Proceedings of the 2004 IEEE
International Conference on Robotics & Automation
New Orleans, LA • April 2004

Guided Expansive Spaces Trees: A Search Strategy for Motion- and Cost-Constrained State Spaces

Jeff M. Phillips
Dept. of Computer Science, Rice University,
6100 Main St. Houston, TX 77005, USA
jeffp@cs.rice.edu

Nazareth Bedrossian
C. S. Draper Laboratories,
2200 Space Park Blvd. Suite 210
Houston, TX 77058, USA

Lydia E. Kavraki
Dept. of Computer Science, Rice University,
6100 Main St. Houston, TX 77005, USA
kavraki@cs.rice.edu

Abstract— **Motion planning for systems with constraints on controls or the need for relatively straight paths for real-time actions presents challenges for modern planners. This paper presents an approach which addresses these types of systems by building on existing motion planning approaches. Guided Expansive Spaces Trees are introduced to search for a low cost and relatively straight path in a space with motion constraints. Path Gradient Descent, which builds on the idea of Elastic Strips, finds the locally optimal path for an existing path. These techniques are tested on simulations of rendezvous and docking of the space shuttle to the International Space Station and of a 4-foot fan-controlled blimp in a factory setting.**

I. INTRODUCTION

The need for autonomous service of satellites and autonomous maintenance of the International Space Station (ISS) requires real-time obstacle avoidance. Advanced technology programs such as the DARPA Orbital Express program [18] are now underway to demonstrate the various technologies required to achieve autonomous in-orbit servicing capability. The rendezvous control problem has been extensively researched [13], [10].

Likewise, control of an autonomous agent, such as a small fan-controlled blimp, in a factory-type setting requires the planning of relatively straight and efficient paths. Obstacles need to be avoided and the path should not wander. Modeling the motion of and controlling small blimps has also been the focus of several research studies [11], [2].

This paper presents a path planning algorithm for handling such systems. We test our algorithm by finding fuel-efficient paths to dock the space shuttle to the ISS amidst a shower of realistically moving asteroids as well as on a blimp efficiently maneuvering through a factory hall mazed with pipes.

The initial phase of the algorithm finds an efficient path using guided Expansive Spaces Trees (guided ESTs) to focus a randomized search on the low cost region while expanding a tree similar to ESTs [9], [8], a probabilistic sample-based search technique [12], [16]. Our method generates waypoints by probabilistically branching off of existing waypoints. It weights each waypoint based on, not only the number of close waypoints, but also the estimated total cost (A* cost) of going through that waypoint on a path to the goal. This not only focuses the search towards the goal, but it greatly reduces the number of waypoints rejected from having too high cost.

The second phase of the algorithm refines the existing path according to a cost function by following the gradient of the path. This path gradient descent technique is similar to work on Elastic Strips [17], [4], [15], but does not enforce elastic properties of the path, an unrealistic constraint in this situation, and takes more robust precautions in repelling from obstacles.

There has been a long history of sample-based probabilistic path planning methods. Some build a roadmap (a graph) in the configuration space [12], [3] and some construct trees for single queries [9], [16]. However, for the particular class of problems we address, none of the existing techniques met our demands as well as the method presented here. The existing techniques expansion properties were important for robustly finding paths, but the problems we address also require the paths generated to obey certain cost constraints and be relatively straight. This work provides further evidence that the way sampling is done and sensitivity of such schemes to distance metrics is far from solved.

Section II gives a brief background in kinodynamic motion. Section III outlines the guided EST algorithm, and IV outlines the path gradient descent algorithm. Section V summarizes our set of experiments. And VI concludes the paper and looks at future directions of this research.

II. KINODYNAMIC MOTION

Kinodynamic motion planning [16] performs in a state space. A state is defined by a static representation (position, orientation, and other state variables) of an object, derivatives of that representation w.r.t. time, and time. Here we consider two modeled systems.

