**Graduate Projects**

University of Colorado at Boulder

Aerospace Engineering Sciences

ASEN 5018/6028 –Fall 2015

|  |
| --- |
| **FlyNet**  **Planning Subsystem Summary/Continuity Document** |

|  |  |  |
| --- | --- | --- |
| **Approvers List** | | |
|  | Title | Name |
| Prepared By | Planning Team | Bryce Hill |
|  |  | Edward Meletyan |
| Approved By | Planning Lead | Bryce Hill |
| Approved By | Systems Engineer |  |
| Approved By | Project Manager | Drew Ellison |

**1: Introduction & Summary**

To complete our objective of searching a space for targets a path must be generated which: avoids obstacles, arrives at desired final destination and is possible to compute in real time environments.

The operational space is 3-dimensional. However, 3D planning algorithms were observed to require more than twenty seconds to compute. Therefore, the decision was made to use 2-dimensional algorithms which can be computed in milliseconds.

Two algorithms have been evaluated: the first is a hybrid Voronoi/A\* method which uses the Voronoi algorithm to generate points between obstacles and the A\* algorithm to plan the most efficient path from start to goal. The second path planning algorithm is a probabilistic road map (PRM) approach. The PRM creates a random sample of nodes on a map that are located in free space (i.e. not an obstacle). Each node is connected to every neighboring node that is within a configured threshold until a path can be made from a starting point to a goal point based on the shortest distance between connected nodes.

# **2: Semester Report**

## 2.1: Objectives and Tasks List

**Completed**:

1. Matlab implemented Voronoi/A\* 2D path planning algorithm
2. Matlab implemented probabilistic road map 2D path planning algorithm

**Incomplete**:

1. Test each planning algorithm in Matlab simulation
2. Implement planning algorithms in C/C++/Python for use on flight hardware
3. Interface with the flight controllers

## 2.2: Issues

1. Testing the planning algorithms in the Matlab simulation remains incomplete due to time constraints.

## 2.3: Lessons Learned

So you just identified some things that prevented you from completing some tasks – is there anything you can learn from that? Having more time to do things isn’t really a lesson learned, so anything like Issue 3 in my example above won’t really have a lesson learned. Remember: lessons learned are anything that you **wish you knew before**, something **learned the hard way**, or a **piece of advice**. Note that you don’t have to address each numbered issue above directly. Also note that you are encouraged to include lessons learned in the process of completing the tasks that you *did* finish.

1. Be very clear on the requirements and their implication prior to beginning implementation.
2. Although planning a path is useful, having a metric for where the robot needs to go is essential.

## 2.4: Procedures

Voronoi/A\*:

We began with Matlab’s built in Voronoi function. Voronoi provides a convenient way to ensure paths are equidistant between obstacles. In Figure 1 below, obstacles are represented by the ‘\*’ and the paths are drawn between vertices which the voronoi algorithm returns. These points are processed and a matrix holding the indices of neighboring nodes is constructed. Additionally, if paths are too close to obstacles they will be marked out of bounds.



Figure . Voronoi generated paths

Next, A\* was implemented in Matlab. A\* plans the path of nodes which will be navigated to reach the goal. In the following, Figure 2, the origin is represented as a circle, the goal as X. The paths with dashed lines (that aren’t red) have been eliminated due to their proximity to obstacles.



Figure . Voronoi/A\*

Probabilistic Roadmap (PRM)

The PRM approach requires an initial obstacle map, starting coordinates, and goal coordinates. The obstacle map is a two-dimensional array containing ‘0’ for coordinates that are empty space and ‘1’ for coordinates that are obstacles. Given this information, the PRM algorithm creates a random set of nodes in empty space.

The code for these algorithms can be found on the team GitHub: <https://github.com/dme722/FlyNet/tree/master/Technical/pathPlanner>

The individual algorithms are found in the sub directories of this folder.



Figure . Probabalistic Road Map

Table 2.: Software list

|  |  |  |
| --- | --- | --- |
| Program Name | Version | Purpose |
| Matlab | R2015a | Algorithmic development and testing |

# **3: Next Semester/Future Expectations**

## 3.1: Prioritized List of Tasks and Objectives

Accomplished

1. Evaluate 3D vs. 2D algorithms
2. Develop PRM and Voronoi/A\* algorithms in Matlab

Next Steps

1. Integrate algorithms with Matlab vehicle simulation
2. Convert algorithms to C
3. Integrate with onboard computer

## 3.2: Starting Points

1. In initial research it was found that a PRM algorithm took over 20 seconds to complete computing a path. As this was well above our requirements we opted to use a 2D path. This is likely an area that does not need revisited.
2. In the Voronoi/A\* algorithm found here: C:\Users\Bryce\Documents \FlyNet\Technical\pathPlanner, the space.m file could be updated. Voronoi generates points well outside the bounds and we have begun attempting to trim this down. While A\* should never select those points because they are much too far out of bounds it would be unwise to leave them there due to the risk that they entail.
3. In the directory: C:\Users\Bryce\Documents\FlyNet\Technical\controls \Quad\_Sim\_AA the simulation which requires an array of points constituting the path as input can be found. Simply generate points with the PRM and Voronoi/A\* algorithms and pass them into this simulation to evaluate performance.
4. This has not been started yet. First step would be to communicate with the integration team to iron out all inputs and outputs that will need to be made.
5. This will also require heavy coordination with integration team.

## 3.3: Improvement, Updates, Verification

The major improvements to be made next semester include merging the two algorithms developed this semester such that the advantages of each algorithm can be exploited. Additionally, the idea of space “coverage” and generating a list of desired goal map positions must be further explored.