Planning With Uncertainty for Autonomous UAV

Sameer Ansari Billy Gallagher Kyel Ok William Sica

***Abstract*—The increasing usage of autonomous UAV's in military and civilian applications requires accompanying algorithms to ensure successful operation. Mapping with UAV's using SLAM is a popular area of research that allows maps to be generated of areas that might be unnavigable for ground based vehicles. SLAM does not prevent collisions in the environment, or bias the robot towards a goal. In the case of uncertain environments such as a forest, these problems could significantly impact the performance of the UAV. In this paper, we address these problems by adding a planning approach that moves the robot towards a goal while avoiding collisions. This approach uses a global Voronoi planner to find a path to the goal that maximizes distance from uncertain obstacles to prevent collision. Local gradient descent in potential fields is performed while traversing the Voronoi vertices to provide more stable navigation decisions.**

I. Introduction

There exist numerous applications that could benefit from the use of a UAV that could autonomously navigate through dense environments such as a forest. Search and rescue of stranded or injured hikers could be accelerated, as well as more quickly finding the wreckage or survivors of an aircraft crash. In addition, finding people trapped by forest fires, or warning people about forest fires, could be easier using a UAV. There are many other situations in which a UAV that could autonomously navigate through a forest to some goal would be beneficial.

Performing autonomous navigation of a UAV through an unmapped forest presents several unique challenges. Forest environments contain many trees, which become small and densely concentrated obstacles. Typically, tree-level maps of forests do not exist, because they are difficult to build without navigating through the forest and change frequently. Therefore, the UAV must detect and map the obstacles it encounters as it travels. In addition, GPS signal may be unavailable due to the density of the forest canopy, so the robot must also continually localize itself. Finally, to reach a desired goal, the robot must additionally be able to continually plan to find the best route.

Autonomous navigation in an environment with many obstacles can be achieved through a variety of planning algorithms, and has many applications. However, these require a priori knowledge of the environment and all obstacles. For the case of navigating through an unknown forest, this is not the case, as the location of obstacles is unknown initially. Simultaneous Localization and Mapping (SLAM) algorithms exist that can track a robot's position and create a map of encountered obstacles at the same time. However, these methods typically do not incorporate any type of planner, as they are intended simply for building a map, and not accomplishing some task. In addition, as the algorithm builds the map, noise in sensor readings introduces uncertainty into the location of obstacles, which few planning algorithms handle.

These two methods can be combined to create an algorithm for Simultaneous Planning, Localization, and Mapping (SPLAM), which can create a plan to reach some goal based on the currently known state of the obstacles and their uncertainties, and will continually update this plan as the knowledge of the environment changes. Both planning and SLAM endeavor to thoroughly explore a space to accomplish the desired purpose. SPLAM methods, however, only seek to explore the region of the environment that is required to create a successful plan, leaving some unnecessary regions unexplored.

For this study, the planning algorithms of a SPLAM system will be analyzed. The algorithm must be able to successfully determine the shortest path to some given goal through the known obstacles. It must have a method for staying further from obstacles with high uncertainty, while being able to approach more certain obstacles more closely. Finally, it must be able to adapt to changes in the environment as the robot makes more accurate maps and finds new obstacles.

II. Related Work

A. Simulatenous Localization and Mapping (SLAM)

The premise of SLAM is to place a robot in an unknown location and then to have the robot map this region, localizing itself based on observations to autonomously navigate the world [1]. While practical solutions exist to this problem [2], in this project the robot is biased towards a known goal. SLAM does not include a planning component, so while the entire area could be mapped and then later navigated towards a goal, in time-critical situations it is useful to move towards the goal as a map is built.

B. Simultaneous Planning, Localization, and Mapping (SPLAM)

SPLAM (also known as active SLAM) combines SLAM with a planning approach, using information about the map being built by navigation to influence decisions. This can be used to bias navigation for more complete maps, or as a way to build a representation of the world in uncertain environments. SPLAM is currently domain-specific, so existing approaches [3]–[6] to the topic are insufficient to be applied directly to the problem of navigation in an uncertain environment. The planning approaches used in these methods vary from RRTs to POMDPs, and at the time of writing there is not a standard method in the field.