Orbital dynamics for rendezvous and docking problems can be effectively modeled with the Clohessy-Wiltshire (CW) equations (1) [6], which are a linear function of change in time, t, and orbital rate, n. These equations allow the system to be planned relative to the target vehicle, instead of relative to the earth. For small changes in altitude, orbital rate, n, can be modeled as constant.

$$\begin{bmatrix} x_f \\ y_f \\ z_f \\ \dot{x}_f \\ \dot{y}_f \\ \dot{z}_f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 6(nt-\sin(nt)) & \frac{4}{n}\sin(nt)-3t & 0 & \frac{2}{n}(1-\cos(n*t)) \\ 0 & \cos(nt) & 0 & 0 & \frac{\sin(nt)}{n} & 0 \\ 0 & 0 & 4-3\cos(n*t) & \frac{2}{n}(\cos(nt)-1) & 0 & \frac{\sin(nt)}{n} \\ 0 & 0 & 6n(1-\cos(nt)) & 4\cos(nt)-3 & 0 & 2\sin(nt) \\ 0 & -n\sin(nt) & 0 & 0 & \cos(nt) & 0 \\ 0 & 0 & 3n\sin(nt) & -2\sin(nt) & 0 & \cos(nt) \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ \dot{x}_i \\ \dot{y}_i \\ \dot{z}_i \end{bmatrix} \quad (1)$$

Orientation and angular rate can similarly be modeled with a set of linear functions of time. Controls can be modeled as instantaneous changes in velocity or rate, resulting from short jet firings.

The model for blimp motion maneuvering in a factory setting is very similar to differential drive motion [1]. The

blimp modeled in the paper has three independent fans, used for control. The first two fans face along the main forward axis of the blimp and are separated by a distance l. The weight of the central control box prevents nonzero pitch and roll, but yaw can be controlled by the difference in the controls of the first two fans: u_1 and u_2.

$$\begin{aligned} \ddot{\theta} &= \tfrac{u_1-u_2}{l} \\ \dot{\theta}_f &= \dot{\theta}_i + \tfrac{u_1-u_2}{l} t \\ \theta_f &= \theta_i + \dot{\theta}_i t + \tfrac{1}{2}(\tfrac{u_1-u_2}{l})t^2 \end{aligned} \tag{2}$$

The horizontal acceleration can then be written:

$$\begin{aligned} \ddot{x} &= (u_1+u_2)\cos(\theta_i + \dot{\theta}_i t + \tfrac{1}{2}(\tfrac{u_1-u_2}{l})t^2) \\ \ddot{y} &= (u_1+u_2)\sin(\theta_i + \dot{\theta}_i t + \tfrac{1}{2}(\tfrac{u_1-u_2}{l})t^2) \end{aligned} \tag{3}$$

The control for the third fan, u_3, determines vertical displacement and can be modeled independently of the first two controls. A damping term, Ω, is added:

$$\begin{aligned} \ddot{z} &= u_3 \\ \dot{z_f} &= (\dot{z_i} + u_3 t)(1-\Omega) \\ z_f &= z_i + (\tfrac{2-\Omega}{2})\dot{z_i} t + \tfrac{1}{2}(1-\Omega)u_3 t^2 \end{aligned} \tag{4}$$

Vertical displacement and yaw can thus be described explicitly, but the equations for the horizontal displacement have no closed form and their Taylor approximation is insufficiently accurate. Thus horizontal displacement and velocity is calculated using Runga-Kutta 3/8, which also takes linear velocity damping into effect. Constant control is assumed over a time step. Only reasonably valued controls are considered and maximum values on angular rate and velocity are enforced.

III. GUIDED EXPANSIVE SEARCH TREES

Guided EST is a variation of conventional probabilistic path planning strategies: PRM [12], RRT [16], and most similarly EST [9], [8]. Guided ESTs have advantages over the conventional techniques when the state space is governed by higher dimensional kinodynamic constraints and when minimizing the control cost of the entire path is important. Guided ESTs borrow from conventional expansion techniques for robustness in finding paths, and also use path cost statistics to guide the tree in the window of acceptably low-cost paths.

Conventional probabilistic path planning algorithms rely on a metric to determine whether two configurations are "close." But in kinodynamic state spaces, it is not obvious when two configurations should be considered "close." Often a 1-norm or weighted 1-norm of the vector of the configuration is used in a kd-tree [7], or a Range Tree [7], or only the 1-norm of the static representation is used, ignoring the derivatives and time. These techniques do not necessarily accurately represent the reachability of the configuration–the intended reason for the metric. For instance, two configurations which are "close" based on a 1-norm metric will likely not be able to reach one another with a reasonable cost. Guided ESTs address some of these concerns.

