 or .5), or to point (2, 0, 0) at time 1 (with speed 2), but not to point (1, 0, 0) at time 3 or point (3, 0, 0) at time 1, since robot speed limitations are exceeded. As another example, assume that robot speed is limited to be between 1 and 1.9. Then the neighbors of (0, 0, 0) at time 0 are the 26 points (i, j, k) at time 1, where i, j, and k are -1, 0, or 1, but not all 0.

5 Experimental Evaluation of the Algorithm

Table 2 summarizes the characteristics of a test scenario. A robot is directed to move to a point near one module of the space station for an inspection. An obstacle is moving by the station. This is unrealistic, but was included to test the ability of the robot to avoid moving obstacles. The robot has no velocity initially in the station-centered local frame. The goal points always have a zero desired velocity as well. The search occurs on a grid with 6 foot, 60 second resolution. The runs were arbitrarily limited to 6 hours of execution time.

The goal points all have a zero desired velocity and no specified arrival time. It was discovered that the algorithm can achieve a goal position and velocity at a reasonable time, if no arrival time is specified. This behavior is obtained by not testing for compliance with a goal time and by making a slight adjustment to the cost heuristic. In the first case, the two-burn problem for the cost heuristic consists of finding two delta-v's to move from a neighbor state (position, velocity, time) to a goal state (position, velocity, time). In the second case, the two-burn problem is modified to motion

Table 2: Characteristics of the Test Scenario

Objects	
Platform	Modeled after Space Station 14 Component objects: cylinders and rectahedrons
Obstacle	Cylinder, moving
Robot	Displayed as cylinder, modeled as sphere .09 feet/second minimum speed .19 feet/second maximum speed .2 feet/second maximum single delta-v magnitude 0 initial velocity 0 goal velocity
Scale	
Grid resolution	6 feet distance 60 seconds time

from a neighbor state (position, velocity, time) to a partial goal state (position, velocity). Since this is not sufficiently constrained to yield a single solution, the constraint is added that the goal time, for the one cost evaluation only, is set equal to the distance from neighbor to goal divided by the magnitude of the neighbor point's velocity. Effectively, the heuristic assumes that the robot will continue directly to the goal position at roughly its current speed.

Several experiments were performed to see how grid search would behave on the scenario for various goal points and cost function weighting schemes. The cost function for these runs was the weighted sum of the total delta-v and the total distance (time was not considered). The results are given in Tables 3 and 4. The path identified by the algorithm for the goal (8.667, -0.667, 9.833) and delta-v and distance both weighted 1 is illustrated in Figure 2. To summarize the results, the algorithm takes a very long time to find an obstacle free, fairly expensive trajectory.

To understand why the A* algorithm is taking so long, consider two points. First, a dynamic programming algorithm, without the cost heuristic, will not work in a domain where the robot may move from point to point without incurring cost. In space, the algorithm will establish a starting velocity for the robot, and will then head off forever in the direction of that velocity. It will have discovered a zero cost path of ever-increasing length, which it considers superior to all other paths requiring some delta-v. This does not happen in two spatial dimensions with a distance-based cost function, because it is not possible to move from point to point without incurring some cost (it is also true that dynamic programming in two dimensions is usually constrained to stay between the start and goal points on one dimension). The A* algorithm does not head off forever on a free trajectory. However, it still prefers to explore passive trajectories once initial velocity has been established, and the cost heuristic does not affect path evaluations enough to counter this effect until a great deal of exploration has occurred.

Second, for the on-orbit problem, the search space is very large. The seven dimension space and the desire for a variable speed robot lead to many neighbors at each point. A limit on the magnitude of the delta-v allowed at each point helps somewhat, but not enough.

Table 3: Number of Points Searched during Grid Search

Cost Function Weights: Delta-v Goal Distance	0.0 1.0	1.0 1.0	1.0 0.5	1.0 0.1	1.0 0.0
<1 2.167 0>	18	73	127	538	1883
<1 4 1>	67	274	492	2351	11,217
<8.667 -.667 9.833>	1560	33,361	*	*	*

* Indicates run did not complete within 6 hours.
The start, <0 0 0>, and goals are in grid-relative coordinates.

Table 4: Total Delta-V for Paths Identified by Grid Search

Cost Function Weights: Delta-v Goal Distance	0.0 1.0	1.0 1.0	1.0 0.5	1.0 0.1	1.0 0.0
<1 2.167 0>	0.355	0.355	0.355	0.333	0.333
<1 4 1>	0.485	0.420	0.420	0.420	0.420
<8.667 -.667 9.833>**	0.872	0.870	*	*	*

* Indicates run did not complete within 6 hours.
** For comparison, total delta-v for a three-burn, obstacle free trajectory was 0.404.
The start, <0 0 0>, and goals are in grid-relative coordinates.

