Human-Automation Collaborative RRT for UAV Mission Path Planning

by

Americo De Jesus Caves

S.B. in Mathematics,
Massachusetts Institute of Technology (2009)
S.B. in Computer Science and Engineering,
Massachusetts Institute of Technology (2009)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

June 2010

©2010 Massachusetts Institute of Technology
All rights reserved.

Author . . . .
Department of Electrical Engineering and Computer Science
May 21, 2010

Certified by . . . .
Mary L. Cummings
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Professor of Electrical Engineering
Chairman, Department Committee on Graduate Theses

Human-Automation Collaborative RRT for UAV Mission Path Planning

by

Americo De Jesus Caves

Submitted to the
Department of Electrical Engineering and Computer Science

May 21, 2010

In partial fulfillment of the requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Future envisioned Unmanned Aerial Vehicle (UAV) missions will be carried out in dynamic and complex environments. Human-automation collaboration will be required in order to distribute the increased mission workload that will naturally arise from these interactions. One of the areas of interest in these missions is the supervision of multiple UAVs by a single operator, and it is critical to understand how individual operators will be able to supervise a team of vehicles performing semi-autonomous path planning while avoiding no-fly zones and replanning on the fly. Unfortunately, real time planning and replanning can be a computationally burdensome task, particularly in the high density obstacle environments that are envisioned in future urban applications. Recent work has proposed the use of a randomized algorithm known as the Rapidly exploring Random Tree (RRT) algorithm for path planning. While capable of finding feasible solutions quickly, it is unclear how well a human operator will be able to supervise a team of UAVs that are planning based on such a randomized algorithm, particularly due to the unpredictable nature of the generated paths. This thesis presents the results of an experiment that tested a modification of the RRT algorithm for use in human supervisory control of UAV missions. The experiment tested how human operators behaved and performed when given different ways of interacting with an RRT to supervise UAV missions in environments with dynamic obstacle fields of different densities. The experimental results demonstrated that some variants of the RRT increase subjective workload, but did not provide conclusive evidence for whether using an RRT algorithm for path planning is better than manual path planning in terms of overall mission times. Analysis of the data and behavioral observations hint at directions for possible future work.

Thesis Supervisor: Mary L. Cummings
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

First of all, I would like to thank my thesis advisor, Missy Cummings, for giving me the opportunity to be in her lab and guiding me through my thesis. Next, I would like to thank Luca Bertuccelli for all of his help with not only my thesis, but with all my work as a graduate student in the lab. Thanks for your patience, putting up with my poor English and writing, and all your guidance. This whole process would have been much more painful without it.

I would also like to thank the Office of Naval Research for funding this research.

También quiero agradecerle a mi madre, Gloria, mi hermana, Glorianna, mi padre, mis abuelitos, y mis tios por todo el amor y apoyo que me han dado toda mi vida para llegar a donde estoy.

Last but not least, I would like to thank all the participants and pilots in my experiment, John Yazbek for helping me run the experiments, my lab mates, and everyone else that helped me in any way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Unmanned Aerial Vehicles (UAVs) are versatile systems that are being used for an increasing number of applications, and in the future, it is envisioned that many of these UAVs could be operated by a single human operator with the aid of supportive automation algorithms [1]. In order to make this vision a reality, humans and automated algorithms onboard the UAVs will need to collaborate effectively. In recent years, Rapidly exploring Random Tree (RRT) algorithms have been a popular choice for path planning, but if RRTs, or similar randomized planners, are to be used for collaborative planning of UAV missions, their suitability for supporting human supervisors must first be explored. This thesis presents some of the first experimental results in human supervision of multiple simulated UAVs planning with an RRT algorithm.

This first chapter is divided into two parts. The first part discusses the need for increased automation for UAVs navigating uncertain and dynamic environments, and the need for a Human In the Loop (HITL) to supervise these complex missions. Justification is then provided for the selection of the RRT algorithm as the path planning algorithm of choice for UAV navigation. The second part discusses the research hypotheses addressed in this thesis and presents an overview of the methods that are used to answer these questions through a HITL experiment.

1.1 Background

UAVs are aircraft that fly without the need for a human crew on board, and are prevalent in many different applications, such as homeland security and search and rescue. UAVs come in a variety of shapes and sizes, are built for different purposes, and can be equipped to carry out a variety of different tasks. One way to categorize UAVs is by placing them into six functional classes [2]:

- **Target and Decoy -** providing ground and aerial gunnery units with a target that simulates an enemy aircraft or missile.
- **Reconnaissance -** provide battlefield intelligence.
- **Combat -** provide attack capability for high-risk missions.
- **Logistics -** carry out cargo and other logistics operations
- **Research and Development -** further develop UAV technologies.
- **Civil and Commercial -** used for civil and commercial applications, such as fighting fires, and applying fertilizer and pesticides to crops.

These categorizations are not rigid, and the same UAV could fall into several of these classes since that the same UAV could be equipped to perform very different tasks due to the diversity of sensing packages and capabilities.

The roles that the UAV plays drives the need for complex mission planning. For example, in the future vision of firefighting roles, multiple UAVs will perform persistent surveillance missions in search for wildfires, will have to identify the fires, relay the position of those wildfires back to base, all the time ensuring that sufficient fuel is maintained to complete the mission. In addition to all of the navigation constraints typical in airspace management, these UAVs will have to comply with altitude and velocity restrictions, .

Planning UAV missions is a difficult problem due a host of real-world complexities, including the presence of vehicle constraints, poorly known obstacle locations, and dynamic information updates that require replanning. In addition, these sophisticated optimization problems must be solved efficiently for them to be of any use in real-time planning.

With the purpose of reducing problem complexity, hierarchical decompositions of the UAV mission planning problem have been proposed and utilized by numerous authors [3–6]. For example, Russo et al. [6], decompose the UAV mission planning problem into solving instances of several well known fundamental problems in computer science, such as the Traveling Salesman Problem (TSP), the assignment problem, clustering, and minimum-cost path finding [7]. While a human may not necessarily think of planning a UAV mission in this way, an automation-centric decomposition is convenient in explaining the role of the RRT within a human-machine collaborative setting.

The decomposition of the UAV mission planning problem considered for this thesis uses the same components that are used by Russo et al. [6], but groups some of the components of the decomposition into three hierarchical layers. Starting from the lower level, a three-layer decomposition can be described as follows: (1) path planning, (2) resource allocation, and (3) high-level policy generation. The path planning problem is the problem of finding a minimum-cost collision-free traversable path through a sequence of goals, while the resource allocation problem is the problem of clustering, ordering, and assigning tasks to UAVs. Finally the policy generation layer requires reasoning for developing the high level strategy required for carrying out the mission.

Figure 1-1 illustrates a simple abstract UAV mission. In this mission, the gray polyhedra are obstacles, icons 1 and 2 represent two UAVs, and points A through E are targets. In this simple example, the high level policy is simply to minimize the sum of the distances traveled by the two UAVs. A possible solution to the resource allocation problem could be to allocate UAV 1 to visit targets F, A, B, then D, while allocating UAV 2 to visit G, C, then E. The path planning problem is simply the task of finding a minimum length collision free path for each UAV that traverses the targets in the order given as a solution to the resource allocation problem. The decomposition of the problem into hierarchical layers has the potential of resulting in sub-optimal solutions, but the reduction in complexity is frequently a justifiable trade off for real-time mission planning [3, 6].

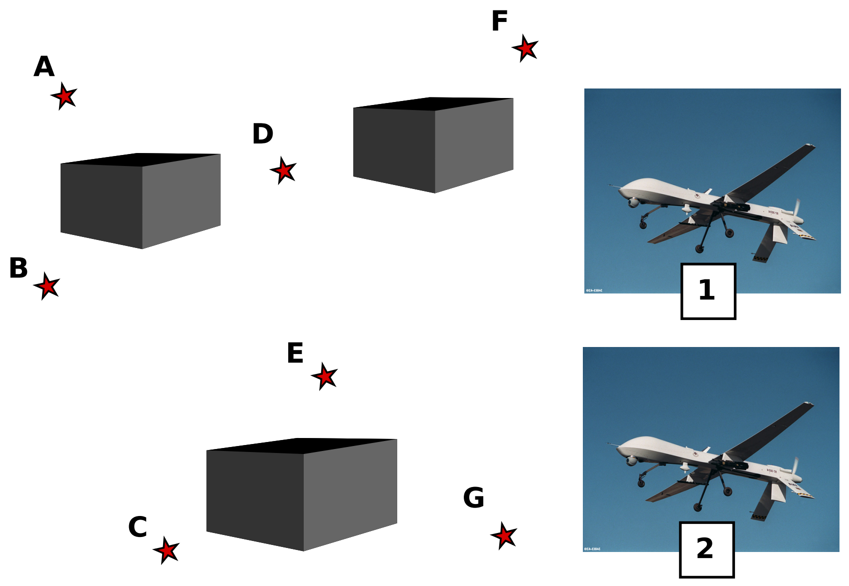


Figure 1-1: Simple UAV mission

1.1.1 Path Planning for UAVs

The path planning problem is still an active area of research, and as previously mentioned, it entails finding collision-free paths for one or multiple UAVs from their current position to their goals in a dynamic environment. Paths that are flyable by UAVs are those that satisfy the kinodynamic constraints of the vehicles.[1] To ensure that the path planning algorithms generate paths that are flyable by the vehicles, it is key that the path planning algorithms incorporate a representative vehicle motion model. For example, a forward-flying UAV in mid-flight cannot instantly reverse its heading due to its kinodynamic constraints; a solution to the path planning problem should therefore not permit a forward-flying UAV to do so. A path that requires a UAV to perform such a maneuver, or any maneuver that is not possible due to the kinodynamic constraints on a UAV is therefore considered *infeasible*. At any point in time, the UAV position is

[1]The kinodynamic constraints of a UAV combine both the constraints from the laws of physics, such as constraints on velocity and acceleration, and the dynamic constraints that take into account the forces acting on the UAV.

limited by the forces exerted on the vehicle. Due to these constraints, a UAV is defined to be a *nonholonomic* vehicle, where a *nonholonomic* system is one in which the state of the system depends on the path taken to reach that state. Examples of nonholonomic constraints are bounds on velocities, accelerations or curvatures [8]. This thesis abstracts UAV kinodynamic constraints into curvature constraints on the feasible paths which will be discussed in greater detail in Chapter 2.

In UAV mission planning there is generally a significant amount of uncertainty in what is known about the environment. For example, targets and obstacles may be moving and/or their exact position may be unknown. In general, finding a solution to the path planning problem with curvature constraints in a large dynamic environment with obstacles is computationally difficult, and doing so under the time pressures of a mission generally increases the computational burden of finding feasible paths in real time [9–11]. An algorithm that finds solutions quickly is therefore desirable, and, for real-time planning considerations, is often required. Real-time planning considerations generally imply that an algorithm must be able to compute a solution fast enough to avoid collisions while generating a plan that is long enough to account for the overall mission performance. Since the rate at which a UAV will need to replan varies depending on the environment, the exact time that the automation has to compute a solution varies as well. Nonetheless, the planner should be flexible enough so that it can deal with highly dynamic environments, requiring that solutions be computed in at most a few seconds.

In order to find a collision free path in such an environment, a representation of the environment is first required. This leads to the consideration of two popular representations: continuous and discrete. In a continuous representation, we simply represent the environment as the Euclidean space $\mathbb{R}^n$ (where $\mathbb{R}$ is the field of real numbers and n indicates the dimensionality of the space), while in the discrete case we must divide the environment into a lattice of points, or equivalently, a collection of cells. A discrete space algorithm cannot be easily adapted to find a path for a nonholonomic vehicle since additional states must be introduced into each discrete point in the environment in order to keep track of the path taken to get to that point in real time.

While there are algorithms, such as Mixed Integer Linear Programming (MILP) solvers, that can solve the path planning problem for nonholonomic vehicles exactly in a continuous space [12], they are not fast enough to handle the complexity of the environments in which UAV missions are carried out. There are also many other algorithms that can be used to solve the path planning problem, but they too have issues, such as long running times and other limitations that arise from their representation of the search space that make them unattractive candidates for UAV mission planning. Some of these algorithms will be discussed in Chapter 2.

An alternative is to consider randomized planners, since they can quickly find feasible solutions in a continuous space [13–15]. The random nature of the solution could prove useful in other ways such as making it difficult for adversaries to know the intended goal of the UAV, adding an element of stealth. While there are several types of randomized path planners, similar to the discrete space planners, many of them do not naturally extend to planning for nonholonomic vehicles [16]. The one exception is the Rapidly exploring Random Tree (RRT) algorithm which will be pursued in this thesis [15–25].

