DYNAMIC OBSTACLE CHARACTERIZATION AND AVOIDANCE FOR UNMANNED AERIAL SYSTEMS

Kyle Norland
Department of Systems and Industrial Engineering, University of Arizona
Tucson, AZ, 85721 kylenorland@email.arizona.edu

Faculty Advisor:
Michael W. Marcellin

ABSTRACT

To address the challenge of avoiding dynamic obstacles during the course of the 2018 SUAS competition, a multistage obstacle characterization and avoidance algorithm was designed and implemented. The obstacle characterization section begins with simple base assumptions about behavior and goes through several more advanced stages of obstacle characterization and prediction as more data arises and advanced behavior is detected. The path finding section of the algorithm uses a recursive Monte Carlo path sampling function with a flexible structure that allows for usage with varying computational budgets. It also restricts its computational usage depending on the level of variability in the obstacles.

INTRODUCTION

The dynamic obstacle avoidance algorithm that will be discussed in this paper forms a small but critical part of a wider unmanned aerial system (UAS). This UAS was designed and built by the University of Arizona Autonomous Vehicles Club (AZA) to compete in the 16th Annual Student Unmanned Aerial Systems Competition (SUAS), which is run by the Association for Unmanned Vehicle Systems International (AUVSI) Seafarer chapter. The full contest consists of several elements, including a technical design paper and a flight readiness test. However, the UAS is built to perform in the mission demonstration section of the competition. Mission demonstration consists of obstacle avoidance, waypoint capture, obstacle detection/identification, and air delivery. Obstacle avoidance tests the ability of the UAS to avoid static and dynamic obstacles while flying autonomously. Waypoint capture tests the ability of the UAS to navigate quickly and accurately between points. Obstacle detection/identification measures the ability of the UAS to autonomously identify alphanumeric ground signs and an emergent object. Finally, air delivery demonstrates the ability of the UAS to autonomously drop a water bottle on a given target location.

SYSTEM OVERVIEW

To successfully perform the mission demonstration, AZA designed and built a UAS consisting of a plane, a computer ground station, and a connection to a Judging/Interoperability system, which

interacts with the ground station to judge and facilitate the contest. To complete the obstacle avoidance portion of the competition, the UAS relied on an avionics subsystem consisting of servo controls, sensors, a radio link to the ground station, and a Pixhawk. The Pixhawk is an open source autopilot module capable of managing the on board supervisory control and data acquisition (SCADA) and communications with the ground station. It contains basic ports for sensor and servo connections and runs a simple autopilot software that, when paired with an onboard GPS, can navigate to a given series of coordinate points.

OBSTACLES

In the obstacle avoidance portion of the SUAS competition the plane must avoid a number of digital obstacles that move around during the course of the mission. There are two types of obstacles, static and dynamic.

Static Obstacles: Static obstacles are represented by cylinders with a height, radius, and static location. This information is provided by the judges through the interoperability system. There are up to 10 stationary obstacles with radii between 30ft and 300ft and heights between 30ft and 750 ft.

Dynamic Obstacles: Dynamic obstacles are represented by spheres with location and radius. Their movement is modeled after the behavior of planes and they do not intentionally move to collide with the UAS. Although speed and heading are also properties of the obstacles, only location and radius are provided, continuously, through the interoperability system. There are up to 10 dynamic obstacles with radii between 30ft and 300ft and speeds between 0KIAS and 40KIAS.

ALGORITHM INPUTS AND OUTPUTS

The details of the obstacles form part of the inputs to the obstacle avoidance algorithm. The full I/O is as follows.
Inputs:

- Stationary Obstacle Details
- Dynamic Obstacle Details
- Current Plane Heading and Speed
- Current Pixhawk Path Coordinate Stack

Outputs:

- Alterations in the path (Pixhawk Stack)
- Obstacle observations and characterizations

RESTRICTIONS

During the mission, the plane is constrained by several limitations. At a minimum, to avoid being disqualified from the competition, the plane must remain in its provided airspace, which consists

of a polygonal flight area and a flight height restricted to 700 ft. In addition to competition rules, the plane's movement is limited by both design and choice. Due to its construction, the plane is incapable of turns that exceed its turn radius. The plane is also restricted by design decisions made by the club. AZA decided to hold the altitude and speed of the plane constant during the mission flight due to technical limitations of the computer vision equipment.