A. Guided EST Algorithm

The basic algorithm is the same as the EST [9] algorithm, except that the weighting function is changed. The pseudo code below outlines the algorithm. At each iteration, an existing waypoint–a configuration in the state space–is chosen at random from a distribution, based on its weight. The waypoint is then expanded by applying a randomly chosen control to produce a new waypoint after some chosen time step. If the edge is in collision or if the new waypoint violates a velocity or rate constraint, then it is discarded. Valid waypoints are added to the tree and are assigned a weight. Then if the new waypoint can connect to the goal, the path is returned.

Algorithm 1 Guided ESTs

```
for i = 0 to N do
    p = choose_waypoint()
    n = expand_waypoint(p)
    if (n is not valid) : continue
    add_to_tree(p,n)
    assign_weight(n);
    if (n connects to goal) : return add_to_tree(n,goal)
end for
```

The weighting function for the EST algorithm only looks at the number of neighbors (*# neighbors*) within a range:

$$weight = \frac{1}{\#neighbors} \tag{5}$$

The guided EST algorithm differs in that it additionally takes into account the *out-degree*, number of out-going edges from the waypoint; the *order* of the waypoint, how recently it was created; and the *A* cost*, estimated total cost to the goal computed as the sum of the control required to reach the waypoint from the root and the estimated control cost to reach the goal. These statistics are taken to the power, α, β, γ, and δ, respectively.

$$weight = \frac{(order)^{\gamma}}{(\#neighbors)^{\alpha} \cdot (out-degree)^{\beta} \cdot (A^*cost)^{\delta}} \tag{6}$$

The *out-degree* term prevents a highly weighted waypoint from being expanded too many times. This is incremented even if the expanded-to waypoint is not valid. The *A* cost* term focuses the search towards the goal and prevents the tree from often expanding high cost waypoints which already are likely to expand into waypoints which violate a velocity or rate constraint. The *order* term, like the number of neighbors term, tends to keep the tree expanding on the frontier. For some extremely high dimensional problems [14] when proximity queries are unrealizable because it is too expensive or no good metric exists, the algorithm can still be effective after dropping the number of neighbors term from (6). However, if a good metric and way to efficiently perform proximity queries exist, then using the number of neighbors is more robust.

In Figure 1 the way the guided EST algorithm focuses the search (on the right) is compared to EST (on the left). Note how guided EST finds the goal with much less space searched.

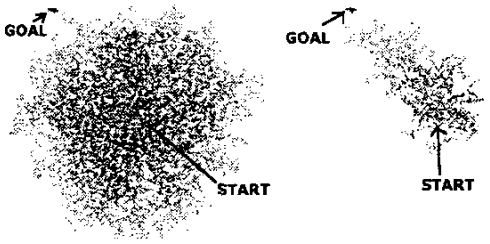

Fig. 1. 2D EST and guided EST : darker edges created first

B. Proximity Queries

The purpose of using the number of neighboring waypoints in the weighting function is to bias the search towards expanding on the frontier. So, whether two waypoints are close should be defined based on the control cost between the two configurations. This indicates how likely one waypoint is to expand into the region of the other. Conventional data structures for quick proximity queries, kd-trees [7] and Range Trees [7], can not handle such a distance function. Metric Trees [5], however, can make proximity queries efficiently with an arbitrary metric. Unfortunately, control cost is not a metric because it violates symmetry and the triangle inequality. In fact, Metric Trees only returned between 70% and 100% of the data within a range on random inputs, substituting control cost as the metric. However, the missing neighbors did not seem to qualitatively effect the tree expansion in a 3D visualizer and did not seem to perform any different with the same number of waypoints compared to a brute force approach.

Experimentally we found that metric trees needed rebalancing often, causing their amortized query time to be slow. This problem can be alleviated and query time reduced by building several metric trees indexed by different time ranges. Because waypoints are limited to only expanding within a limited time frame, only neighbors within a limited time range need to be considered and thus a limited number of our time-indexed Metric Trees. This also reduces the amount of rebalancing because the center of mass of a Metric Tree within a limited time range tends to drift less. Time-indexed Metric Trees with a cost distance function are used for all experimental results.