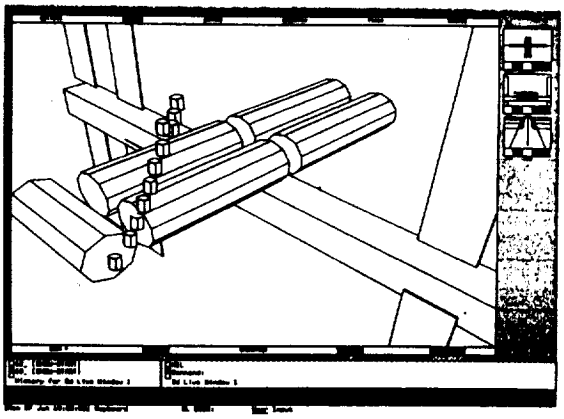

Figure 2: Path Identified by the Grid Search Algorithm

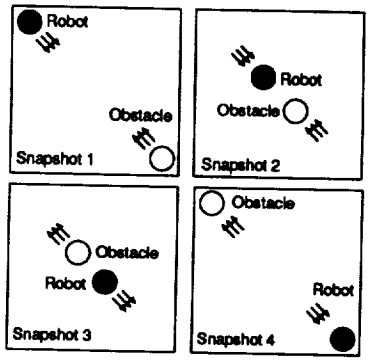

Figure 3: Approach to Obstacle Avoidance Permits Collisions on a Coarse Grid

The size of the search space in combination with the lack of tight focus on the goal makes the identification of paths of more than five burns unfeasible. In fact, the number of points evaluated grows exponentially with the number of waypoints in the path. The search problem is NP-hard, which means that order-of-magnitude improvements in computing hardware will only make a small dent in the performance problem.

However, guidance engineers have been quick to point out that the problem of Figure 2 could probably be solved with two or three burns. To verify this, a three burn, obstacle free trajectory was proposed by viewing the scenario and selecting an intermediate waypoint above the modules. Simulation of the trajectory revealed that its total delta-v would be half that of the trajectory identified by the A* search. 13 burns is a needlessly complicated and suboptimal solution.

This suggests that a coarser grid may be appropriate. Unfortunately, there is an upper limit to grid resolution which is imposed indirectly by the approach to obstacle avoidance. Obstacles are avoided by testing for robot-obstacle intersections at the states defined by the grid points. No consideration is given to whether the trajectories between grid points intersect obstacles. This is perfectly acceptable if the grid points are close together compared to the sizes and relative velocities of robot and obstacles. If the grid points are too far apart, the planner may promote a trajectory which appears safe at the waypoints, but which will in fact lead to a collision (see Figure 3).

Therefore, the conclusion from the first experiment was that a second algorithm was needed which would employ coarser grids and consider obstacle avoidance over the trajectories, not just at the grid points.

6 Proposed Second Experiment

The second experiment will utilize numerical techniques for collision testing and will assume that most trajectories are accomplished efficiently with a small number of burns. The approach is to solve a two burn problem for an optimal trajectory, and if the resulting trajectory fails to avoid obstacles, to move to successively more burns. The timing and location of additional burns will be identified by searching for burn points on a grid which is scaled to the current total number of burns. Thus the search will retain aspects of a grid search, but will occur on a sequence of grids of increasingly finer resolution.

Obstacle trajectories and robot trajectory segments between burns will be represented by polynomials. Collisions will be detected by testing the polynomials for intersection. Previous experiments indicate that obstacles which must be avoided in space can be modeled as circumscribing spheres and their entire vicinity avoided. Thus it should be possible to describe passive, spherical obstacles trajectories mathematically.

The platform of interest, on the other hand, must be modeled in more detail. Converting complex, moving shapes into mathematical representations is generally not feasible. However, by restricting the platform to be unarticulated and by employing a platform-relative reference frame, we can transform robot-platform collision testing into comparing a robot trajectory to a set of stationary shapes.

7 Conclusions

This project has demonstrated the potential for combining trajectory optimization and AI path planning for on-orbit robotic guidance. It has clarified several issues about algorithm design which require further investigation. In particular, it remains to be seen whether critical assumptions about the on-orbit scenario can be made which circumvent the combinatorial explosion in computation time. This explosion plagues all approaches to path optimization with obstacle avoidance.

This effort has also revealed several other directions for work in support of autonomous space robotics, including further analysis of requirements for servicing scenarios and development of functionality outside of predictive planning for free-flying guidance. In particular, research is needed on sensing, obstacle modeling, and reactive planning in conjunction with robot control.

References

[1] *Advanced Technology Advisory Committee, Executive Overview.* TM 87566, NASA, April 1984.

[2] *Space Station Automation and Robotics Plan.* Document D483-50055-1, Boeing Aerospace, October 1986. Data Requirement 17 for Work Package 01, NASA Contract NAS8-36526.

[3] A.F. Brindle, W. Kohn, G.M. Lobdell, and J.H. Albert. *Space Robot Path Planning: Project Proposal.* Document D180-30735-1, Boeing Aerospace, December 1987.

[4] H.H. Chen, N. Ahuja, and T.S. Huang. Septree representations of moving objects using hexagonal cylindrical decomposition. *Optical Engineering*, 23(5):531–535, September/October 1984.

[5] A. Elfes. A sonar-based mapping and navigation system. In *International Conference on Robotics and Automation, Volume 2*, pages 1151–1156, IEEE, San Francisco, CA, April 1986.

[6] M. Herman. Fast, three-dimensional, collision-free motion planning. In *International Conference on Robotics and Automation, Volume 2*, pages 1056–1063, IEEE, San Francisco, CA, April 1986.

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

[8] C.E. Thorpe. Path relaxation: path planning for a mobile robot. In *Autonomous Mobile Robots Annual Report 1985, CMU-RI-TR-86-4*, pages 39–42, Carnegie-Mellon University, February 1985.

[9] P.H. Winston. *Artificial Intelligence.* Addison-Wesley, 1984.