EVALUATION

The scoring of the SUAS Mission component is complex, with a full rubric included in the contest instructions [1]. However, most of these scoring areas are of no concern to the obstacle avoidance algorithm. In fact, only the mission time and obstacle avoidance scoring categories are directly affected. These evaluation metrics are intertwined, as, in an effort to avoid obstacles, the time taken by the mission may be affected. As a result, when choosing a path, both flight time and the risk of obstacle collision must be considered. To wisely make this tradeoff, the scoring of each was analyzed. Mission time, worth 8% of the total score, was found to be less valuable than obstacle avoidance, which is worth 20% of the score and drops drastically if any obstacles are hit. From the score calculations, it was found that, if there are ten dynamic obstacles, hitting one obstacle is equivalent to a loss of 2.71% of the mission points, which is equivalent to around 15 minutes of additional flight time. Therefore, it is almost universally better to fully minimize risk of collision to the detriment of flight time unless the time limit of 45 minutes will be exceeded.

OBSTACLE CHARACTERIZATION

STARTING ASSUMPTIONS AND KNOWLEDGE

From the contest instructions, it can be gathered that the purpose of the obstacles is to model a congested airspace in which a UAS would be expected to perform its mission. From experience in previous years, this is taken to mean that the dynamic obstacles can be assumed to be modeled after the behavior of planes and that the obstacles will change course non-instantaneously in a smooth curve. However, although this assumption is expected to hold true, with the rise of multi-copter drones, flight dynamics of other UAS increasingly differ from simple, fixed wing patterns. Additionally, the competing teams are given no knowledge of the individual tendencies and patterns of each of the dynamic obstacles. The combination of this starting knowledge leaves a situation in which the behavior of the obstacles can be weakly assumed to resemble fixed wing aircraft, and, otherwise, is only limited by the speed restrictions given.

GUIDING PRINCIPLES AND THEORY

To best avoid obstacles, an algorithm must characterize and predict the behavior of the obstacles beyond the coordinate information that is given. To do this, most current obstacle avoidance algorithms either rely on significant baseline behavioral assumptions or a vast quantity of training data to teach their software how to best avoid obstacles. However, in the unpredictable, real world situations that a UAS might reasonably be expected to perform, these simplifying techniques may not be accurate or available. In situations like these, an approach that assumes behavior only when it

has been shown, and integrates more complex behavior into increasingly accurate prediction methods is advantageous. In this paper, the obstacle characterization section of the algorithm seeks to do just that. Its methods are modeled after the way that a human or animal would approach obstacle avoidance. It begins with no behavioral assumptions, and, as behavioral patterns are shown, moves between methods to find the optimal balance of computational usage and accuracy.

RISK MAPS

In order to enable more effective path planning and guide the plane safely, each of the obstacle characterization steps must communicate what they have learned to the path planning section of the algorithm. To do this, each behavioral characterization stage outputs a risk map. This risk map spans the flight region and assigns a probability of an obstacle being at a location for each time t. This risk mapping strategy models the way that, when faced with risk, intelligent beings assess the relative risk of pursuing different paths.

PROGRESSION OF OBSTACLE CHARACTERIZATION

The behavioral characterization section of the algorithm moves between several obstacle path prediction methods. Each method represents a possible approach to obstacle path prediction. As the characterizations become more complex, they may become more accurate, however, they also require more computational resources and take more time. To determine whether to move to a higher level characterization method, the algorithm tests for deviations from the low level assumptions. If an obstacle is found to display more complex behavior, the algorithm moves to a higher level method to take advantage. The methods used in the algorithm progress as follows.

1. No Fly Zones
2. Random Probability Mapping
3. Advanced Probability Mapping
4. Probability Mapping with Memory

NO FLY ZONES

When a person or animal first detects a potentially dangerous unknown object, their first reaction is to keep a distance away from it in all directions. Although basic, this avoidance strategy makes sense, as an object with unknown speed and heading could theoretically move quickly in any direction. In the algorithm, this behavior is modeled by a circular risk mapping that surrounds the object in all directions and assumes a high rate of obstacle speed.