IV. PATH GRADIENT DESCENT

Although guided ESTs produce cost-constrained paths, applications such as the space shuttle docking on the space station want to minimize the cost function. Preexisting paths can be refined to obtain lower cost paths by calculating and following the gradient of the cost function. An entire path has many variables over which to minimize, but by analyzing each waypoint on the path individually and incrementally following that waypoint's gradient, the approach becomes manageable.

A. Path Representation

A path is stored as an array of waypoints, the controls applied to them, and the time between controls. Figure 2 shows a cartoon of a path segment where the circles are waypoints, wp, the solid lines, u, are impulsive controls instantaneously changing the derivative part of the configuration at the waypoint, and the dotted lines are the integration of the change of configuration over time. Alternatively, the controls and the integrated path can be combined for continuous control systems, such as for a blimp.

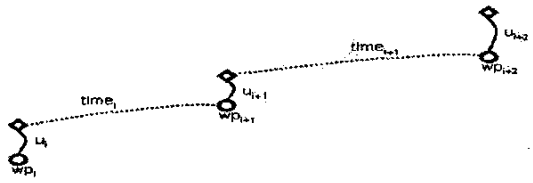

Fig. 2. Path segment

B. Path Cost Function

The cost function (7) which favors paths a safe distance from obstacles and low control cost is the sum of a term to penalize being near obstacles and a term to penalize for high control cost. The cost function can be written in terms of the controls applied and elapsed time and its gradient is taken in terms of the control.

$$cost_{wp_{i+1}}(u_i) = \begin{array}{l} avoid(obstacles, wp_{i+1}(u_i)) \\ +control(u_i) \end{array} \tag{7}$$

In the case where the propagation equations are linear, like the CW equations [6], the control cost part of the function can be calculated explicitly. An arbitrary set of linear equations ϕ which propagate a kinodynamic configuration forward as a function of time, can be written generically:

$$\begin{bmatrix} x_f \\ \dot{x}_f \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix} \begin{bmatrix} x_i \\ \dot{x}_i \end{bmatrix} \tag{8}$$

The *control* function takes into account the control applied at the particular waypoint and at its neighboring waypoints. Consider the waypoint labeled wp_{i+1} in Figure 2. When wp_{i+1} is perturbed, the controls u_i, u_{i+1}, u_{i+2} must be altered. u_i will determine the value of wp_{i+1}, then u_{i+1} can be solved for to extend wp_{i+1} to reach the position at wp_{i+2} and u_{i+2} can be solved for to appropriately adjust the derivative part of wp_{i+2}:

$$\begin{aligned} u_{i+1} &= \phi_{12}^{-1}(x_{i+2} - \phi_{11}x_{i+1}) - \dot{x}_{i+1} \\ x_{i+1} &= \phi_{11}x_i + \phi_{12}(\dot{x}_i + u_i) \\ \dot{x}_{i+1} &= \phi_{21}x_i + \phi_{22}(\dot{x}_i + u_i) \\ u_{i+2} &= \dot{x}_{i+2} + u_{i+2}\prime - \phi_{21}x_{i+1} - \phi_{22}(\dot{x}_{i+1} + u_{i+1}). \end{aligned} \tag{9}$$

$u_{i+2}\prime$ is the original value of the third control.

The *avoid* function is illustrated in Figure 3. Previous work in deforming paths [17], [4] propelled obstacles from waypoints and also put an elastic force on the path segments

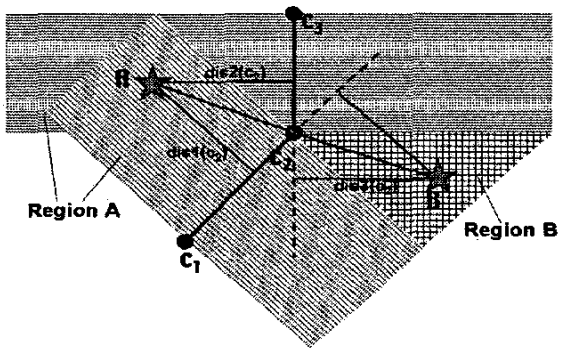

Fig. 3. Graphical representation of obstacle avoidance function calculation. $\overline{c_1c_2}$ and $\overline{c_2c_3}$ are path segments. A and B are obstacles.

between waypoints to prevent the path from stretching too far. By ensuring that the obstacles remain outside an ε-ball surrounding each waypoint and ensuring that the path remained within this series of ε-balls, no collision will occur. However, the elasticity of the path has no direct meaning in our cost function and should be avoided. Without the elastic term, if an obstacle is close to the path between two waypoints and about equidistant from both waypoints, the cost gradient will force the waypoints apart, but not dramatically increase the actual path clearance from the obstacle.