C. Potential Fields

Potential fields are used in motion planning to bias the robot or manipulator away from obstacles [7]. By performing a gradient descent, a plan can be generated towards the goal even for manipulators with many degrees of freedom. The problem with this approach is that it is not guaranteed to be complete, as when a local minima is reached in the potential field during gradient descent the algorithm will be stuck in place.

D. Voronoi Diagrams

Voronoi decomposition of a region draws polygons around points or polygons in the space such that the edges of the polygons maximize distance between points in the region [8]. This approach has been applied in motion planning [9] for obstacle avoidance, as navigating the maximum distance from obstacles is useful for avoiding collision.

E. Combining Voronoi Diagrams with Potential Fields

Previous work has solved the local minima problem of potential fields using Voronoi diagrams. A Voronoi fields approach has been used in the DARPA challenge for a robot navigating traffic [10]. The domain for this problem differs in the fact that the most common obstacle in the environment is a tree, which is relatively stationary. By mapping the area as the robot traverses toward the goal, this map can be used for planning when backtracking is required to reach the goal.

III. Methods

A. Hierarchical Design

To accomplish the task of planning within SPLAM, the planner must be able to both succesfully search the known global environment for the best path as well a navigate around the obstacles with uncertainty in the local environment. Very few planners exist that can efficiently handle both, so a hierarchical system was designed that used two different planning algorithms to handle the global and local environments seperately. Figure 1 shows the basic concept of the system. The lowest level, which runs at the highest frequency, consists of the robot controller, which, given a waypoint, will navigate the robot to it. Above that, the local planner runs less often and, given a local goal point, provides a series of waypoints to the controller that will best avoid the obstacles around the robot. At the highest level, the global planner, which is given the global goal, provides the best path to it and chooses the most approrpiate local goal for the local planner.

B. Local Planner

The local plaanner takes in the list of the currently visible obstacles and their uncertainties, as well as the local goal, and finds a path from the robot's position that will attempt to reach the local goal. It does this by calculating a potential field and using a gradient descent search to calculate the path.

The potential field is comprised of three components: 1) the linear distance from the local goal, 2) the value of a Gaussian distribution based on the distance from an obstacle and its uncertainty, 3) the linear distance from an obstacle. Component 1 ensures that the potential field will push the robot towards its intended goal. Component 2 is given by Equation 1, where σ is indicative of the uncertainty in the obstacle's position and d is the distance from to the obstacle. This ensures that more uncertain obstacles will push the robot further away than well localized ones. Finally, Component 3 will ensure that the robot won't get too close to any obstacles, including ones with little to no uncertainty. Each component is weighted to ensure a smooth and consistent continuous function.

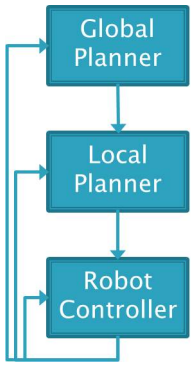

Fig. 1: Conceptual hierarchy of planning system

$$p = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{d^2}{2\sigma^2}} \tag{1}$$

Gradient descent methods find the minimum of a function by gradually stepping down the steepest slope iteratively. This is done by estimating the gradient, ∇f, of the function at each step, and taking a step in the $-\frac{\nabla f}{|\nabla f|}$ direction with a step size based on the magnitude of the gradient. This continues until the gradient goes to zero, the step size gets extremely small, or the robot reachs its goal. This results in a path that reaches the minimum most rapidly, and will avoid obstacles as best as possible.

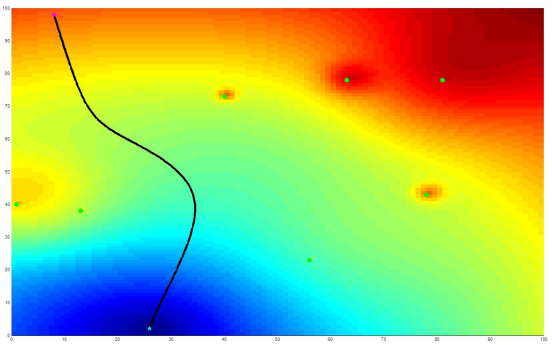

Fig. 2: Example potential field

This method, however, will not traverse uphill. Therefore, it will get trapped by local minima of the potential function.