In their simplest form, RRTs search for a path from a starting point to a known goal by randomly sampling the space and growing a tree until the goal becomes part of the tree. Each iteration of the basic RRT algorithm can be broken up into three phases: (1) sampling, (2) finding a candidate node in the tree to extend from, and (3) extending the tree. In the sampling phase, the algorithm randomly chooses a node v to sample from the continuous search space. Next in the second phase, the algorithm selects the point u from the tree which it will extend toward the sample v. In the extend phase, the tree is extended by connecting u to v and adding v to the tree. Four snapshots showing a tree being grown by an RRT algorithm are shown in Figure 1-2.

The algorithm just described is the most basic form of the RRT algorithm and can be modified in order to improve its computational efficiency and solve planning problems for many different systems. For example, the sampling of points (phase 1) could be biased to obtain better results, the selection of a tree point in (phase 2) could be altered to produce smoother solutions, and the extend phase (phase 3) of the RRT can be modified to take the kinodynamic constraints of a vehicle into account.

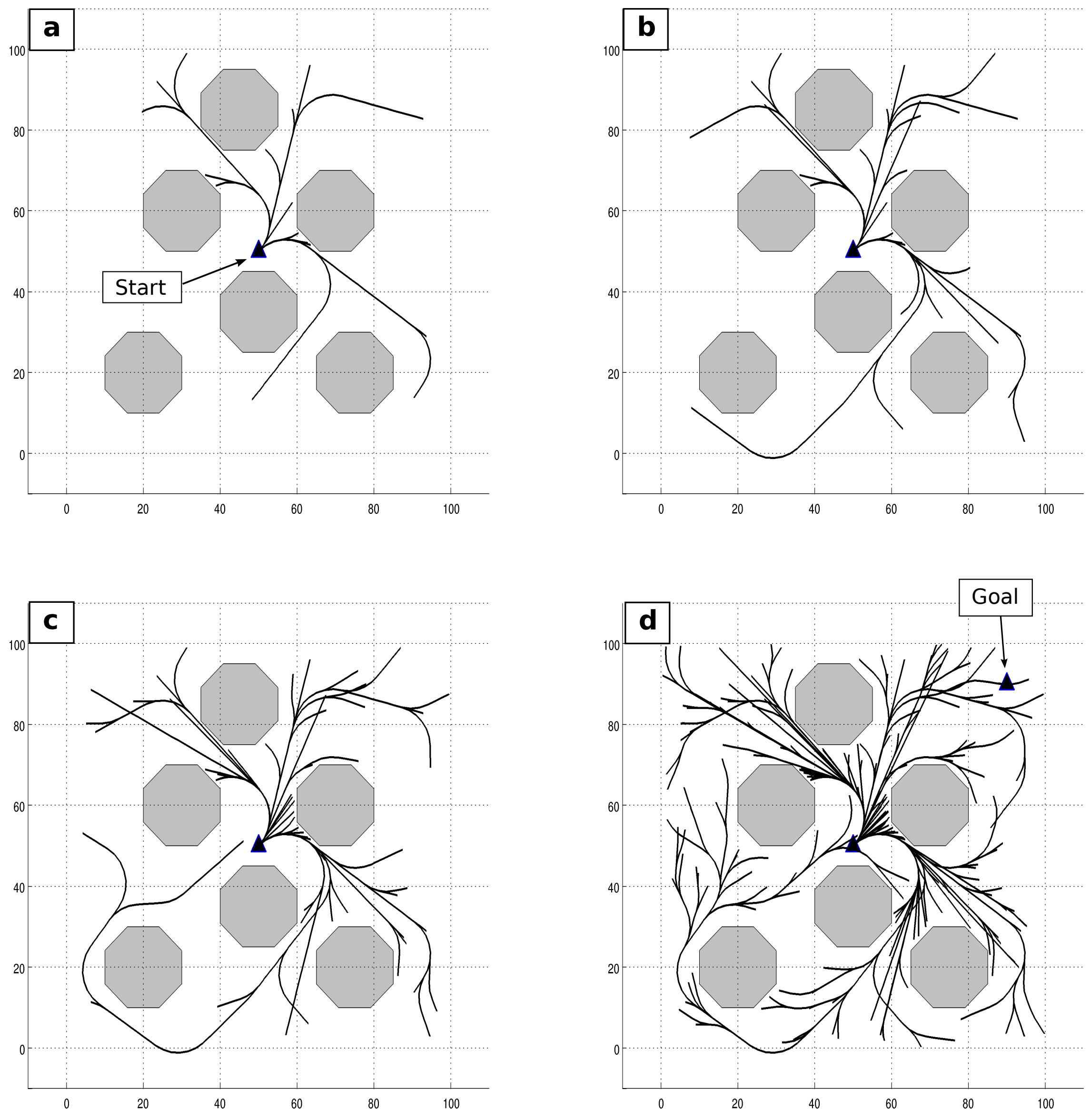


Figure 1-2: Progression of RRT
The tree when the RRT algorithm has gone through (a) 20, (b) 40, (c) 80, and (d) 200 iterations.

1.1.2 Human Supervision

While automation is essential in UAV mission planning for reducing human workload, human operators are nonetheless envisioned to play fundamental roles in the supervision of these sophisticated systems. Since the UAV mission planning problem is very complex, too complex to fully automate, having a HITL to supervise the planning of the mission can prevent a variety of inappropriate automated solutions. For example, a human supervisor could decide how the UAV should approach a target for taking the best possible image, or decide what is the best ordering of the targets based on some mission prioritization scheme that is not well encoded in the algorithms. In addition, there are many high level decisions which should be left to a human, ranging from choosing a high level policy for the mission to firing a weapon. Due to the difference in problem solving abilities between humans and computers, humans are also able to solve perceptually-based problems or problems that computers cannot currently solve at all [26,27]. However, having a single human operator supervise more than one UAV without the aid of automation is an extremely difficult task due to the potentially large workload that arises with operating a UAV [1]. As a result, it is important to find algorithms that allow humans and automation to effectively collaborate in order to reduce the workload on the human operator.

Humans' abilities to think abstractly and qualitatively makes humans much better than computers at solving certain types of path finding problems. In these cases, humans are able to find solutions by simply looking at the representation of the environment. An example of such a path planning problem is shown in Figure 1-3, where the objective is to draw a line from point A to point B without touching any of the black walls. This is a trivial problem for most people, but there are numerous path planning algorithms, including RRTs, that tend to perform poorly in environments such as this one. In fact, RRTs (as well as many other algorithms) exhibit poor performance in maps that have bottlenecks and long narrow corridors [17]. While finding a feasible path from A to B may be trivial for a human, additional real-world complexities may limit the humans' ability to generate feasible plans. What happens, for example, when a vehicle is constrained to fly at a minimum velocity, or to be limited in the turn rate that it can sustain?

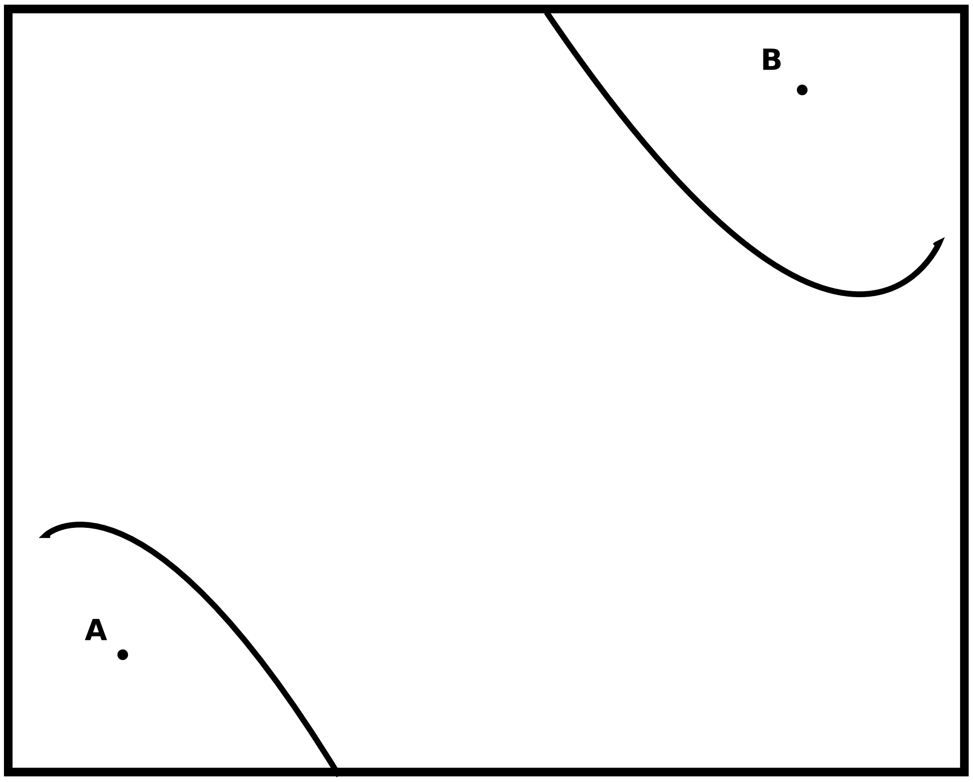


Figure 1-3: Path Finding Map
Connecting Points A and B is more difficult for algorithms than humans.

For a human, these additional vehicle restrictions could transform a trivial problem into a difficult one that could nonetheless be solved by a path planning algorithm. These observations reinforce the ideas introduced in the previous paragraphs that a good UAV mission planner needs to exploit the strengths of both the human operator and the path planning algorithm. Determining the degree to which RRTs are useful for planning with HITL can be an asset to both algorithm development, as well as using the algorithm for real-life missions.

1.2 Research Questions

The overarching goal of this thesis is to validate the use of RRTs for the path planning component of a UAV mission planner. To assess the value of the RRT algorithm for such a planner, a user interface was developed that allows a human operator to collaborate with the automation in order to guide a set of UAVs to their targets through a dynamic

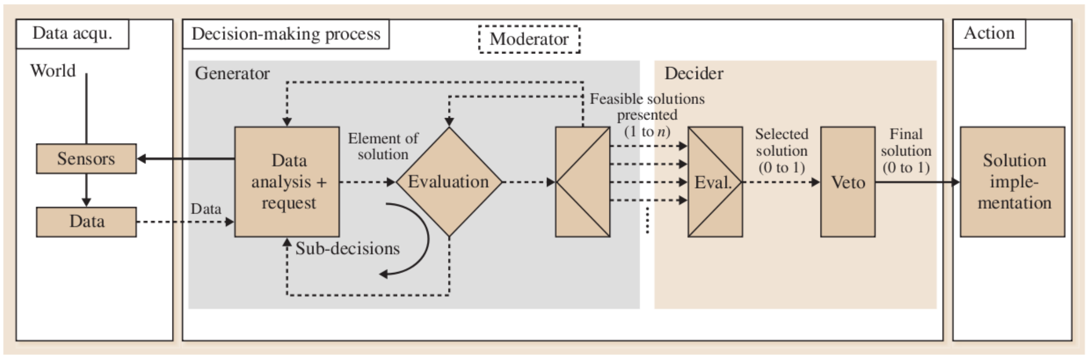


Figure 1-4: Human Automation Collaboration Taxonomy (HACT)
Collaborative Decision-Making Process Roles of moderator (dashed box), generator (gray box), and decider [28]

obstacle field. The Human-Automaton Collaboration Taxonomy (HACT) was introduced by Cummings and Bruni [28] as a model for human operator and automation collaboration, and is a useful representation for explaining the role of the human supervisor in this thesis. Figure 1-4 shows a diagram of the subdivision of the HACT into the three roles also defined by Cummings and Bruni in Ref. [28]: moderator, generator, and decider. The moderator is responsible for keeping the decision making process moving forward, the generator creates feasible solutions from the data, and decider makes the final decision. Note that the human, the automation, or both can take part in any of these roles.

In the planner developed for this thesis, the human takes on the role of moderator. The human prompts the planner for a new path by either adding, moving, or deleting waypoints or asking the RRT algorithm to find a path. The role of generator is fulfilled by the automation as the RRT generates entire paths. The human is the decider and either accepts the solution that is being displayed, or prompts the automation to either modify the path or generate a new one.