Our solution simplifies the path to a series of line segments between waypoints ($\overline{c_1c_2}$ and $\overline{c_2c_3}$ in Figure 3), and propels obstacles (A and B in Figure 3) from the line segments. By guaranteeing the actual path stays within some δ-distance of the line segment and that the obstacles stay further than δ from the line segment, it can guarantee no collisions.

In this approach the waypoint only has a repelling cost on the obstacle if the edge's closest point to the obstacle is between the waypoints (Region A in Figure 3), or if both edges would have their closest point to the obstacle if they were extended past the waypoint (Region B in Figure 3). These regions can be easily calculated with dot products over vectors. In Equation (10) $\overrightarrow{v}$ is a vector and $\overline{u}$ is a unit vector.

$$\begin{aligned}
\overrightarrow{dis1(c_2)} &= \overrightarrow{(c_2-A)} - (\overrightarrow{(c_2-A)} \cdot \overline{(c_2-c_1)})\overline{(c_2-A)} \\
\overrightarrow{dis2(c_2)} &= \overrightarrow{(c_2-A)} - (\overrightarrow{(c_2-A)} \cdot \overline{(c_2-c_3)})\overline{(c_2-A)} \\
\overrightarrow{dis3(c_2)} &= \overrightarrow{(c_2-B)} - (\overrightarrow{(c_2-B)} \cdot \overline{(c_1-c_2)})\overline{(c_2-B)} + \\
&\quad \overrightarrow{(c_2-B)} - (\overrightarrow{(c_2-B)} \cdot \overline{(c_3-c_2)})\overline{(c_2-B)} \\
\overrightarrow{grad(c_2)} &= \nabla \frac{1}{\overrightarrow{dis1(c_2)}^2} + \nabla \frac{1}{\overrightarrow{dis2(c_2)}^2} + \nabla \frac{1}{\overrightarrow{dis3(c_2)}^2}
\end{aligned} \tag{10}$$

C. Algorithm

To minimize the cost of the path, the gradient direction of each waypoint is independently calculated, with its neighbors fixed, and is followed a short distance ε. The process is iterated N times, as is shown in the pseudocode below:

The order the waypoints are visited is randomly permuted to prevent bias in how the path deforms.

Algorithm 2 Path Gradient Descent

```
for i = 0 to N do
    randomly permute order of waypoints
    for j = 0 to # waypoints do
        calculate cost gradient
        follow the direction of the gradient a distance ε
    end for
end for
```

D. Dynamic Obstacle Avoidance

The path gradient descent technique can be extended to work in a changing environment simply by updating and adjusting the cost function. It can be used as a feedback control loop to account for errors in propagation or to avoid dynamically changing obstacles. As the cost function is updated, the path will immediately deform to avoid obstacles and then reconverge to the local minimum–the path will remain within the same homotopy class. But a collision free path can be extracted at anytime, so this can be used for real-time obstacle avoidance.

V. EXPERIMENTS

Guided ESTs were tested using a variety of parameters, including those which represent ESTs, applied to (a) the space shuttle docking, to (b) maneuvering a blimp in a factory setting, and in (c) a simple 2D environment. For each experiment, variations of the weight function (6) were tested on 50 independent trials. Parameter ranges were as follows *#neighbors* $\{\alpha : 0-6\}$; *out-degree* $\{\beta : 0-3\}$; *order* $\{\gamma : 0-3\}$; *A* cost* $\{\delta : 0-4\}$. Restrictions were enforced to bound the total control cost of the path–the sum of all controls applied–or in the case of the 2D environment the total path length. If this restriction was violated, the configuration was rejected as if in collision.