Typically, this is considered problematic, as it means the algorithm will not find a path to the goal. However, the local minima will prevent the robot from colliding with intervening obstacles between it and its goal, as shown by the example in Figure 3. As the robot moves and the uncertainty of the obstacles decreases, the higher level global planner will find new paths, which will result in a local goal that will bypass any local minima.

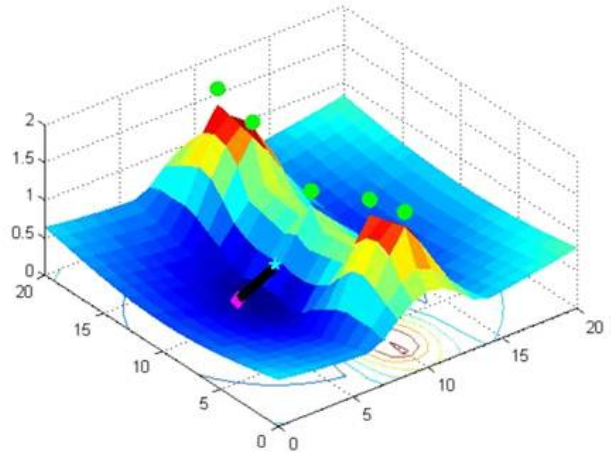

Fig. 3: Gradient descent method trapped by local minimum, prevent from hitting wall of obstacles

C. Global Planner

As a solution to the local minima problem of the potential field planner, a global planner is implemented to guide the algorithm. Voronoi tessellation on the Euclidean space can produce Voronoi vertices, which are the optimally farthest point from nearby obstacles, and Voronoi edges that connect every Voronoi vertex. Since potential field exerts forces linearly proportional to the distance to the obstacles, and the Voronoi vertices are also farthest distanced points from the obstacles, the local minima in the fields and the Voronoi vertices are coincident. Using this property, and the latter property that all Voronoi vertices are connected, it can be shown that following the Voronoi vertices guarantees a path to the goal as a sequence of Voronoi edges between the vertices. Also, the vertices are locally optimal in terms of distance to the obstacles.

Thus, the global planner will generate Voronoi vertices and use A* search on the vertices to find the shortest path to the global goal. This path is then passed to the local planner to follow as waypoints. The resulting behavior is the robot traversing from one local minimum to another a safe distance away from the obstacles to ensure no collisions with the obstacles occur. The global planner will give the shortest euclidean distance to the goal when no obstacles are observed.

D. Integrated System

The final system consists of 4 major components: SLAM, controller, local planner, global planner.

The SLAM and controller are simulated with the assumption that we have the ground truth of the position of the robot. The sensors used to detect the obstacles are to have 2d gaussian noise in the x-y- directions. The controller is assumed to be able to navigate a robot from one position to another with no error. The global planner updates the Voronoi vertex way point every significant changes in the uncertainty of the obstacles occur as they converge. The global planner updates the Voronoi vertex waypoints when the uncertainty of the obstacles changes significantly, while the local planner updates every iteration.

A typical cycle would start with updating the local position and the estimated position of the obstacles in the SLAM segment. Then the position of the robot and the obstacles were passed to the global planner along with the global goal. The global planner would do the Voronoi tessellation on the space and A* search to find the path to the global goal. The closest way point on the path is used as a local goal for the local planner. Doing gradient decent on the potential field map of the local space, a path is found for the robot to follow.

E. Software Used

The project was coded in MATLAB with A* search performed using compiled C++ MEX libraries.

IV. Experiments

A random set of obstacles was generated in an environment and tested to see if the robot could navigate between the start and goal location while avoiding collisions. Testing was performed across a range of 0 to 10,000 obstacle scenarios. In addition to testing with randomly generated obstacles, to show completeness testing was performed in situations where no valid path existed to ensure the planner was able to terminate, and did not attempt a path that might cause a collision.