It was hypothesized that, on average, the guidance of the human operator would improve both the RRT algorithm's solution quality and runtime. Since RRTs are random, it is difficult to predict the behavior of the algorithm, and there could be cases where human input can produce seemingly strange and unexpected solutions. This observation led to the hypothesis that frustration could arise from a lack of understanding of the RRT solutions, mission time pressures, and UAV kinodynamic constraints. Based on these predictions, an experiment was designed to answer the following research questions:

Research Question # 1: In what ways does having a HITL impact RRT performance?

The RRT is selected as a candidate algorithm for its ability to produce solutions quickly. While it is conjectured that a HITL should improve the performance of the algorithm to account for unforeseen conditions in the environment, it is still useful to determine to what degree the algorithm is capable of dealing with the demands of human input into path planning for a UAV mission. To complement this first research question, the following question is then addressed:

Research Question # 2: How does the RRT algorithm impact the human operator's performance?

Having established that a HITL is essential for supervising UAV missions, it is of paramount importance to understand how the RRT impacts the supervisor's overall performance. If it is determined that RRTs, and other randomized planners, are not suitable for human-automation collaboration, then another approach may be required for increasing the level of automation for path planning.

In addition to any objective HITL performance impacts, it is also important to evaluate subjective assessments of the RRT use since operator acceptance is a critical consideration for successful implementation. This leads to the following research question:

Research Question # 3: What is a human operator's subjective assessment of using a randomized algorithm?

By answering these three research questions, an experiment was designed with the ultimate goal of obtaining a better understanding of the performance of a collaborative and integrated randomized path planning algorithm, and demonstrating the feasibility of using these fast, but typically suboptimal algorithms, with a human in the loop.

The remainder of this thesis is outlined as follows:

- Chapter 2 presents previous relevant work in planning algorithms and human supervisory control to further motivate the research in this thesis,
- Chapter 3 describes the path planning algorithm, the planner, and the experiment that was developed for this thesis,
- Chapter 4 analyzes and discusses the results of the experiment,
- Chapter 5 summarizes the work of this thesis and presents directions for future research.

Chapter 2

Background and Previous Work

This chapter presents a detailed discussion of the previous existing work for path planning problems as it relates to UAVs. The first part of this chapter provides a background on the complexities associated with solving UAV path planning problems, and discusses some common solution approaches. The second part of the chapter provides a more detailed background on the Rapidly exploring Random Tree (RRT) algorithm. Finally, the third part of the chapter introduces relevant background on human supervisory control.

2.1 Background on Planning Algorithms

There are many different formulations of the path planning problem. In one such formulation, known as the mover's problem [9], the objective is to move an agent a, represented by a convex polygon P, from a point s to a point g while avoiding a set of obstacles. By applying a sequence of rotations and translations, a must get from s to g. The *generalized* mover's problem allows the agent a to be represented by any set of connected polygons.[1]

The generalized mover's problem is provably PSPACE-complete [9], where PSPACE is the complexity class of problems that can be solved in an amount of space that is polynomial in the input size. This complexity result means that the path planning problem, even without the presence of any kinodynamic constraints, can be very difficult to solve.

[1]The terminology *agent* is commonplace in the computer science literature to mean an autonomous or semi-autonomous system such as a robot, and will be used interchangeably for UAV in this thesis.

PSPACE-complete problems are at least as hard to solve as the the more well known NP-complete problems. However, for many path planning applications, including the one addressed in this thesis, the generalized mover's problem can be relaxed to problems where the agent a is a point [29,30]. This relaxation reduces the complexity of the problem since the agent's orientation does not have to be considered.

Since this thesis is specific to the control of UAVs, the path planning problem has to consider the generation of paths that relocate the UAV from one location to another while avoiding obstacles, as well as ensuring that the generated paths are flyable by the UAV. A common method for ensuring that the paths are flyable abstracts the vehicle constraints with a minimum turning radius (or maximum average curvature). This approach of abstracting a UAV's constraints to curvature constraints on the path is used in this thesis. Unfortunately, even in considering the 2-D path planning problem for a point agent, it has been shown that finding a curvature constrained path in an environment with obstacles is NP-complete [10].

The complexity of the curvature constrained path planning problem is described in terms of its computational complexity, which expresses the problem's asymptotic runtime in terms of the problem size. It is believed, but not proven, that the runtime for solving any NP-complete problem is exponential in the problem size, and therefore finding the shortest curvature constrained path in an environment with obstacles is expected to take time exponential in the number of obstacles [31]. For example, such an algorithm could find the shortest curvature constrained path quickly for 3 UAVs and 20 obstacles, but take an unacceptable amount of time for 3 UAVs and 25 obstacles. Furthermore, for a path planning algorithm to be useful for the future of UAV mission planning, the algorithm must be able to plan for a sufficiently large time horizon. The time horizon for a UAVs path needs to be large enough so that human supervisors can focus their attention on the supervision of other UAVs or on other tasks. Even in dynamic environments, a sufficiently large time horizon is desired since an operator needs to know which, if not all, UAVs require the operator's attention. Having an algorithm that can quickly plan and replan for large time horizons will prove crucial in reducing a human operator's cognitive workload.

Algorithms that find solutions to the curvature constrained path planning problem are essential, despite the problem's complexity, since this problem is present in countless autonomy tasks. The next sections investigate the various approaches to solving the problem, discussing the multiple methods for solving the curvature constrained path planning problem and assessing possible candidates for UAV mission planning with a HITL.

2.1.1 Discrete Space Motion Planning

One way of approaching the minimum cost path planning problem is to search a discretized space. By discretizing the space, a graph $G = (V, E)$ can be defined, where the set of vertices V represents the set of discrete points in the space, and the set of edges E connects adjacent vertices in V. By defining a cost function $c : E \longrightarrow \mathbb{R}$ on the edges of G, the graph G becomes a weighted graph. On such a weighted graph, the problem of finding the minimum cost (or min-cost) path from a start vertex to a goal vertex can be defined as follows: find the set of vertices $\{v_i\}$ that minimizes $\sum_{i=1}^{n} c(v_i)$ such that $(v_i, v_{i+1}) \in E \; \forall i$.

If we restrict the range of the cost function c to be the non-negative reals, $\mathbb{R}_0^+$, then Dijkstra's algorithm can be invoked for solving the single source min-cost path problem in G [7]. Dijkstra's algorithm greedily selects the vertex with the smallest cost path to the start vertex, and attempts to perform a *relaxation* of its neighbors. *Relaxation* is the process of setting the cost associated with a vertex v to the cost of its predecessor u plus the cost of traveling from u to v if this value is less than the current cost associated with v. Once all nodes have been selected, the cost associated with the goal is the cost of the min-cost path from the start to the goal. Dijkstra's algorithm can be implemented using dynamic programming since a minimum cost path exhibits the optimal substructure property [7, 32]. Dijkstra's algorithm has been used in different ways as a discrete path finding subroutine for planning for UAVs [33, 34].

Another algorithm that exhibits even better performance in practice is A* [35–37]. The standard version of the A* algorithm is very similar to Dijkstra's algorithm except that when greedily selecting the next vertex to relax, it takes a heuristic function into account. Using an admissible heuristic (where an admissible heuristic is one that lower bounds the actual cost) A* is guaranteed to solve the min-cost path problem [37]. While

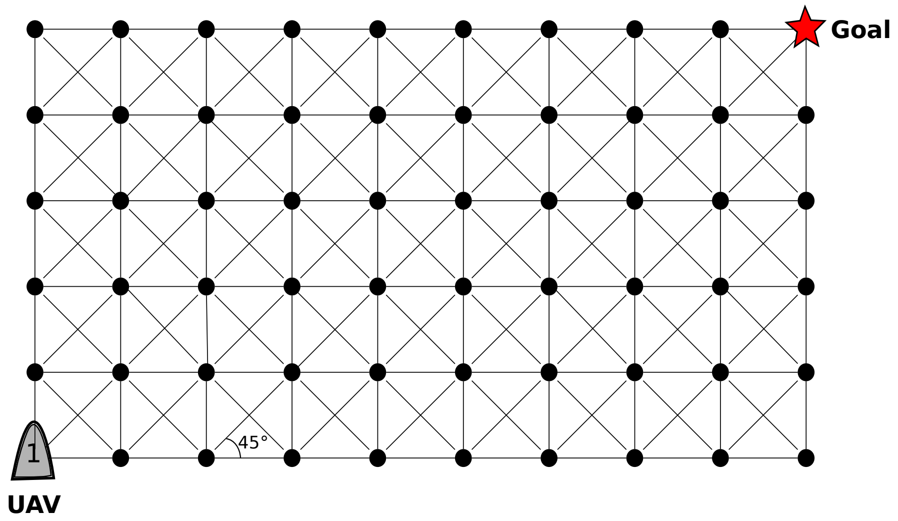


Figure 2-1: 2D search graph for A*
Search graph where the points represent physical locations, while the lines represent graph edges connecting points that are adjacent in the search graph.

A* performs well in practice, there are two principal disadvantages associated with it due to the discretization of the state space.

Finer discretization increases computation time: By discretizing the space, there are several issues that that limit the usefulness of the representation. First, discrete path planning algorithms are efficient in terms of the total number of states in the search space. The degree to which a solution approximates the continuous space min-cost path depends on how finely the space is discretized. Thus, the better the approximation, the higher the computation times. Even if solution quality is sacrificed in order to improve running time, there is still the issue of generating a UAV traversable path, leading to the second limitation.

Discretization may not reflect the continuous path constraints: The paths obtained by the A* algorithm are limited to those that can be represented by a sequence of neighboring points in the search graph. If a traversable path is to adhere to a minimum turning radius, then the representation of the environment needs to allow for A* to find such paths. For example, consider the UAV in the 2D world in Figure 2-1. The UAV is only allowed to be on the points, and can only move between two points if they are connected by a line. Now suppose that the UAV is not allowed to have a change in heading greater than 20 degrees and is initially heading upward. In order for the UAV to move to any of the points that lie to the right, it would have to change its heading by at least 45 degrees. Since this violates the turning constraints on the UAV, A* will not be able find a UAV-traversable path to the goal.

2.1.2 Continuous Space Motion Planning

Given the potential limitation of the discrete space motion algorithms, it is advantageous to analyze continuous space motion planning algorithms. For a given objective function, there are algorithms that can find optimal solutions to the path planning problem with kinodynamic constraints in a continuous state space representation of the environment. A Mixed Integer Linear Program (MILP) provides a framework for formulating and solving the path planning problem exactly for a finite time horizon [12, 38]. A MILP is a mathematical method for determining the optimal value of an objective function given a mathematical model of the system in terms of linear equalities/inequalities and constraints on the variable values. One of the advantages of a MILP approach is that a planner that solves a MILP to find a solution is a *complete planner*, meaning that it returns a solution when one exists, otherwise, the planner indicates that a solution does not exist. Despite this advantage, solving MILPs is computationally difficult since MILPs are NP-hard [39].

One method for handling the computational difficulties associated with MILPs is with the use of receding horizon approaches [40–42], where a solution is optimized over a smaller horizon, and a re-plan is performed. Caution must be exercised since this might result in plan infeasibilities or optimality gaps later on in the mission [41]. Nonetheless, the objective is to find an algorithm for path planning with HITL, and a fairly long horizon is essential. Since the difficulty of solving a MILP increases very quickly by increasing the horizon, this formulation may have practical drawbacks, leading to the investigation of other continuous space planners.

Due to the complexity of the path planning problem, it is likely that any complete planner will have an exponential running time [13]. Thus, we consider planners that are *probabilistically complete*, where a *probabilistically complete* planner is a planner that becomes complete as the running time of the algorithm increases. In other words, if a solution exists, as the running time goes to infinity, the probability that the the path planner returns the solution is 1. While probabilistic completeness says virtually nothing about the rate of convergence, the probabilistically complete algorithms that will be discussed in this chapter search for a solution by randomly sampling the state space, and these algorithms have been shown to have a convergence rate exponential in the number of samples [13–15]. This exponential convergence rate means that the algorithms are expected to find solutions quickly, which is why such algorithms are considered.

This section has thus far demonstrated that solving the path planning problem is provably difficult. The additional specification of meeting the real-time requirements has justified the use of a probabilistically complete algorithm for path finding with curvature constraints. One of these algorithms is the RRT, which can be described as follows: let $T = (V, E)$ be a tree with nodes V and undirected edges $(v_i, v_j) \in E$ connecting pairs of nodes in V. A *path* in T is a sequence of nodes $\{v_1, v_2, ..., v_n\}$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq n - 1$. T is a valid tree if and only if there exists a single unique path between every two nodes in T [7]. In their simplest form, RRTs search for a path from a starting point to a goal by randomly sampling states in the space and growing a tree until the goal state becomes a node in the tree. The unique path from the root to the goal state is a solution to the path planning problem.