In general, values of β and γ in equation (6) are most effective at 2 or 3. α is more effective near 1 for more cost-constrained and less obstacle-constrained systems, and more effective near 4 or 5 for less cost-constrained and more obstacle-constrained systems. Inversely, δ was more effective near 4 for more cost-constrained and less obstacle-constrained systems, and more effective near 1 for less cost-constrained and more obstacle-constrained systems.

An RRT [16] implementation was tested on the same experiments using the same primitives and data structures when appropriate. It performed well on the 2D environment, but we were unsuccessful in finding solution paths with the RRT on the cost-constrained examples. The problem seemed to be in choosing a bound on the state space to sample from and in trying to move towards random samples of the state space. Although individual parameters may be reasonable, parameter combinations are not always reasonable, and it is hard to define the set of reasonable combinations of parameters in the state space as opposed to setting ranges for each parameter individually.

All tests were run on AMD 1GHz processors.

A. Space Shuttle Docking

In this series of experiments, the space shuttle begins 1000 by 1000 by 1000 feet away and rotated 180 degrees from its docking position on the space station. Fifteen moving asteroids are generated randomly such that they do not collide with the space station but will be within 250 feet of it at the time the shuttle is scheduled to approach. The simulation plans a path for the space shuttle according to the kinodynamic equations of motion shown in (1). Controls were chosen uniformly at random from a reasonable range. However 15% of the controls were chosen to go directly towards the goal, as in [16], by symbolically inverting the matrix in (1) and setting the final configuration to the goal. This is also how the A* cost is calculated.

Several combinations successfully planned a path 100% of the time. The most efficient combination of parameters set $\alpha = 1$, $\beta = 2$, $\gamma = 3$, and $\delta = 3$ in equation (6) and found a path in an average of .254 seconds. However, a few parameters set $\alpha = 0$, thus not using the sampling density information, even though the metrics trees were still maintained, and queried. These operations represent a significant amount of the algorithm's runtime. When assigning $\alpha = 0$, $\beta = 3$, $\gamma = 2$, and $\delta = 2$ in equation (6) 100% of the runs found paths in an average time of 1.23 seconds.

Of the parameter sets that solved 100% or 98% of the systems, the assignment of $\alpha = 1$, $\beta = 1$, $\gamma = 1$, and $\delta = 4$ in equation (6) had the lowest average path cost, and solved in an average of 1.20 seconds. Of the parameter assignments with the lowest costs, it was indicative for $\alpha = \{0, 1\}$ and $\delta = \{3, 4\}$.

Comparatively, the parameter assignment representing EST, $\alpha = 1$, $\beta = 0$, $\gamma = 0$, and $\delta = 0$ in equation (6), only returned a path 2% of the time and of those paths returned, the average cost of successful paths was about two times that of the average cost for above parameters and required an average search time of 3.46 seconds. Note in Figure 4 how the EST searches much more space (the image is zoomed out), but most of the this extra coverage is in the wrong direction.

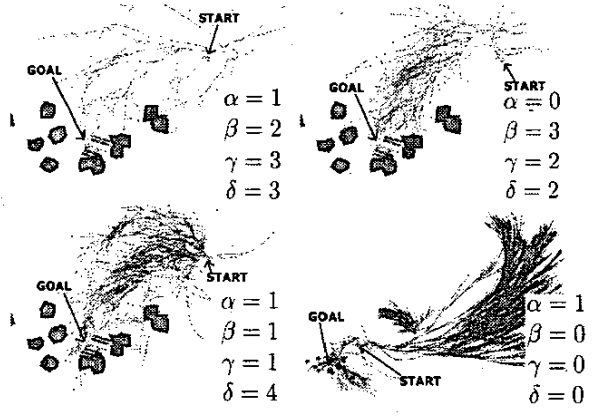

Fig. 4. Guided ESTs for space shuttle docking, with varying parameters

B. Maneuvering a Blimp in Factory Setting

A factory-type setting is simulated as a hallway mazed with pipes that a 4-foot blimp needs to maneuver 80 feet through and rotate 180 degrees. The movement of the blimp is governed by the kinodynamic equations described in (2), (3), and (4). For propagation, controls u_1 and u_3 are chosen uniformly at randomly from a reasonable range, then u_2 is chosen uniformly at randomly from some smaller range as a difference from u_1. If it is chosen independently of u_1, the blimp will too often end up with too much angular rate. Estimating the A* cost is harder for this nonlinear problem. The amount of control required to independently appropriately direct the displacement velocities if orientated correctly and to orient yaw velocities correctly are used. Again, bounding restrictions were put on angular rate, velocities, and total path cost, and violating configurations were rejected from the tree. The blimp's requirement for minimal control cost is not as much of an issue as with the space shuttle, but these bounds tend to keep the blimp from getting in hard to control spins and from reaching unreasonable velocities. In the end, this leads to straighter, more desirable paths.