HEADING, SPEED, TURN RADIUS CHARACTERIZATIONS

Of course, avoiding whole regions leads to a needlessly inefficient path. As a result, additional characterizations are needed. The simplest additional information that can be determined from the information provided is the current heading and approximate speed of the obstacle. By fitting a

curve to the last several points, the algorithm indicates the heading of the obstacle with an increasing degree of accuracy as more recent points are used. Additionally, by using the coordinates of the obstacles paired with the time stamps at which they were retrieved, the speed of the obstacle can be estimated. Under the assumption that the obstacles will model plane flight, speed is expected to stay relatively constant throughout the mission. Another pertinent obstacle characterization is turn radius. Even if it is assumed that the obstacles are modeled after planes, it is not known how quickly they can turn. To estimate the maximum turn radius for each obstacle, the algorithm samples a number of previous sets of three sequential points. For each set, a certain turn radius is necessary to accomplish the movement. Using the results of these samples, an estimation of the maximum turn radius can be determined.

RANDOM PROBABILITY MAPPING

With these simple estimated behaviors of the dynamic obstacles, an improved prediction can be made. Using heading, speed, and turn radius, an expected movement distribution can be defined. At this early time, there is no evidence to suggest anything other than random turning behavior, so the risk mapping for the expected movement area is evenly distributed.

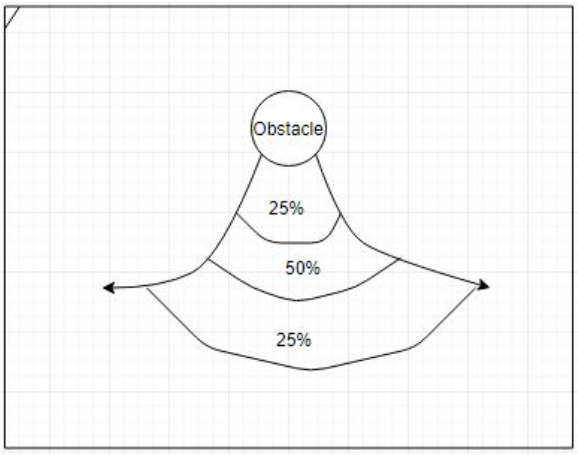

Figure 1: Random Probability Map for One Obstacle

ADVANCED PROBABILITY MAPPING

It is possible that the obstacles are in fact turning randomly. If this is the case, a more advanced characterization would be a waste of computational time. However, most planes with a set destination will not swerve excessively, but, rather, will proceed in a straight line. To take advantage of this and other simple tendencies in turning, a slightly more advanced method is used. To determine when to use it, the algorithm checks whether the behavior of the obstacles is conforming to the expectations of the random probability mapping. If it is detected that the obstacle is not turning randomly, then the algorithm begins to use the advanced probability mapping. The advanced probability mapping stores the past turning behavior of each obstacle and uses it to build a more granular distribution map. By using this mapping, the tendencies of each obstacle come clearly into focus, allowing for more precise path planning (See Figure 2)

CONSIDERATION OF PAST BEHAVIOR

Having developed an expectation of where an obstacle will move from any certain point, the next step that an intelligent observer might take is to determine whether future behavior is strongly correlated to past behavior. That is, if an obstacle takes a certain action, does the probability distribution of the next one differ from the average distribution? To model this as efficiently as possible, the algorithm categorizes recent behavior into slices of approach. Then, for each slice, a rough probability distribution is created based on past data.

Although this approach is potentially more accurate, again, its computational usage is greatly increased, therefore, a strong expectation that behavior is not memoryless is needed. In our implementation, the algorithm attempts this characterization and then compares the created probability distributions. If they are too similar, then the algorithm "regresses" to the simpler memoryless distribution.