The RRT algorithm is not the only randomized planner that is used for planning for curvature constrained systems. The probabilistic road map algorithm constructs a graph in the state space by randomly sampling states, attempting to connect nearby states, and searching for a path from the start state to the goal state [43, 44]. The main problem with utilizing a probabilistic road map algorithm for planning for constrained systems lies in creating the road map and connecting nearby states as it may require connecting of thousands of states, which is impractical for complex curvature constrained systems [16]. Other randomized planners randomly select a state in the tree and apply control functions to obtain a new state to add to the tree [13]. These algorithms seem very similar to RRTs, but when searching the state space, these algorithms have a strong bias toward previously visited regions, while RRTs are biased toward places not yet visited, thereby encouraging exploration [16].

Table 2.1: High level pseudocode for RRT algorithm

```
RRT($q_{start}$,$q_{goal}$)
1  $T \leftarrow$ initialize($q_{start}$);
2  while ($\neg$ T.contains($q_{goal}$)):
3        $q_a \leftarrow$ randomState();
4        $q_b \leftarrow$ bestNode($q_a$,$T$);
5        $T \leftarrow$ extend($q_a$,$q_b$,$T$);
```

Pseudocode for the main method of the most basic form of the RRT algorithm is shown in Table 2.1. Each iteration of the algorithm can be broken down into three phases, which are shown in the algorithm as the methods in lines 3, 4, and 5. In the first phase, the `randomState()` method returns a random sample from the search space. In the next phase, the `bestNode`(q_a,T) method searches the tree for the node in T that is the closest to the random sample q_a according to some metric. In the third and final phase, the `extend`(q_a,q_b,T) method extends the tree T by growing from node q_b toward state q_a.

The RRT algorithm nonetheless may exhibit some pitfalls in that there classes of scenarios in which finding a solution may take an unacceptably long time. In particular, these instances include environments where a solution must go through long narrow corridors or any other bottleneck; the large number of samples required to find a feasible plan may be detrimental to the running time of the algorithm [17]. It turns out, however, that the RRT algorithm as presented in Table 2.1 is very flexible, and each of the methods (lines 3, 4, and 5) can be implemented in many different ways in order to improve the algorithm. Possible modifications to the standard versions of these methods are discussed below.

The first phase, the sampling phase (line 3 in Table 2.1), of the standard RRT algorithm allows for improvements by biasing the sampling of the state space. Using the Euclidean distance as the metric for the nearest neighbor search for a random sample in the RRT algorithm, the nodes of the current RRT define a *Voronoi partition* of the search space into Voronoi regions. For a set of points P in the space, where P in this case is the points in the current RRT, the Voronoi region that corresponds to a point $p_i \in P$ is the region such that any point $q \notin P$ that lies in this region is closer to p_i than any point in

$P \setminus \{p_i\}$. The dynamic domain RRT of Yershova et al. [21] reduces the sampling domain by restricting the Voronoi region of boundary points to the intersection of the Voronoi region of these points and a circle of radius R. A boundary point is a point that is at most a distance ϵ from any obstacle[2]. For a boundary point b and its Voronoi region V_b and a sample s that lies in V_b but is a distance greater than R from b, no attempt will be made to extend the tree from b to s since it is likely that there is an obstacle between s and b. The fact that the extension from b to s is not made without performing the computationally expensive check for a collision is the primary motivation behind the this method of reducing the Voronoi region for boundary points.

If something about the structure of the environment is known or assumed to be known, then this can be taken advantage of in the sampling strategy to obtain samples that will lead to the RRT growing toward a feasible solution. Variants of the RRT algorithm can bias the sampling by excluding points that cannot be a part of a solution due to a priori knowledge of the environment such as intraversable terrain or an upper bound on path length [19, 20]. For example, one could use the RRT algorithm for driving a car in an urban environment and bias the sampling away from states that are unlikely to produce solutions that would be allowable for a car driving on a street, as demonstrated in the DARPA Urban Challenge [18].

The second and third phase of the RRT algorithm (lines 4 and 5 in Table 2.1) consists of choosing a good node in the tree to expand toward the sample and actually expanding the selected node in the tree. It has been empirically shown that choosing the expansion node based on a system's constraints, such as curvature constraints, resulted in a tree that better explores the state space [45]. For this thesis, the kinodynamic constraints are abstracted to curvature constraints on the path of the UAV, and the curvature constrained path generated by the planner in this thesis are Dubins curves [46]. The following subsection discusses the selection and use of Dubins curves for approximating a UAVs path.

[2]The value of ϵ is chosen dynamically during the `extend()` method of the algorithm, and R is set to a multiple of ϵ.

Curvature Constrained Path Modification for RRT

Once the decision is made to abstract the kinodynamic constraints of the UAV, or any agent, to curvature constraints on the path, the next step is to choose a type of curve that represents the curvature-constrained path. There are a number of different curves that have been used for planning for agents that require curvature-constrained paths. These curves include Dubins curves [46], clothoids and anticlothoids [47], Bezier curves [24], Cartesian polynomials [48], cubic spirals [49], and sinusoidal series [49], each with its own drawback. Dubins curves do not have a continuous second derivative; hence constraints on the acceleration of the agent are ignored, while the remaining curves are also limited in that they cannot be computed quickly and accurately. Bezier curves and Cartesian polynomials, for example, do not have a closed form expression for curvature, the quality of a sinusoidal series curve depends on the number of terms, and clothoids, anticlothoids, and cubic spirals do not even have a closed form expression for position [24, 49]. These other curves are already approximations to the kinodynamic constraints on an agent and further inaccuracies only result in further deviation from realistic paths for these agents.

The fact that Dubins curves are well characterized, can be closely followed by real vehicles, and have a closed form expression make them an attractive candidate for generating curvature constrained paths [18, 46]. Let (x, y) be the position of a UAV, θ its current heading, and ϕ the steering direction. Using Dubins paths to approximate a UAVs path, the velocity $\vec{u} = (u_x, u_y)$ is assumed to be constant, and the vehicles kinematic constraints are given by the following equations [17]:

$$\begin{aligned} \dot{x} &= u_x \cos\theta \\ \dot{y} &= u_y \sin\theta \\ \dot{\theta} &= u_\theta \end{aligned} \tag{2.1}$$

Here $u_\theta \in [-\tan\phi_{max}, \tan\phi_{max}]$ and $\phi_{max} \in (0, \pi/2)$.

Dubins curves can be analyzed as follows: Let p_0 be an initial point with velocity $\vec{P}$, and q be a terminal point with velocity $\vec{Q}$. In n-dimensional Euclidean space, $\mathbb{R}^n$, Dubins proved the existence of and characterized the minimum length paths from u to

v with average curvature less than κ [46]. Dubins curves can be broken up into two types of segments: (1) circle segments of radius $\frac{1}{\kappa}$, which will be designated by C, and and (2) straight line segments, which will be designated by S. Dubins determined that these minimum length curvature constrained paths in $\mathbb{R}^2$ are of the form CCC (meaning three sequential circles), CSC (circle, followed by a straight line segment, followed by a circle), or any subsequence of these two. In other words, Dubins characterized the minimum length curvature-constrained paths in 2 dimensions which will be discussed for the remainder of this section.

Dubins characterized the minimum length curvature constrained paths with a given initial velocity $\vec{P}$ and terminal velocity $\vec{Q}$, but for an RRT algorithm, when extending the tree, the vector $\vec{Q}$ is unknown because during the search, it is unknown which $\vec{Q}$ for the sampled state will produce the best path to the goal. Instead $\vec{Q}$ is left unspecified, and the shortest curvature constrained path to the sampled state is computed. These Dubins paths in which $\vec{Q}$ is unspecified are be characterized by two types of paths, each with a closed form expression for the distance [18].

These paths are of the form CS, CC, or subsequences of one of the two, and can be described as follows: Let the current position of the UAV be described by the point $p_0 = (0, 0, 0) \in \mathrm{SE}(2)$, which has an angle of 0, meaning that the UAV is facing along the $+x$ axis. Since the length of the Dubins path for a sample in the half-plane $y \geq 0$ is equivalent to the length of a Dubins path for a sample in the half-plane $y \leq 0$, then for a sample $q = (x, y) \in \mathbb{R}^2$ we will only consider $\tilde{q} = (x, |y|) \in \mathbb{R} \times \mathbb{R}_0^+$. Now let ρ be the minimum turning radius of the UAV, where $\rho = \frac{1}{\kappa}$. If we define $\mathcal{D}_\rho^+ = \{z \in \mathbb{R}^2 : \|z - (0, \rho)\| < \rho\}$ then the minimum length Dubins path from p_0 to $\tilde{q}$, $L_\rho(\tilde{q})$, is given by [18]:

$$L_\rho(\tilde{q}) = \begin{cases} f(\tilde{q}) & \text{for } \tilde{q} \notin \mathcal{D}_\rho^+ \\ g(\tilde{q}) & \text{otherwise.} \end{cases}$$

where

$$f(\tilde{q}) = \sqrt{d_c^2(\tilde{q}) - \rho^2} + \rho \left(\theta_c(\tilde{q}) - \arccos \frac{\rho}{d_c(\tilde{q})} \right)$$
$$g(\tilde{q}) = \rho \left(2\pi - \alpha(\tilde{q}) + \arcsin \frac{x}{d_f(\tilde{q})} + \arcsin \frac{\rho \sin \alpha(\tilde{q})}{d_f(\tilde{q})} \right)$$

In the equations above, $d_c(\tilde{q}) = \sqrt{x^2 + (|y| - \rho)^2}$ is the distance from $\tilde{q}$ to $(0, \rho)$. The angle $\theta_c(\tilde{q}) = \text{atan2}\ (\frac{x}{\rho - |y|})$ is the angle of $\tilde{q}$ from the point $(0, \rho)$, measured counter-clockwise from the negative y-axis; $d_f(\tilde{q}) = \sqrt{x^2 + (|y| + \rho)^2}$ is the distance of $\tilde{q}$ from the point $(0, -p)$, and $\alpha(\tilde{q}) = \arccos \left(\frac{5\rho^2 - d_f(\tilde{q})^2}{4\rho^2} \right)$. Note that the atan2 function in the definition of $\theta_c(\tilde{q})$ is the four quadrant function with the range [0,2π) [18].

To gain an understanding of where $f(\tilde{q})$ and $g(\tilde{q})$ come from, we will look at the paths that these two functions correspond to. With a UAV, we associate a pair of turning circles of radius ρ which can be seen as the dashed circles in Figure 2-2. Next, consider a sample v. If $v \in \mathcal{D}_\rho$ then this means that v lies outside both of the UAV's two turning circles of radius ρ, and corresponds to the function $f(\tilde{q})$. Otherwise v lies inside one of the turning circles of the UAV, and corresponds to the function $g(\tilde{q})$. Figure 2-2 shows the two paths. In (a) we see the path for the case where the sample $v \notin \mathcal{D}_\rho$, while in (b) $v \in \mathcal{D}_\rho$.

The Dubins curves and the corresponding distance function can be easily incorporated into the RRT algorithm. The Dubins distance, $L_\rho(q)$, can be used as the metric for selecting a node in the tree to expand toward the sample. Once a node is selected, the corresponding Dubins curve can be generated from this node in the tree to the sampled state.

This section has discussed optimal curvature constrained paths for the 2D case, but the results that were discussed in this section have been extended to the three dimensional case [50]. Similar to the 2D case, the Dubins curves for the 3D case are also well defined and classified. Since the interface for the experiment run for this thesis displays a 2D representation of the environment, the 3D case will not be discussed any further but left as an area for future work.