The most efficient sets of parameters found paths 66% of the time. The assignment $\alpha = 6$, $\beta = 3$, $\gamma = 1$, and $\delta = 4$ in equation (6) found paths 66% of the time in an average of 3.29 seconds. Values for α of 5 or 6 and for δ of 3 or 4 was indicative of the sets of parameters which had high percentage of successful runs. In contrast to the shuttle example, no set of parameters with $\alpha = 0$ found a path on more than 28% of the trials. These times are much higher than those of the space shuttle experiment mainly because the propagation function needs to use numerical integration to extend a path.

Of the parameter sets which found paths more than 60% of the time, the set $\alpha = 5$, $\beta = 2$, $\gamma = 0$, and $\delta = 4$ in equation (6) had the lowest average cost of the paths found. A δ value of 4 was common among the sets of parameters with low average path costs.

In comparison, the set of parameters representing the standard EST scheme only solved for paths 18% of the time. The paths were found in an average of 5.53 seconds and with an average cost of 3 times that of the best one described above.

Figure 5 shows from above a tree which found a path for the blimp in a randomly generated factory environment.

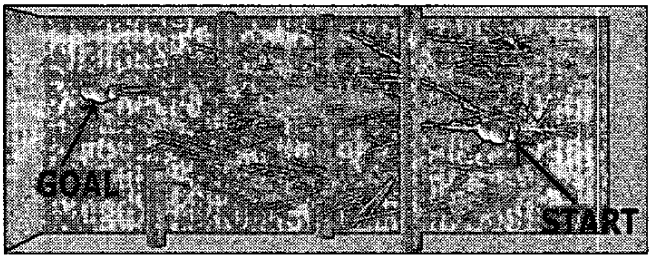

Fig. 5. Guided EST for blimp in factory environment. Darker edges for yaw at goal, lighter edges for yaw at start

C. 2D Environment

Experiments varying the same parameters were run in the 2D environment in Figure 6. Euclidean distance was used as cost for the purpose of calculating the A* cost. The most time efficient sets of parameters solved for a path in about .5 seconds and assigned $\alpha = \{5,6\}$ and $\delta = \{0.1\}$. Although the trees with $\delta \geq 1$ fruitlessly, almost certainly explored the large region on the right first, when that region was explored, they were able to find the passage on the bottom left and then reached the goal about as quickly as trees with $\delta = 0$. An example tree with $\delta = 1$ can be seen in Figure 6.

This algorithm with $\delta \geq 3$ and $\alpha \leq 4$ in Equation (6) will search primarily in the region on the right and will rarely make it through the passage at the bottom. This example does not benefit from the guidance of the A* cost as does a higher dimensional problem with kinodynamic constraints.

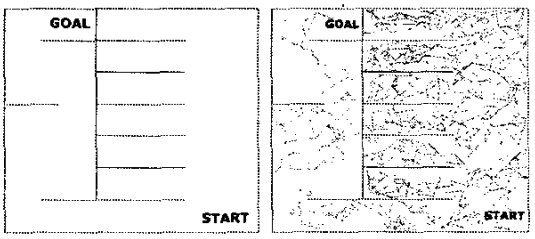

Fig. 6. 2D Environment: empty and explored : darker edges created first

D. Path Gradient Descent

Path gradient descent was tested on 50 paths generated by the docking space shuttle avoiding 15 moving obstacles using the parameters $\{\alpha = 1, \beta = 2, \gamma = 3, \delta = 3\}$ in Equation (6). With 10, 20, and 100 iterations, the cost dropped to 52%, 44%, and 37% of the original cost found by the guided EST algorithm, respectively. Most of the improvement is realized in the first few iterations. The path gradient descent took an average of .006, .011, and .056 seconds for each number of iterations, respectively.