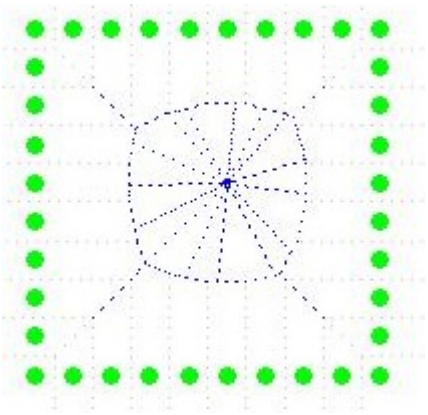

Fig. 4: Termination in scenario with no solution

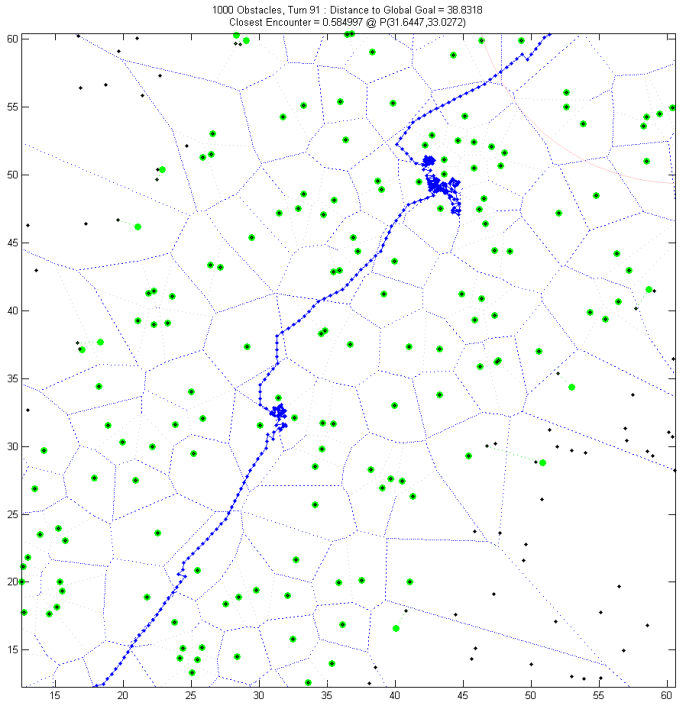

Fig. 5: The global planner navigates out of a local minima in this scenario

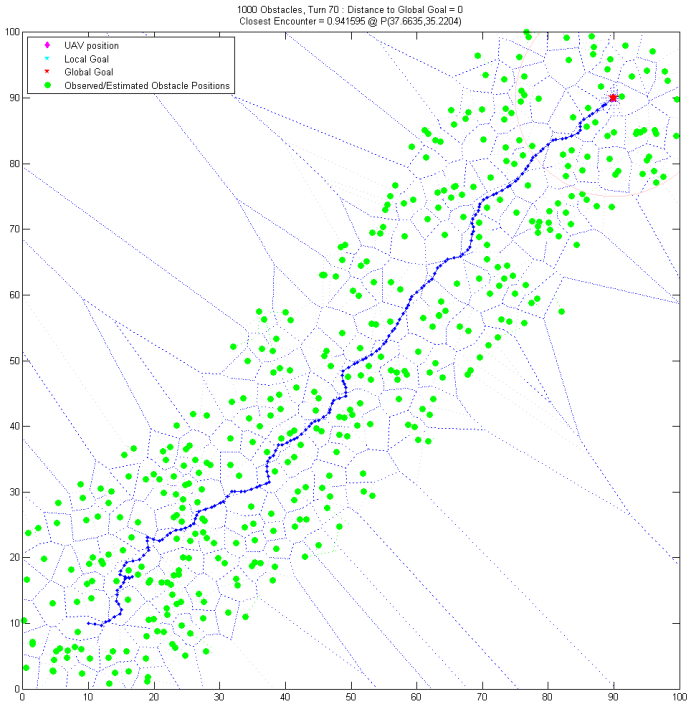

Fig. 6: 1,000 obstacle scenario

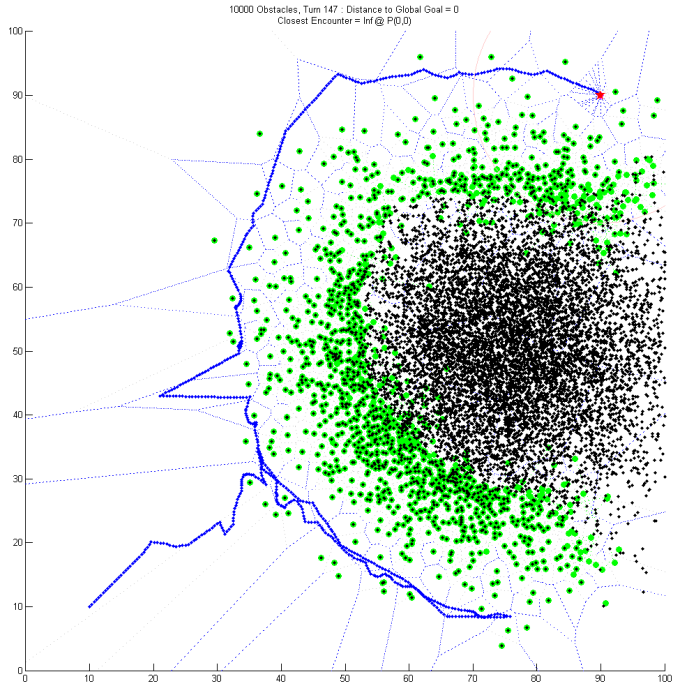

Fig. 7: 10,000 obstacle scenario

V. Analysis

A. *Completeness*

To prove that a planning algorithm is complete it must be shown that:

1) When no solution exists, the planner terminates
2) Given that a solution exists, the planner returns it in finite time

Completeness can be shown for the global planner, which is used to determine whether or not the program should terminate. After performing Voronoi decomposition on the region, the maximum distance between obstacles is obtained. By drawing the points of a regular polygon around the robot and goal, Voronoi vertices are guaranteed to be present at the location of the robot and goal which are connected by the properties of Voronoi decomposition. Edges which are not traversable, and whose distance to obstacles would cause collisions, are cut from the graph. In this modified graph, the assumption that the goal and robot are connected may not be true.

In order to check that a path still exists, a breadth first search approach is used to enumerate the vertices connected to the robot. If the goal vertex is not in this enumeration, the robot and goal are not connected by a collision free path. While this provides reason to terminate for a single timestep, the obstacles in the planning domain have an uncertain location. The robot waits and collects more observations in this scenario, until the uncertainty of the obstacles location converges below a threshold. If no path still exists, the planner terminates and returns that there is no solution.

To show that the planner finds a solution when a solution exists, A* search is performed in the graph to locate a collision free path between the robot and goal. Assuming A* runs in finite-time, the planner is complete.

B. Optimality

Optimality is not guaranteed for the global plan. Because the representation of the world is constantly updating as the planner progresses, it is not possible to have an optimal solution that is guaranteed to be collision free before the region has been explored. The A* search provided the shortest path to the goal, providing the euclidean distance heuristic is admissible. However, this is the shortest distance in the Voronoi region, which may not be the shortest path in the actual region.

For the local plan, gradient descent ensures a locally optimal solution assuming a local minima does not exist in the potential field representation.

C. Efficiency

The highest time-bound of the planning approach is the A* search, which is exponential on the number of obstacles. For practical applications, the number of obstacles that could be processed by the global Voronoi planner would need to be limited to provide acceptable runtimes in realtime scenarios.

D. Experimental Time Data

Timing tests were done on a basic setup. The field size is 100 by 100 meters divided into a grid from the origin in the bottom left corner viewed from above. The robot starts at the point (0,0) and the goal is at the point (90,90). Obstacles are distributed normally about the center of the field located at point (50,50). Timing was done on a per-turn basis, with an average over all turns.