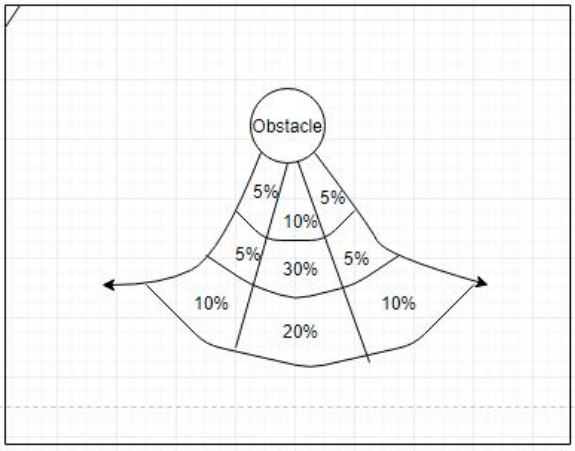

Figure 2: Advanced Probability Distribution

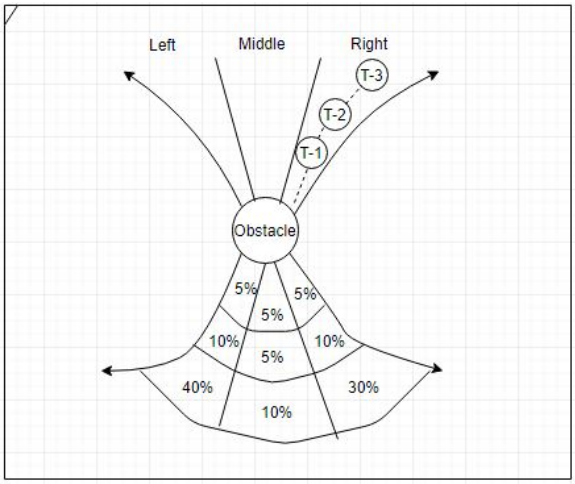

Figure 3: Distribution With Past Behavior Considered

PATHFINDING

Having characterized the behavior of the obstacles and created a risk mapping, the focus turns next to what to do with these characterizations in service of the overall mission. From the risk map provided by the object characterization/prediction stage, the algorithm must identify a safe and efficient path.

PATH CONSTRAINTS

In the process of deciding how to avoid an obstacle, the algorithm faces several constraints. First, any dodging movement is constrained by certain waypoints that must be hit, flight area boundaries, and speed tradeoffs. Determining a path is also limited by computational power/tradeoff time. In a scenario where an obstacle has drawn threateningly close, excessive computation/consideration can be disastrous. Therefore, within the limits of the information given by object characterizations and the various constraints, a flexible algorithm capable of running over a wide set of situations was designed.

LIMITED PREDICTABILITY AND ROBUST SOLUTIONS

Due to the unpredictable nature of the obstacles, the ability to predict the exact state of the obstacles decreases over time, making it difficult to predict whether an initially optimal path will still be safe at a significantly later time. Due to this unpredictability, a robust, slightly inferior solution that guides the plane through a generally safe area may actually be the best choice. If the situation changes slightly, a fully optimal but less robust solution is liable to change into a very poor one. To decide among the many possible paths, a successful pathfinding algorithm must efficiently find a path that is relatively safe and has alternative safe paths nearby in case conditions change.

PATHFINDING ALGORITHM

To accomplish all of these objectives, a Monte Carlo path finding process was developed that would first determine a generally safe region to go through, and then lock into the optimal path for that region. In preparation for the run of this main algorithm, several preprocessing steps occur. The first is the creation of a basic feasible path for the plane. Before takeoff, waypoints and a simple ladder pattern for searching the computer vision area are inputted as a sequential list of coordinates. These coordinates allow for successful completion of the mission even if communications fail. Additionally, before beginning to avoid obstacles the algorithm must determine which obstacles to consider as imminent threats. By ignoring far away obstacles, the algorithm simplifies the calculations necessary to form the risk map. To determine which obstacles to include, the computer tracks the maximum distance that an obstacle has moved in the time that is being considered by the pathing algorithm. If the intended path falls within a circle with this maximum radius, the risk caused by the obstacle will be included.