VI. DISCUSSION AND FUTURE WORK

Guided ESTs are an improvement to the repertoire of motion planning tools and are particularly effective for a hard class of kinodynamic problems with a path cost function. The path gradient descent algorithm provides a tool for further refining cost driven paths.

Guided ESTs without the use of nearest neighbors proximity queries can be particularly useful for high dimensional problems [14] as well as problems where it is expensive or not clear how to define a neighborhood.

Although the most effective sets of parameters in equation (6) vary from one system to another, fixing the values of β and γ at 2 would not change the expansion results significantly. α and δ need to be adjusted to the specific system based on certain criteria. Higher values of α seem to be more effective when there are narrower passages described by physical obstacles, and higher values of δ seem to be more effective with more constraints on the motion. Potentially, an adaptive system could be devised to occasionally readjust the weightings based on new parameter values which are determined by measuring the number of collisions with obstacles to tune α and the number of constraint violations to tune δ.

Existing path planning techniques are shown to have difficulty with the class of motion- and cost-constrained systems presented. By using a directed sampling technique, guided ESTs perform much better in these constrained systems. This indicates that guided ESTs provide an important tool for motion planning and that different sampling techniques need to be considered.

VII. ACKNOWLEDGMENTS

JP would like to thank Andrew Ladd and the rest of the Physical and Biological Computing Group of Rice University for much constructive feedback. Work on the paper by Jeff Phillips has been partially supported by Draper Laboratories and by NSF 9702288. Work on this paper by Lydia Kavraki has been supported in part by NSF 0308237 and a Sloan Fellowship to Lydia Kavraki.

REFERENCES

[1] D. Balkcom and M. Mason. Time optimal trajectory for bounded velocity differential drive robots. *IEEE International Conference on Robotics and Automation*, 2000.

[2] Y. Bestaoui, S. Hima, and C. Sentouh. Motion planning of a fully actuated unmanned aerial vehicle. *AIAA Guidance, Navigation, and Control*, 2003.

[3] R. Bohlin and L. E. Kavraki. A randomized algorithm for robot path planning based on lazy evaluation. *Handbook on Randomized Computing*, pages 221–249, 2001.

[4] O. Brock and O. Khatib. Elastic strips: Real-time path modification for mobile manipulation. *Robotics Research*, pages 5–13, 1998. Springer-Verlag.

[5] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity in metric spaces. *the VLDB Journal*, 1997.

[6] W. H. Clohessy and R. S. Wiltshire. Terminal guidance system for satellite rendezvous. *J. of the Aerospace Sciences*, 27:653–658, 1960.

[7] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Comptational Geometry: Algorithms and Applications*. Springer, 1997.

[8] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *International Workshop on Algorithmic Foudations of Robotics*, pages 233–255, 2000.

[9] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *International Journal of Computational Geometry and Applications*, 9(4-5):495–512, 1999.

[10] M. C. Jackson. A six degree of freedom, plume-fuel optimal trajectory planner for spacecraft proximity operations using an a* node search. Master's thesis, MIT, 1994.

[11] H. Z. James and J. Ostrowski. Visual servoing with dynamics: Control of an unmanned blimp. *IEEE International Conference of Robotics and Automation*, 1999.

[12] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE International Transactions on Robotics and Automation*, 12(4):566–580, June 1996.

[13] C. Kluever. Feedback control for spacecraft rednzvous and docking. *Journal of Guidance, Control, and Dynamics*, 22(4):609–611, 1999.

[14] A. M. Ladd and L. E. Kavraki. Using motion planning for knot untangling. *International Workshop on Algorithmic Foudations of Robotics*, 2002.

[15] F. Lamiraux and D. Bonnafous. Reactive trajectory deformation for nonholonomic systems: Applications to mobile robots. *IEEE International Conference on Robotics and Automation*, pages 3099–3104, 2002.

[16] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

[17] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. *IEEE International Conference on Robotics and Automation*, 2:802–807, 1993.

[18] James Shoemaker. "Orbital Express Space Operations Architecture". URL: http://www.darpa.mil/tto/programs/astro.html [cited 19 January 2004].