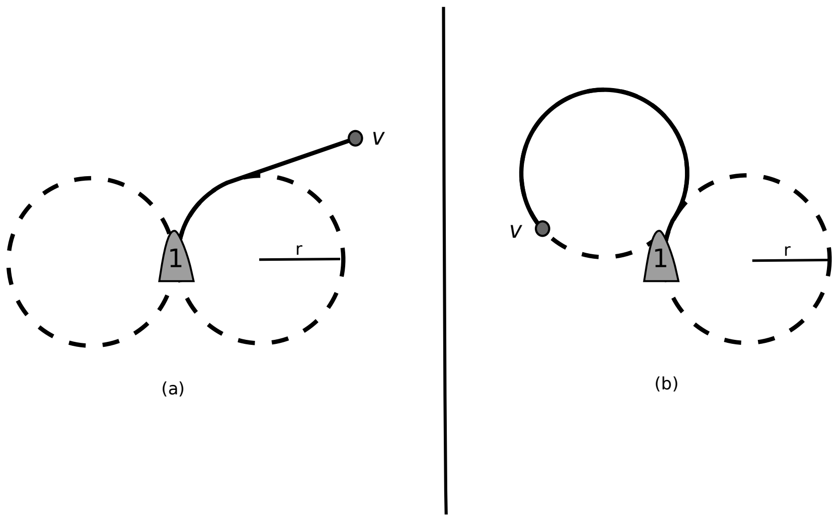


Figure 2-2: Dubins Shortest Paths
Shortest length Dubins path for UAV 1 from its current position and heading to the sampled point v. (a) $v \notin \mathcal{D}_\rho$ (b) $v \in \mathcal{D}_\rho$.

Improving solution quality in the RRT

The use of the Dubins distance $L_\rho(\tilde{q})$ for the selection of the expansion point is a method for smoothing out a solution [45]. In attempt to increase the smoothness and decrease the length of the solution, Kuwata et al. in [18] make use of a heuristic function mapping a point in the tree to its distance to the root. Using the sum of this heuristic and the Dubins distance, the algorithm attempts to minimize the distance from the sample to the start, and not just the distance to the expansion node in the tree. Using the heuristic tends to create trees that are very dense near the root, which can increase the time it takes to find a path to the goal. When searching for an expansion node, the RRT algorithm of Kuwata et al. only uses the heuristic 30% of the time, where the value of 30% was selected from empirical observation.

An approach that attempts to incorporate a notion of a global solution quality in the RRT is one that associates a cost map with the search space. Lee et al. [20] makes use of such a cost map along with a heuristic to attempt to minimize the total cost of the path from the root of the tree, to the sample node, to the goal. The cost function is the cost of the path from the root to the sample, and the heuristic is the distance from the sample to the goal, weighted by a factor h. The use of the heuristic also results in a slow down of the algorithm for the same reason as in the method used in Kuwata et al. [18]: an increased branching factor. Hence, these heuristics must be used with caution for time sensitive applications.

In conclusion, the RRT seems to be a promising candidate for path planning UAV missions due to its ability to quickly generate solutions in complex environments for long time horizons. Due to the flexibility of the RRT algorithm, there are actually methods for the human supervisor to collaborate with the RRT by guiding the search and restricting the search space. More importantly, since the future of UAV mission planning lies in the collaboration of humans and automation, it is crucial that the RRT algorithm be able to easily collaborate with the human supervisor. This next section will overview previous relevant work in human supervisory control and provide motivation for the need for such human planner collaboration.

2.2 Human Supervisory Control

UAV operators in the near future will be expected to be high level managers of the overall mission [1]. Before human operators can take on a supervisory role[3] in UAV mission planning, many components of the UAV mission planning process and human-automation interaction must be studied and understood.

Since workload is a major factor in determining the number of UAVs a single human operator can effectively control, the effect of workload on UAV path planning is an important relation to study. Cummings and Mitchell [52] investigate the relation between workload and performance in UAV task scheduling, and proposed an upper limit formulation for predicting the number of agents a human can effectively control. This number of agents a single human can control is given in terms of the time an agent can be neglected and the time it takes to interact with the agent to raise its performance to an acceptable level.

In the taxonomy of UAV missions proposed by Nehme et al. [53], navigation is a

[3]A supervisory role is defined as the role an operator takes when intermittently interacting with a computer to close an autonomous control loop [51].

common cognitive function across all types of UAV missions. Cummings et al. [1] argue that path planning for UAV missions is part of a lower level loop in the planning system, and in order for a single human to control multiple UAVs, complex automation is needed. This is due to the fact that human operators must divide their cognitive capacity amongst the various tasks the UAVs are performing. If for each UAV, the human operator is forced to dedicate excessive cognitive capacity to navigation, then the operator will be unable to properly deal with high level mission management.

UAV navigation cannot be fully automated with current technology, so to reduce the workload on the human operator, human-automation collaboration can be utilized to combine the strengths of humans and computers to navigate UAVs. A variety of work has been done in studying human-automation collaboration for path planning and in reducing problem complexity. Hwang et al. [54] developed an interface that allows a human to control multiple robot paths by drawing lines on a touchscreen. The automation was only responsible for taking the human operator's input, and converting it into a curvature constrained path by generating Bezier curves with filtered human input. While the system allows a single human to navigate multiple agents by generating curvature constrained paths, a large portion of the navigation workload was still left to the human operator.

In studying human-automation collaboration for systems that include path planning, tools have been developed that were designed to integrate path planning with other aspects of the planning environment. For example, Carrigan developed and tested the Mobile Situational Awareness Tool (MSAT) for assisting users in dealing with health and status monitoring and path planning in a maritime environment [55]. MSAT incorporated MAPP (Maritime Automated Path Planning tool), and human-in-the-loop studies showed that humans could effectively and quickly use this A*-based path planner, as well as develop appropriate levels of trust for such an automated tool [56].

The degree to which the level of automation in path planning affects other factors of high level mission planning has been studied as well. Marquez [57] studied the effect of different levels of automation and visualization in path planning in the domain of human planetary surface exploration. The results indicate that an increased level of automation can leads to a decreased level of situational awareness, and vice versa. Both trust and loss of situational awareness should be considered carefully in work that integrates both high level and low level tasks for UAV mission planning.

One example of using the strengths of humans and computers to reduce problem complexity is to use human input to make an typically intractable problem tractable. For the plan recognition problem,[4] Lesh et. al. [59] implemented an algorithm such that when an ambiguity arises, the automation was designed to ask the human user for clarification in an attempt to guide the search and reduce the size of the state space. For the domains in which the algorithm was tested, the collaborative algorithm was successful at making the plan recognition problem tractable. The same strategy of using human input to guide the search and reduce the state space is adopted by the work in this thesis.

In order for human operators to become high level supervisors of UAV missions, additional layers of automation are needed. Navigation is a common, high workload, cognitive task in UAV mission planning, but due to mission environment complexities, current technology cannot fully automate UAV path planning. The RRT path planning algorithm is an attractive candidate for its ability to quickly generate solutions and its adaptability for human-automation collaboration. As such, the human-automation collaborative RRT (c-RRT) developed for the work in this thesis will be described, analyzed, and discussed in the the remaining chapters.

[4]The problem of plan recognition is to take in as input a sequence of actions performed by an actor, and to infer the goal pursued by the actor and also organize the action sequence in terms of plan structure [58]. In general, this problem is intractable.

Chapter 3

Experimental Design

This chapter discusses the experimental design used for the assessment of a Rapidly exploring Random Tree (RRT) algorithm for collaborative UAV mission planning. The first section in this chapter reviews the human-automation collaborative RRT algorithm (c-RRT) that was implemented for this experiment, the second section describes the planner, experiment, and measurements that were made during the experiment, while the third section addresses how the choice of variables addresses the research questions of this thesis.

3.1 Collaborative Human-Automation RRT (c-RRT)

The c-RRT algorithm differs from the standard RRT algorithm originally introduced by LaValle [17] in three main ways: (1) the algorithm generates curvature-constrained paths, (2) the human operator can constrain the solution space by providing as input a sequence of points to the algorithm, and (3) the human operator can modify paths by moving points of interaction, referred to as waypoints, that are along the UAV paths.

In order to permit the human operator to plan/replan the UAV paths, the interface must allow the operator to obtain UAV flyable paths, where a flyable path is one that satisfies the curvature constraints on a path. To do so, the mission planner was designed to allow for two types of paths: waypoint-defined and c-RRT generated paths. The c-RRT generated paths are simply the paths generated from growing the tree in the c-RRT

Table 3.1: c-RRT Algorithm Pseudocode for Growing Tree

```
RRT($q_{start}$,$q_{goal}$)
1  $T \leftarrow$ initialize($q_{start}$);
2  while ($\neg$ T.contains($q_{goal}$)):
3      $q_a \leftarrow$ randomState();
4      $q_b \leftarrow$ bestNode($q_a$,$T$);
5      $T \leftarrow$ extend($q_a$,$q_b$,$T$);
```

algorithm while waypoint-defined paths are computed from Dubins curves. Before the c-RRT algorithm is described in more detail, the state space in which the algorithm performs its search is described first.

3.1.1 State Space

For this thesis, the state space C that the algorithm searches is a mathematical representation of a 2D environment. For an unbounded, obstacle-free environment, $C = \mathbb{R}^2$. Obstacles can be represented as arbitrary closed connected subsets of $\mathbb{R}^2$. For the set of obstacles O, the unbounded 2D search space is then the set difference $C = \mathbb{R}^2 \setminus O$. For practical purposes for the experiment that will be described in more detail in a later section, restrictions were applied to the state space just described. These restrictions include constraints on the obstacle-free state space due to representational limitations of the Graphical User Interface (GUI), as well as constraints on the allowable obstacle sets due to limited computational resources.

The entire UAV mission environment is contained on a single screen, thus it is desirable for the obstacle-free state space to be limited to a bounded subset of $\mathbb{R}^2$. In order to achieve computational efficiency both while sampling and extending the RRT, it is necessary to quickly determine whether a point $p \in \mathbb{R}^2$ lies in an obstacle. Arbitrary obstacle shapes are admissible, but without loss of generality, all of the obstacles that were used for the experiment in this thesis were selected to be regular octagons.

3.1.2 c-RRT Generated Paths

To grow a tree, the c-RRT algorithm takes a start state and a goal state as input, initializes a tree with the start state as its only node, then grows a tree until the the tree contains the goal state. The non-trivial part of the algorithm, the growing of the tree, occurs during the three methods shown in lines 3, 4, and 5 of Table 3.1 and it is sufficient to describe the `randomState()`, `bestNode(`q_a`,` T`)`, and `extend(`q_a`,` q_b`,` T`)` methods in order to obtain an understanding of what the algorithm does.

For simplicity and generality, the sampling strategy used by the `randomState()` method in the c-RRT algorithm is the straightforward, unbiased sampling strategy used by the basic RRT algorithm presented in LaValle [17]. With probability r, the `randomState()` method simply returns a point from the space $\mathbb{R}^2$ at random, and with probability $(1-r)$, `randomState()` returns the goal state. The value of r needs to be selected so that a sufficient portion of the state space is explored by the tree. This in turn allows a collision-free extension of the tree to the goal to be made when the goal is selected as the sample state by the `randomState()` method. With this in mind, the probability r was chosen empirically by setting the value of r close enough to 1 such that the algorithm found paths to the goal most of the time, but not so close that the interface would freeze or lag often due to high memory consumption caused by a large tree. After tuning this parameter through extensive numerical experiments, the value of r was set to r=0.95.

The second method, the `bestNode(`q_a`,` T`)` method, takes as input the sample state that was returned by the `randomState()` method and finds the best node in the current tree for extending to the sampled state q_a. There are different notions of what it means to be the best node, and in the standard form of the RRT algorithm [17], the best node is defined as the nearest neighbor (in terms of Euclidean distance) in the tree T to q_a. While there are efficient data structures, such as KD-Trees [7], that can be exploited for finding the nearest neighbor with the Euclidean distance metric, using a metric based on system constraints, such as curvature constraints, results in an RRT search that better explores the state space [45]. In addition, using system constraints as the metric for selecting the nearest neighbor tends to produce smoother solutions [18].

This thesis abstracts UAV kinodynamic constraints as curvature constraints on the paths, which are represented as Dubins paths [46]. Thus, the metric used for finding the nearest neighbor is the Dubins distance $L_\rho(\tilde{q})$ described in Section 2.1.2. In order to make use of the function $L_\rho(\tilde{q})$ in the algorithm, two transformations need to be applied to both the sample state q_a and each tree node $p_i \in T$. Two assumptions are made in the expressions that are in the definition of the function $L_\rho(\tilde{q})$: (1) $p_i = (0, 0, 0) \in SE(2)$, where $SE(2)$ is the special Euclidean space of rotations and translations in 2 dimensions, and (2) $\tilde{q} \in \mathbb{R} \times \mathbb{R}_0^+$. Since a node in the tree T will usually not be at the origin $(0, 0, 0)$, then for $p_i = (x_i, y_i, \phi_i)$, both p_i and q_a are and translated by $(-x_i, -y_i)$ to the origin then rotated by $-\phi_i$. In addition, the translated and rotated q_a, $q'_a = (x'_a, y'_a)$, is transformed into $\tilde{q'_a}$ by setting $\tilde{q'_a} = (x'_a, \|y'_a\|)$. After applying these transformations, the function $L_\rho(\tilde{q'_a})$ can be computed and used as the metric in a linear search for the nearest neighbor.