The number of obstacles, N, was varied for 10 tests, and Table I shows the number of turns each round took, as well as the average over that round. As the number of obstacles increases, so does the time taken by both planners. However, the local potential field planner increases from a minimum of 0.036 seconds for 0 obstacles to 0.81 seconds for 500 obstacles, a single magnitude increase compared to a 0.0017 to 9.6 second change, or a threefold magnitude change for the global Voronoi planner. Both planners increase in time due to the increased obstacles computed. The global planner computes using all known obstacles including those outside of the current visible range, as previously visible obstacles were remembered, which increases over the course of the simulation. The local planner only interacts with visible obstacles on a local range, which is a subset of the entire field, and increases and decreases based on location, often a very small percentage of the field. For 200 obstacles, on average the local planner only interacts with 34.4 obstacles, whereas the global planner eventually interacted with 139 obstacles per turn at the end of the simulation.

Over the course of a simulation, the time of the local planner stays roughly constant as a result of these average interactions, but the global increases. Table II shows the difference in timing for the global and local planners at the beginning and end of the simulation, as well as when there is no uncertainty, how the Voronoi planner is not updated, which brings down the overall average time of the global planner versus the local planner, which always runs. The increase in global planning time, and ability to perform navigation in certain scenarios without the global planner, is illustrated in Figure 9.

Table I shows the ratio of time taken by the global planner versus the number of obstacles for different tests. As the obstacles increases, the ratio also increases, when shown in a plot on Figure 8, it appears to increase exponentially, showing that for the way the Voronoi decomposition is coded based on known obstacles, the relationship between number of obstacles and time taken is exponential. In Figure 8, the ratio of time taken for the local planner based on the number of obstacles remains relatively constant, as the area the local planner operates on remains fixed.

TABLE I: Time taken for planning

# of Obstacles	Total (s)	Global Plan (s)	Local Plan (s)
0	0.128	0.000	0.035
1	0.137	0.001	0.036
2	0.164	0.004	0.039
5	0.204	0.007	0.054
10	0.246	0.016	0.062
20	0.329	0.038	0.087
50	0.574	0.154	0.125
100	1.19	0.511	0.251
200	2.51	1.566	0.413
500	9.608	7.916	0.810

TABLE II: Time taken for planning as program operation progresses

50 obstacles			
Timestep	Total (s)	Global Plan (s)	Local Plan (s)
Beginning	1.150	0.424	0.360
No Voronoi	0.246	0.000	0.033
End	1.817	1.367	0.063
200 obstacles			
Timestep	Total (s)	Global Plan (s)	Local Plan (s)
Beginning	0.333	0.051	0.022
No Voronoi	1.06	0.000	0.262
End	12.233	10.477	1.025
500 obstacles			
Timestep	Total (s)	Global Plan (s)	Local Plan (s)
Beginning	1.151	0.425	0.361
No Voronoi	1.477	0.000	0.205
End	104.888	103.417	0.156