When the pathfinding algorithm is run, it is given this modified risk map as well as a desired time span to calculate a path for. To begin, the algorithm establishes a set of five turning choices in the center of the desired distance. Although these turns do not cover all situations, in most obstacle

avoidance scenarios, simply turning left or right is sufficient to avoid an obstacle.

Having simplified the choices in this way, the algorithm samples the paths through each of the avoidance choices. To do this, it generates a number of potential paths between the current location of the plane, the avoidance choice, and the end point. The paths are comprised of a number of equidistant connected points with coordinates. To generate sample paths, these pathing points are varied up and down perpendicular to the original path in a psuedo-random fashion that follows a normal distribution spanning the range of possible values based on the turn radius of the plane.

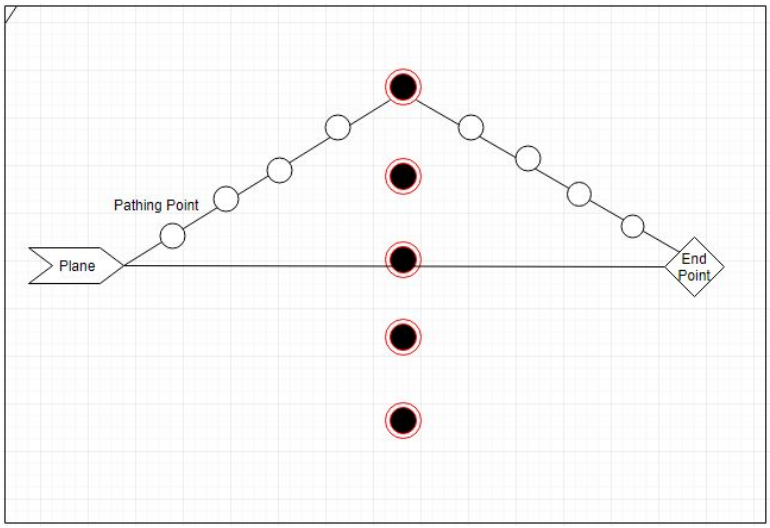

Figure 4: Creating Pathing Points

Each of these created paths are scored based on the risk levels at each pathing point. After the test paths for each avoidance choice have been scored, the avoidance choice with the lowest average risk is selected.

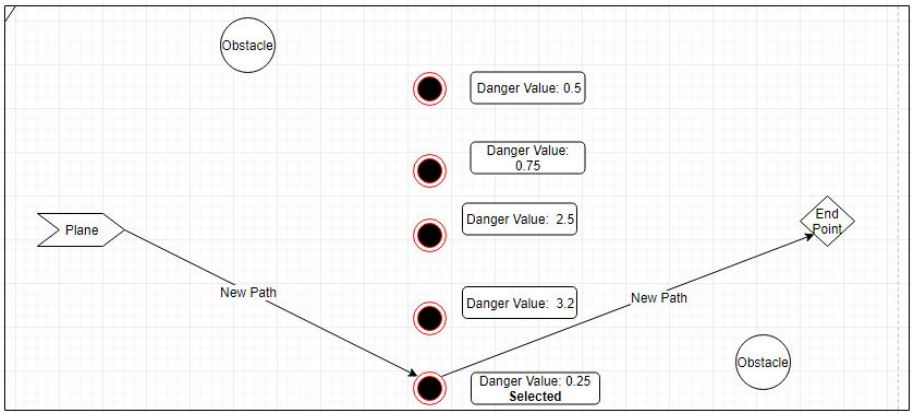

Figure 5: Selecting Avoidance Node.

After this first avoidance choice has been chosen, the optimal path has still only very roughly

been determined. There are still many possible paths. To determine a more exact path choice, the algorithm repeats the same voting process as was done initially with each of the two newly created path sections. Again, five avoidance choices are created, and voted on by sampling the risk levels of the potential paths.