In an attempt to obtain smoother and possibly shorter paths, with probability α, a heuristic value, $h(q_b)$, is added to the Dubins distance. For a tree node q_b, the heuristic function $h(q_b)$ maps the tree node q_b to the distance from q_b to the root of the tree, where this distance is computed by following q_b's ancestors in the tree back to the root. By using the sum $h(q_b) + L_\rho(\tilde{q'_a})$ as the metric for determining the nearest neighbor, the best node is the node in the current tree that will minimize the distance from the root to the sampled state. Since the use of this heuristic results in trees with a smaller depth (thus impeding the exploration of the space), the value of α should not be set too high, and is therefore set at $\alpha = 0.3$. This is the same value that was empirically selected by Kuwata et al. [18].

The last of the three methods, the `extend(`q_a`, `q_b`, `T`)` method, extends the tree by generating a Dubins path from $q_b = (x_b, y_b, \phi_b) \in SE(2)$ to $q_a = (x_a, y_a)$. Similar to the transformations that are done for the `bestNode(`q_a`, `T`)` method, both q_b and q_a are translated by $(-x_b, -y_b)$, then rotated by $-\phi_b$. The Dubins path that corresponds to the transformed points q_b and q_a is then computed. If the tree were continuous, then it would be possible for the best node to lie somewhere on a Dubins path between some q_b and q_a. Due to computational limitations, the tree is not continuous, but instead defined by

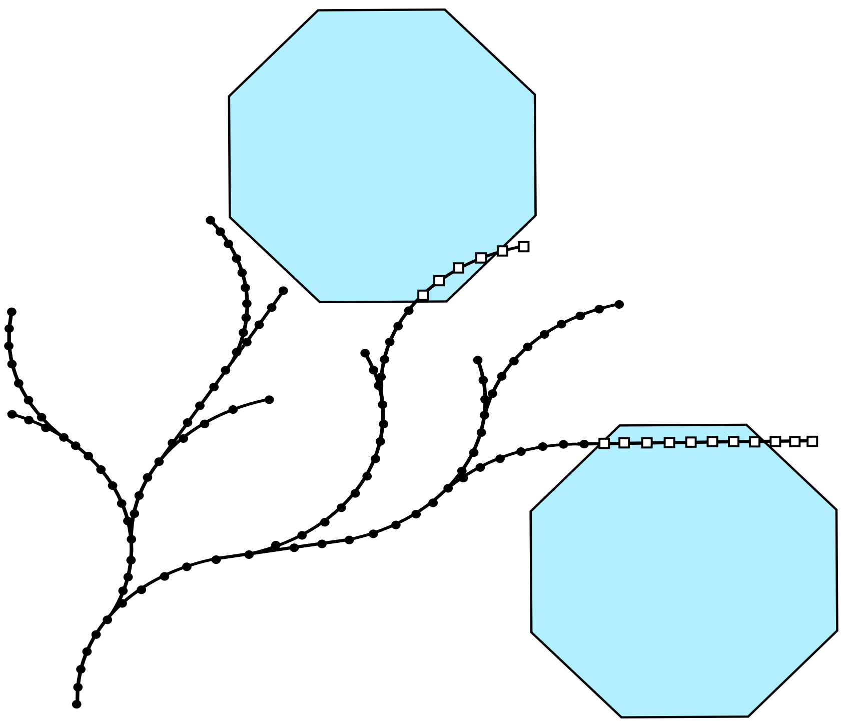


Figure 3-1: Growing the Tree
The tree is defined by nodes (black circles) sampled from Dubins paths. Samples from a Dubins path that has collided are not included (white squares).

a set of nodes. Since q_b and q_a may be far apart, many potential best nodes may be lost; thus it is desirable to include some of the points along the Dubins path between every q_b-q_a pair. The c-RRT algorithm samples points equidistantly along the Dubins paths and adds them to the tree. Once added to the tree, points become nodes in the tree. Tree nodes belong to the space $SE(2)$ since in addition to a node's location in Euclidean space, each node is assigned a heading which is determined by the spatial relation between the node and its parent. To prevent the tree from growing into an obstacle, the sample points are added sequentially along the Dubins path. When a sample point is determined to be inside an obstacle, the extension of the tree toward the sample state q_a halts, and the method terminates. This is shown for a few iterations of the algorithm in Figure 3-1.

3.1.3 Guiding the Search with Subgoals

From the algorithm's perspective, having human-automation collaboration has two main purposes: using the human operator's input for (1) reducing the amount of computation required by the algorithm and (2) guiding the algorithm to find a solution that is in the neighborhood of what the human wants or expects.

It is well known that RRTs tend to perform poorly in scenarios where there are bottlenecks and/or long narrow corridors [17], and one way that these problems can be circumvented is by growing multiple trees then connecting them to form a single solution. One version of the RRT that uses more than one tree is the bi-directional RRT with kinodynamic constraints [15]. The bi-directional RRT simultaneously grows a tree from both the start and the goal states by alternating which tree is expanded at each iteration. The algorithm looks for a path by attempting to grow the two trees until they merge. This approach comes with the drawback that in general, a discontinuity of the curvature constraint in the path will exist at the point where the two trees meet [15]. There are methods for trying to correct the discontinuity in the curvature, but this thesis avoids that problem altogether by using a different approach from the bi-directional RRT.

Instead of growing two trees from opposite ends, the c-RRT algorithm grows a sequence of trees rooted at a given sequence of points referred to as subgoals. The c-RRT algorithm allows a sequence of subgoals to be set by the human operator by clicking on the map interface with the mouse. The heading of the root of each tree is set to be the heading of the goal in the previous tree. Within each tree, the curvature constraints are guaranteed to hold, and by growing the trees sequentially and giving each tree root an initial heading, the curvature constraints are guaranteed to hold from start to final goal. This method of growing multiple trees from a sequence of subgoals provides the human operator with a method for mitigating the complete randomness of the generated path, since the solution must go through all the specified subgoals. In the mission planner, the goals, or targets, are red squares, while subgoals are solid black squares. An example of a target and a subgoal is shown in the environment in Figure 3-2. The difference between subgoals and waypoints, shown in Figure 3-2, is that subgoals are points that act as waypoints for the algorithm, while waypoints are points on the UAV paths that are displayed for human interaction.

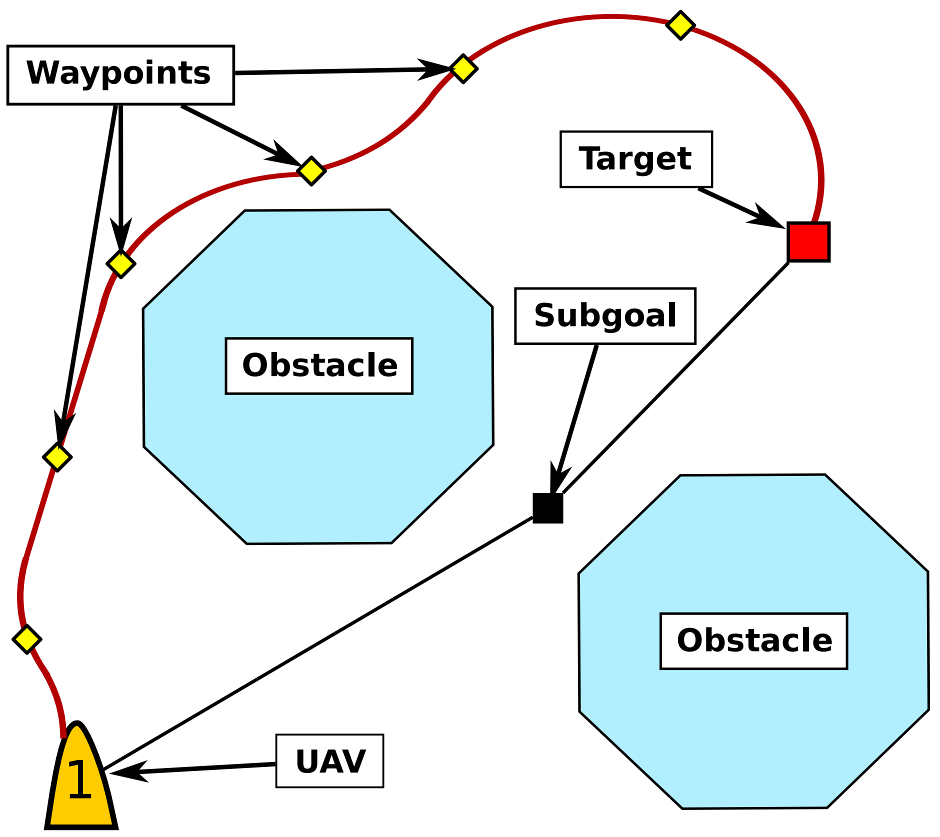


Figure 3-2: UAV in Simple Environment

3.1.4 Waypoint-Defined Paths

A UAV's waypoint-defined curvature constrained path is defined by the UAV's current heading and the following sequence of points: the UAV's current position, a sequence of waypoints, and a target. For every pair of consecutive points in this sequence of points, the path from the first point in the pair to its successor in the path is generated in the same way that the `extend(`q_a`,` q_b`,` T`)` method generates a Dubins path from points q_b to q_a. Note that in order to generate such a Dubins path, the heading of the first point must be known, while the heading of the second point should not be initially specified, but instead set after the path is computed. When computing the path, the only point that has a heading already assigned to it is the point at the UAV's current location. Thus, the path from the UAV to the target is computed by first generating the path from the UAV to the first waypoint, then from the first waypoint to the second waypoint, and eventually, the last waypoint to the target.

The Dubins paths generated between pairs of points are the shortest possible curvature constrained paths. However, it is important to note that the overall path generated from a UAV, through a sequence of waypoints, to the UAV's next target is not guaranteed to be the shortest possible. The sub-optimality of this policy for constructing paths can be shown using the fact that the heading of any point in the path only depends on its predecessors [46]. Determining which heading for each of the waypoints produces the shortest Dubins path is a combinatorial optimization problem. Therefore, trying to determine the optimal curvature constrained path through the waypoints can quickly become computationally intensive and is not suitable for the real-time computations carried out in experiment for this thesis. Instead, the heading of a waypoint is determined completely by the location of the waypoint and its predecessor's location and heading. A waypoint's heading is computed by finding the tangent to the Dubins path at the point where the Dubins path reaches the waypoint.

3.2 The Experiment

The mission planner was designed to support a simulation of a simple UAV mission while allowing different levels of interaction between the human operator and the c- RRT algorithm. For the purpose of the experiment for this thesis, the mission planner was designed to operate in four modes of operation: Human-Only (**HO**), Human-Guided 1 (**HG1**), Human-Guided 2 (**HG2**), and Human-Constrained (**HC**). These four modes of operation differ in the level of automation presented to the user. The **HO** mode requires human operator to manually guide the UAVs by manipulating waypoints, while the **HG1** mode gives the human operator the choice of using the c-RRT algorithm without the ability to set subgoals for generating paths. The **HG2** mode forces the operator to ask the c-RRT algorithm, without the use of subgoals, to find a collision-free path, and the **HC** mode gives the human operator full access to the c-RRT algorithm.

The c-RRT algorithm allows human operators to interact with the RRT algorithm both before and after the algorithm searches for a solution. Interactions prior to the search are in the form of specifying subgoals, while interactions after the search are in the form of waypoint modification. In the **HG1** and **HG2** modes, operators interact with the c-RRT algorithm only after the search, while in the **HC** mode, operators interact with the algorithm both before and after the search.

3.2.1 Mission Planner Modes of Operation and Obstacle Densities

These modes were designed to give the human operators in the experiment three levels of interaction with the c-RRT algorithm: (1) no interaction in the **HO** mode, (2) the ability to use the algorithm to obtain solutions and modify them after they are presented in the **HG1** and the **HG2** modes, and (3) the ability to use the algorithm by both constraining the solution space with subgoals and modifying the solutions after they are presented in the **HC** mode. The **HG2** and the **HC** mode were designed such that participants are forced to use the c-RRT algorithm without and with specified subgoals, respectfully. This was done for purposes of the data analysis. Forcing participants to use the algorithm and specify subgoals enables measurement of the algorithm's performance to be made during the mission for every participant. Both the **HG1** and the **HG2** modes were included in the mission planner design in order to compare the performance and behavior of human operators when given the choice of whether or not to use the c-RRT for planning.