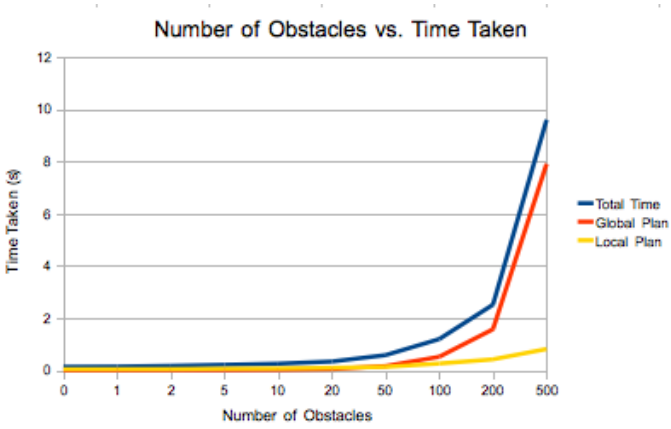

Fig. 8: Plot of time taken for each planner

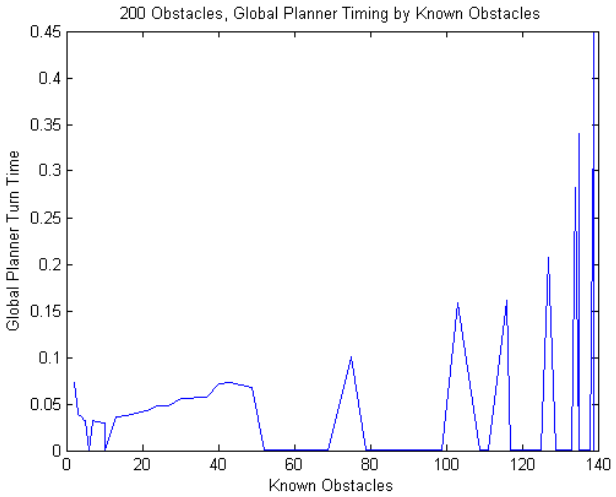

Fig. 9: Time of global planner over course of navigation

VI. Discussion

The results of this project show that collision free navigation of uncertain environments is possible. Results in Figure 7 show that the global map generated during the programs operation can be used successfully to replan when a dead end is encountered. In Figure 5, it is evidenced that the global Voronoi planner has the capability of moving the robot out of local minima in the gradient descent approach. Additionally, evidence is supplied in Figure 4 that the planner is able to correctly terminate when a collision free path to the goal is not present.

While the results are positive towards completeness, the time taken for program operation seen in Figure 8 show that operation of the planner can not be performed in real time for a large number of obstacles. In order to implement the system on a UAV, a solution would need to be created that allows the global planner to limit the number of obstacles it processes in each step while still enabling the capability to backtrack upon encountering a dead-end. One solution to this problem would be to further divide the hierarchical approach. The global planner could use a subset of obstacles, and only use the full global map when no solution exists in the local path. The choice of language for the project also limits the runtime of the program. Runtime speed could be further improved by recoding the project in C or C++ instead of MATLAB.

Extending the problem to execution on a UAV additionally requires solving the associated controls and vision problem, which were abstracted in the simulation of this project. While the controls problem is platform specific, the velocity of the vehicle must be factored into the planner to allow for plans that can be executed. It is possible to extend the planners used into higher dimension spaces, such that the velocity of the robot might be represented as another dimension and plans solved in this manner.

The vision problem is unique in that obstacles may be encountered in the environment that do not possess the behavior of those used in the simulation. As an example, the planner would have difficulty navigating around a moving obstacle as the uncertainty around it would not converge over time, assuming it remains mobile. While this may be ignorable for a forest environment, extending the planner to an urban environment could benefit from being able to handle obstacles with this type of behavior.

References

[1] P. Corke, J. Trevelyan, M. Dissanayake, P. Newman, H. Durrant-Whyte, S. Clark, and M. Csorba, "An experimental and theoretical investigation into simultaneous localisation and map building," in *Experimental Robotics VI*, vol. 250 of *Lecture Notes in Control and Information Sciences*, pp. 265–274, Springer Berlin / Heidelberg, 2000. 10.1007/BFb0119405.

[2] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 593–598, AAAI, 2002.

[3] C. Leung, S. Huang, and G. Dissanayake, "Active slam in structured environments," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 1898 –1903, may 2008.

[4] C. Leung, S. Huang, and G. Dissanayake, "Active slam using model predictive control and attractor based exploration," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 5026 – 5031, oct. 2006.

[5] R. Sim and N. Roy, "Global a-optimal robot exploration in slam," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 661 – 666, april 2005.

[6] T. Tao, Y. Huang, F. Sun, and T. Wang, "Motion planning for slam based on frontier exploration," in *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, pp. 2120 –2125, aug. 2007.

[7] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[8] D. G. Kirkpatrick, "Efficient computation of continuous skeletons," in *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, SFCS '79, (Washington, DC, USA), pp. 18–27, IEEE Computer Society, 1979.

[9] F. Aurenhammer, "Voronoi diagrams : a survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, pp. 345–405, September 1991.

[10] O. Khatib, V. Kumar, and G. J. Pappas, eds., *Experimental Robotics, The Eleventh International Symposium, ISER 2008, July 13-16, 2008, Athens, Greece*, vol. 54 of *Springer Tracts in Advanced Robotics*, Springer, 2009.