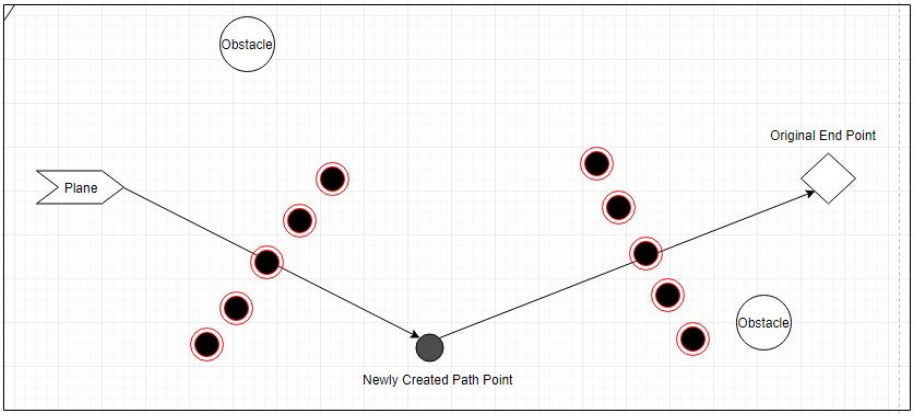

Figure 6: Repeating the Process for Each Bisection

The number of times to do this bisection varies, but when the lowest level bisection has been voted upon, the best path in its test run is chosen as the actual one. With this method of bisection and sampling, the algorithm can address several major challenges in pathfinding with stochastic obstacles. The first challenge is the unpredictability of the obstacles and the decreasing advantage to planning further out. With this basic methodology, it is easy to focus on pathfinding closer to the immediate path of the plane. To do so, the algorithm simply cuts off the recursion process when a desired path resolution has been achieved. The second is the importance of finding a robust solution with mostly safe paths surrounding the chosen one. This is accomplished by the path sampling process, which selects the avoidance choice that has the safest average path through it. Third, the method's general methodology can be easily adapted to situations with both moveable and immoveable points. To incorporate an immovable point, such as a waypoint, the algorithm simply makes that point an endpoint or pathing point and prevents it from moving during the pathfinding process.

IMPLEMENTATION

CODING METHODS, MODULARITY, AND GITHUB

Due to the complexity of the project, a systems approach to the coding process was necessary. To begin, C++ was chosen as coding language. This allowed for an object oriented approach, which matched the structure of the problem. From this object oriented base, the code was then segmented into modules. With this modular approach, an improved characterization or pathfinding approach developed later on could easily be swapped out with the old one. To ensure interoperability in this modular code structure, necessary interfaces between modules were mapped and enforced. Finally, to facilitate collaborative work, the code base was put on Github, where changes could be recorded without extensive paperwork.

SYSTEM INTEGRATION/ PIXHAWK STACK

As stated at the beginning of this paper, the obstacle avoidance algorithm also needed to integrate smoothly into a wider system. The primary integration challenges that the algorithm faced were with the radio communication channel and the Pixhawk. Each had its own challenges. For the radio pathway, because it could be faulty and was often monopolized by other data, the algorithm had to be able to queue communications with the plane and implement data integrity checks. The other implementation challenge was with the Pixhawk. The Pixhawk autopilot uses a stack of coordinates, which it navigates to sequentially. In order to change path midflight, the algorithm had to alter this stack without throwing off the flight path. If this was done incorrectly, a double path could occur, causing the plane to zigzag. To overcome this challenge, the algorithm uses the Pixhawk API to identify and delete all changed path nodes and insert the new path.

CONCLUSIONS

This year's obstacle avoidance algorithm represents a significant advance in complexity and ability relative to the efforts of previous years. With the dual focuses of progressive obstacle characterization and a flexible path planning methodology, the obstacle avoidance algorithm detailed in this paper is suitable to a wide variety of challenging obstacle environments. Additionally, the modular software design allows for expansions to higher level obstacle characterization and future improvements. The algorithm promises to perform successfully at the summer SUAS competition and in similar applications across the growing UAS field.

ACKNOWLEDGEMENTS

The author would like to acknowledge Michael Marcellin for being an available and helpful advisor. Thanks also to Ricky, James, Michael, Wesley, Becky and the rest of the 2018 AZA team for their willingness to talk over the details of the various software and hardware components of the UAS system.

REFERENCES

[1] AUVSI Seafarer Chapter, "Competition Rules — SUAS 2018," Oct. 2017.