In all four of the modes of operation, for each UAV-target pair, the path is initially set to the Dubins curve generated by the UAV's location and heading and the target's location. In other words, ignoring obstacles, the initial path is the shortest curvature-constrained path to the next target. To avoid collisions with obstacles, the human operator has different abilities, which vary between the modes of operation, to guide the three UAVs.

The simplest of the four modes in which the missions are carried out is the **HO** mode of operation. In this mode, the human operator has to guide the UAVs to their next target while avoiding the obstacle field by manually generating waypoint-defined paths, which are generated manually by adding, deleting, or moving waypoints.

In the **HG1** mode, human operators are still able to manipulate the UAV paths manually, except that they no longer have the ability to add waypoints. Instead, waypoints are given on the UAV paths so that users have the ability to modify the current paths by moving or deleting the waypoints. In addition, operators have the option of having the c-RRT algorithm generate a collision-free path to the next target. Every time the c-RRT algorithm is called, it attempts to look for a collision-free path for the selected UAV from its current location to its next target. If a path is not found, then the UAV's current path is not modified. In this mode of operation, waypoints are always present on all of the UAV paths, giving operators the option of using the algorithm or manually guiding the UAVs.

The **HG2** mode is similar to the **HG1** mode, but was designed to force operators to use the algorithm. Unlike the **HG1** mode, waypoints are not always present on the UAV paths. Waypoints only appear on the selected UAV's path after the c-RRT algorithm searches for a solution, and they disappear once the selected UAV is no longer selected. Since operators have no way of interacting with the UAV paths to make modifications, they are essentially forced to use the algorithm to guide the UAVs around obstacles.

The most complex of the four modes, the **HC**, is similar to the **HG2** mode, except that it requires the human operator to specify at least one subgoal in order for the c-RRT algorithm to run. The **HC** mode forces operators to guide the c-RRT algorithm's search by specifying a sequence of subgoals which the algorithm must include, in order, on the path it generates to the goal. It is the only mode that allows this additional interaction of specifying subgoals to the c-RRT. Once the subgoals are set, the mechanics of this mode of operation are identical to the **HG2** mode.

Obstacles are present in the the UAV mission environment, and while the obstacles do not move, they do appear and disappear. For the purposes of the experiment for this thesis, two different levels of obstacle density were used to represent environment complexity: Low Density (**LD**) and High Density (**HD**). The obstacle density refers to the constant number of obstacles present on the map at any time. There were two different obstacle density environments that were incorporated to the experiment: **LD** and **HD**. The obstacle density refers to the constant number of obstacles that are present at any given time during the mission. The **HD** obstacle fields maintained 20 obstacles, while the

LD obstacle fields maintained 12 obstacles.

For the experiment, a total of eight maps were generated; four for each of the two obstacle densities. For each map, a fixed grid of obstacles was generated, and the obstacles were randomly scheduled to alternatingly appear and disappear at five second intervals to maintain a constant number of obstacles on the map.

3.2.2 Simulating UAV Missions

For each UAV mission and each UAV, the same starting location and sequence of four targets was assigned to each UAV. Once the mission began, the UAVs started moving at a constant speed along their paths toward their next target. Once a UAV reached its target, the UAV's next target in the sequence appeared and the UAV continued to move toward it at the same constant speed. To simulate time, the computer was asked to execute a "timer task" periodically a couple of times per second. Every time the the timer task was executed, two crucial events took place: (1) UAVs updated their position by moving along their corresponding path, and (2) a check was performed to see whether any obstacles needed to be added or removed. Unknown to the user, obstacles were added or removed based on a predefined schedule assigned to each obstacle.

3.2.3 Operating the Interface

In the mission planner interface shown in Figure 3-3, we see three UAVs and three targets, with one target per UAV. The UAVs are connected to their target by a path. On each on these paths there can be a sequence of waypoints, which are the yellow diamond shapes on the paths. Waypoints are the points of interaction for the human operators and can be added only in the **HO** mode, and deleted and moved in all modes. A waypoint can be added to the selected UAV's path by left-clicking on the map at the desired location, while a waypoint can be deleted by right-clicking the desired waypoint on the selected UAV's path. A waypoint can be moved by pressing the left mouse button over the desired waypoint on the selected UAV's path, dragging the mouse, then releasing the mouse button when the waypoint is at the desired location. While a waypoint is being

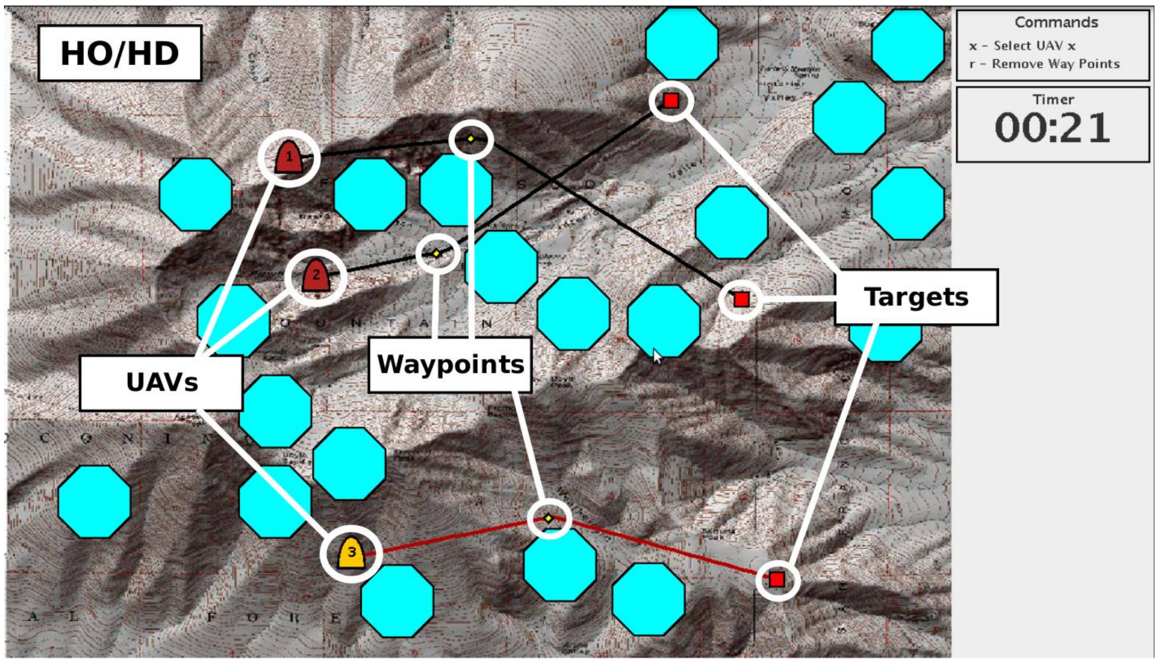


Figure 3-3: Human-Only (HO) Mode with High Density (HD) Obstacles

dragged, the UAV's path is constantly recomputed and displayed to the operator.

An unselected UAV is dark red and its path is black, and when a UAV is selected, it becomes yellow-orange while its path becomes red. Only one UAV can be selected at a time. The selected/unselected UAVs are labeled in Figure 3-4. In the mission planner, a UAV can be selected by pressing the number on the keyboard that corresponds to the desired UAV's ID. For example, UAV 2 in Figure 3-4 was selected by pressing the number "2" on the keyboard.

In the **HG1**, **HG2**, and **HC** modes of operation, the human operator has the ability to use the c-RRT algorithm to look for a collision-free path for the selected UAV. In the **HG1** and **HG2** modes, the human operator can ask the c-RRT algorithm to compute a path by pressing the "g" key. If the algorithm finds a collision-free path for the selected UAV to its next target, the computed path is set as the UAV's path. This feature was not included as a selectable button in the interface since the users were trained to choose the desired UAV by selecting the numeric keys on the keyboard to avoid confusion between selecting a UAV or adding/selecting waypoint when clicking on the map.

In the **HC** mode of operation, before asking the c-RRT algorithm to search for a

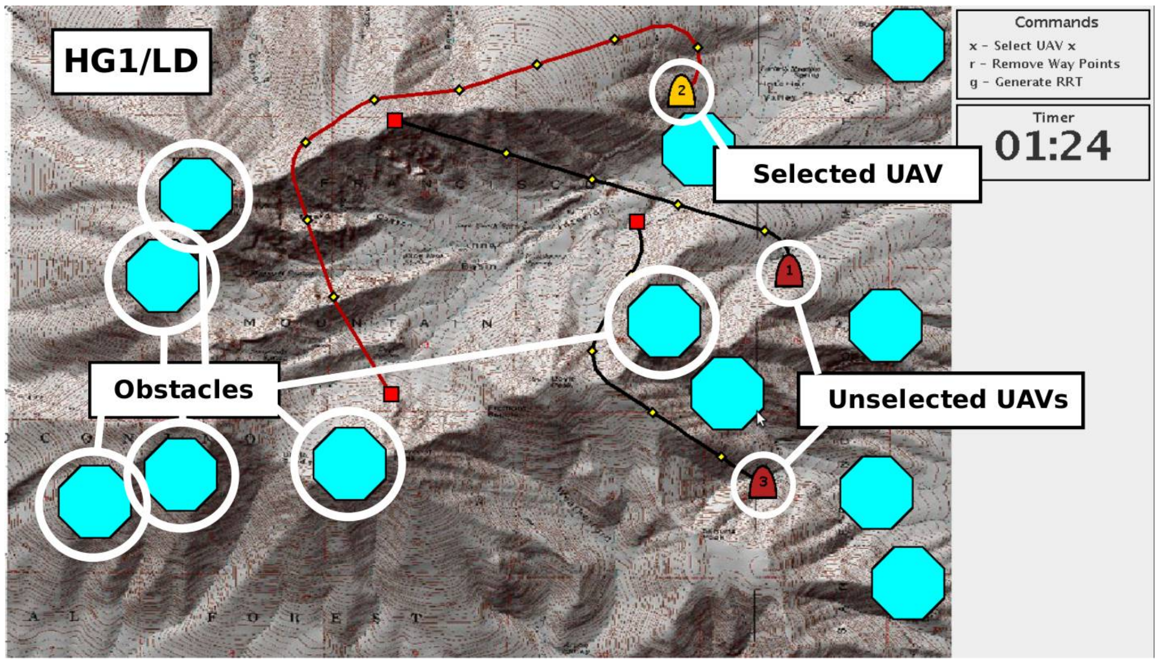


Figure 3-4: Human-Guided 1 (HG1) Mode with Low Density (LD) Obstacles

collision-free path for the selected UAV, the operator must select a sequence of at least one subgoal on the map. To do so, the human operator must first go into subgoal mode by pressing “s”, then add the desired subgoals by left-clicking on the map at the desired locations , and finally press “g” to ask the c-RRT algorithm to search for a collision-free path that goes through the specified subgoals. By pressing “s”, the interface toggles in and out of subgoal mode. When in subgoal mode, a black straight-line path from the selected UAV, through any subgoals, to the UAV’s next target appears. Subgoals can be added, deleted, and moved in the same way that waypoints can be manipulated. Once the algorithm completes its search, the planner automatically toggles out of subgoal mode: participants were trained that this was the *modus operandi* of the subgoal mode. A screenshot of the interface in **HC** mode is shown in Figure 3-5.

For the experiment, the human was not allowed to add waypoints in the **HG1**, **HG2**, and **HC** modes with the idea that the human operator should have the ability to modify the c-RRT solution, but not manually generate completely new paths. Thus, regardless of whether a path is found by the c-RRT algorithm, waypoints with which the operator can interact by moving or deleting are embedded at periodic intervals in the selected UAV’s

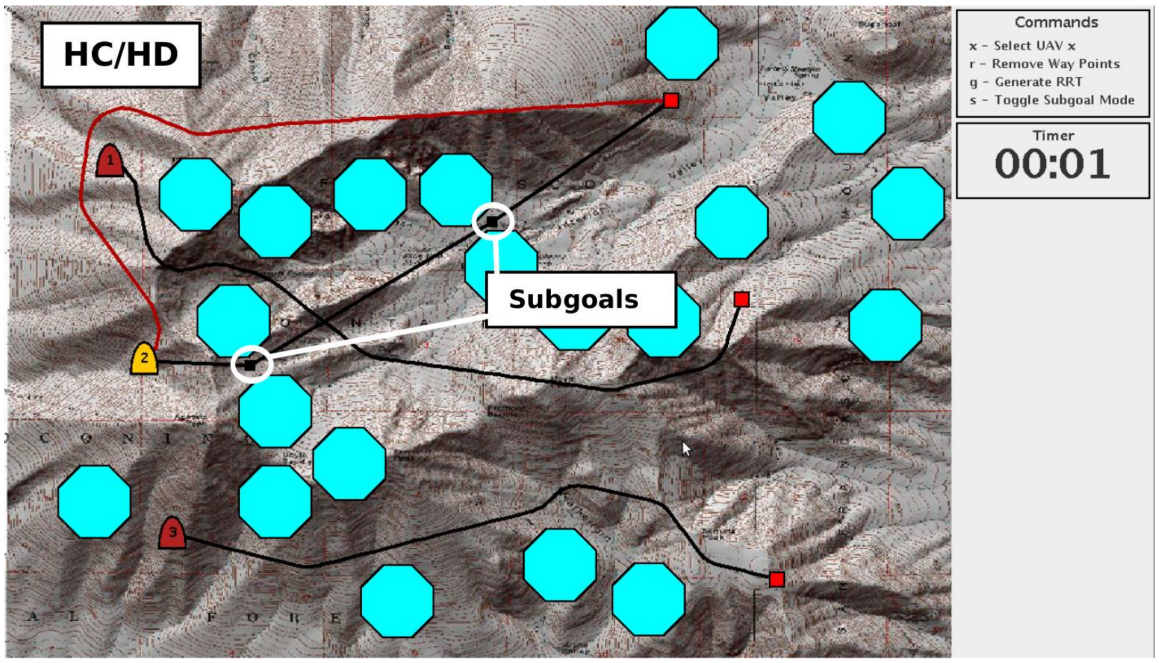


Figure 3-5: Human-Constrained (HC) Mode with High Density (HD) Obstacles

path. The separation between the points was chosen empirically such that they were close enough to give the operator enough freedom to change the path, but not so close that minor tweaks create paths with many loops due to the nature of the Dubins curves.

While the use of keystrokes, such as "g" and "s" for interfacing with the c-RRT algorithm is not appropriate for a final interface, the interface designed for this experiment is an engineering interface that acts as a proof of concept. Future work will address the improvements to be made in this display.

The c-RRT-generated paths are the paths that are returned as solutions by the c-RRT algorithm and are presented to the operators with embedded waypoints that can be modified. When a waypoint is modified manually, the part of the path that is recomputed becomes a waypoint-defined path, while the part closest to the UAV is still the c-RRT generated solution.

This is shown in Figure 3-6, where the first part of the path for UAV 3 was generated by the c-RRT and the second part is a waypoint-defined path. The distinction between the two types of paths is slight, but can be made by comparing the path segment between the first and second waypoint, and the path segment between the fourth and fifth waypoint

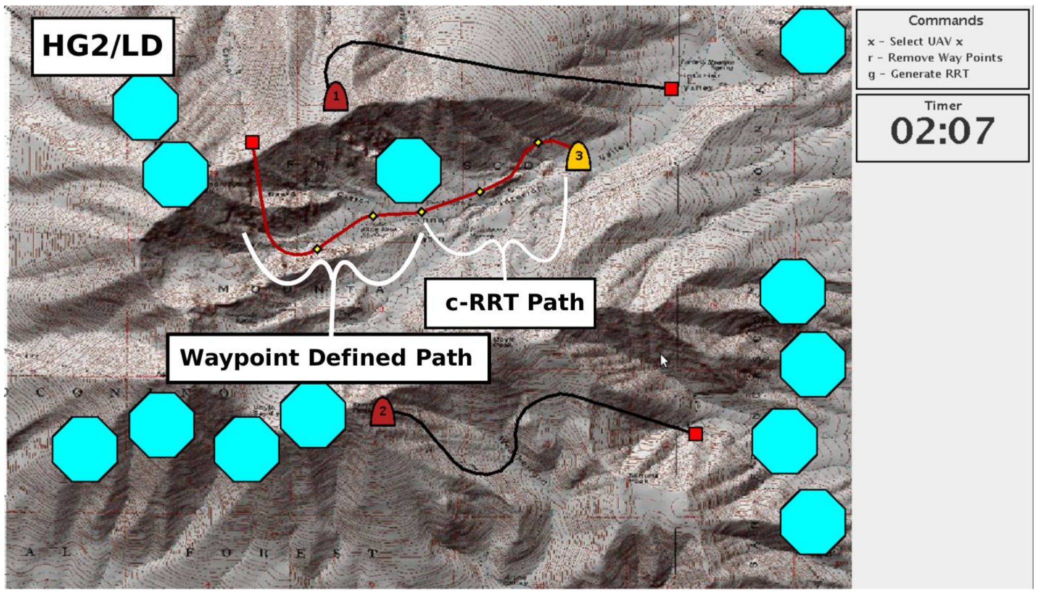


Figure 3-6: Human-Guided 2 (HG2) Mode with Low Density (LD) Obstacles

(denoted as segments *a* and *b*, respectively). The heading at the first and fourth waypoint is about the same, but while segment *b* is an almost straight line to the fifth waypoint, segment *a* is more of an "S" shape. Since waypoint-defined paths are created by connecting the waypoints using the shortest Dubins path, this indicates that *a* was generated by the c-RRT algorithm, while *b* is a waypoint-defined segment of the UAV's path.

3.2.4 Experimental Protocol

The experiment was designed to last at most an hour, with an expected duration of 45 minutes. Participants signed an informed consent form, and filled out a demographic form requesting information such as age, job experience, and previous video game experience (Appendix A and B). Next, a two-part training session was completed. The first portion, approximately 10 minutes, consisted of participants reviewing training slides (Appendix C). These slides include a brief introduction to the UAV mission planning problem to provide context for the experiment, and information on how to supervise the UAV mission planning using the GUI in the four different modes of operation: (i) human-only (**HO**), (ii) human-guided 1 (**HG1**), (iii) human-guided 2 (**HG2**), and (iv) human-constrained (**HC**).

Participants were given ten minutes for practice. The participants were given an opportunity to become accustomed to the controls by completing the task of guiding two UAVs from a starting position, each to two pre-assigned targets, through a simple, medium-density obstacle field in each of the four modes of operation. The experiment began once the ten minutes of practice were over.

Every participant was asked to complete four missions, one in each mode of operation, with the objective of avoiding as many collisions as possible while reaching all the targets as fast as possible. The four missions were carried out in either an **HD** or an **LD** obstacle field. To avoid confounding among variables, the obstacle density given to the participants and the order in which the different modes were presented to the participants was randomized and counter-balanced. After each mission, the participants were given a survey in which they were asked several subjective questions including an assessment of their performance, the workload the participant felt, and the level of frustration felt while completing the mission (Appendix D). Final interviews were completed to debrief the participant.

3.2.5 Variables and Measurements

From the experimental protocol, the independent variables in this experiment are the planner's mode of operation, and the obstacle density. The experiment was a 4 (**HO**, **HG1**, **HG2**, **HC**) $\times$ 2 (**HD**, **LD**) repeated measures analysis of variance (ANOVA) with one within (mode of operation) and one between (obstacle density) factor. The dependent variables that were examined in order to asses the usefulness of the c-RRT algorithm for UAV mission planning included the algorithm runtime, algorithm solution length, average mission completion time, operator interaction count, obstacle collision count, algorithm replan count, self-assessed performance, operator frustration, and operator workload.

The dependent variables were observed from recorded data during the experiment. The algorithm runtimes and solution length were recorded during the mission, while average mission completion times were recorded as the mean time it took for each of the three UAVs to travel from its starting position to its final target. Operator interaction count was measured by counting the number of times a participant added, moved, or deleted a waypoint or subgoal, while obstacle collision count is simply the sum of number of times the three UAVs collided into an obstacle during a mission. The algorithm replan count was taken as the number of times participants asked the c-RRT algorithm to find a path to the next target. The remaining three dependent variables are all subjective measures and were recorded by having the participants fill out a survey after every mission (Appendix D).

3.3 Relation to Research Questions

Chapter 1 introduced the three research questions addressed in this thesis.

- **Research Question #1: In what ways does having a HITL impact the algorithm's performance?** This first question looks at the effect of a human operator input on the runtime and quality of the algorithm, and can be answered by comparing the algorithm runtime and length dependent variables in the different planner modes and obstacle densities. The research question is addressed by comparing the two modes of operation that require the use of the RRT, the **HG2** and the **HC** modes, since one mode requires the operator to specify subgoals while the other mode does not allow the operator to specify any.

- **Research Question # 2: How does the RRT algorithm impact the human operator's performance?** The second research question asks how the algorithm affects the performance of the human operator in supervising UAV missions. This second research question can be answered by looking at the interaction between mission completion time, interaction count, collision count, and replan count against mode of operation and obstacle density.

- **Research Question # 3: What is a human operator's subjective assessment of using a randomized algorithm?** The third research question looks at the human operator's subjective assessment of the use of the c-RRT algorithm for collaborative UAV mission planning. This third question can be answered by looking at the relation between the participant's responses to the subjective questions and mode of operation and obstacle density. Observing these relations can lead to conclusions whether the algorithm's random nature, obstacle density, or a combination of these are causing human operators problems during UAV mission planning.

This chapter introduced the experiment and briefly discussed the approach for addressing the research questions. The approach will be discussed in greater detail along with the results of the experiment in the following chapter.

Chapter 4

Experimental Results

This chapter presents the analysis of the data collected from the experiment. The first section outlines the experimental design and the methods used for the data analysis. The second section then revisits the primary research questions of this thesis, and several aspects of each of the research questions are analyzed using the data from the experiment. Finally, the third section discusses the results of the data analysis and some common behaviors observed during the experiment.All descriptive statistics for all dependent variables can be found in Appendix E.

4.1 Experiment Design and Analysis

A total of 48 participants were recruited for the experiment: 32 males and 16 females. The participants first signed a consent form to participate in the experiment (Appendix A) and filled out a demographic survey (Appendix B). This was followed by a 10-minute self-guided training presentation (Appendix C). Following the presentation, the participants proceeded to train for 10 minutes in steering the UAVs using the 4 modes of operation: human-only (**HO**), human-guided 1 (**HG1**), human-guided 2 (**HG2**), and human-constrained (**HC**).

Each participant completed four UAV missions, each mission in one of the four modes of operation. The participants were asked to complete all four of the missions in either a high density (**HD**) or low density (**LD**) obstacle environment. This experiment was designed as a repeated measures experiment where the mode of operation is the within subject factor, and obstacle density is the between subject factor of the experiment.

The data analysis was systematically completed using the statistical package *R* [60]. First an omnibus test was conducted to see if there was an additive effect from either of the individual factors or from the interaction of the two factors. If the ANOVA/MANOVA showed significant effects, then multiple pairwise comparisons were performed using Tukey Honest Significant Difference (HSD) tests to see which pairs of levels of a factor have means that are significantly different. In addition to these pairwise comparisons, interaction plots were generated to visualize interaction effects and box plots were generated to visualize the distribution of the data. Since the assumptions of the ANOVA/MANOVA are not met by the discrete scale used by the survey questions, the analysis of the subjective data was conducted using the Wilcoxon Signed Rank and the Wilcoxon Rank Sum non-parametric tests. All tests were performed at the $\alpha = 0.05$ level.

4.2 Answering the Research Questions

The differing levels of interactions with the c-RRT algorithm are represented in the experiment analysis as the four modes of operation, while the environment complexity is associated with obstacle density and is represented in the experiment analysis as either a **HD** or **LD** obstacle environment.

4.2.1 Human's Impact on Algorithm

To address the first research question regarding the human's impact on the performance of the c-RRT algorithm, two measures were considered: (1) average of the c-RRT run times (averaged over all the missions), and (2) the average of the path length ratio for each mission.[1]

Since participants were not allowed to use the c-RRT algorithm in **HO** mode and were not required to use the algorithm in the **HG1** mode, the analysis was conducted

[1]For a c-RRT solution, the path length ratio is the length of the c-RRT solution divided by the straight line distance from the start to the goal.

Table 4.1: Repeated Measures ANOVA - Average c-RRT Runtime

	num df	den df	F	p
Mode of Operation	1	46	0.596	.4441
Obstacle Density	1	46	34.531	0^+
Interaction	1	46	5.699	.0211

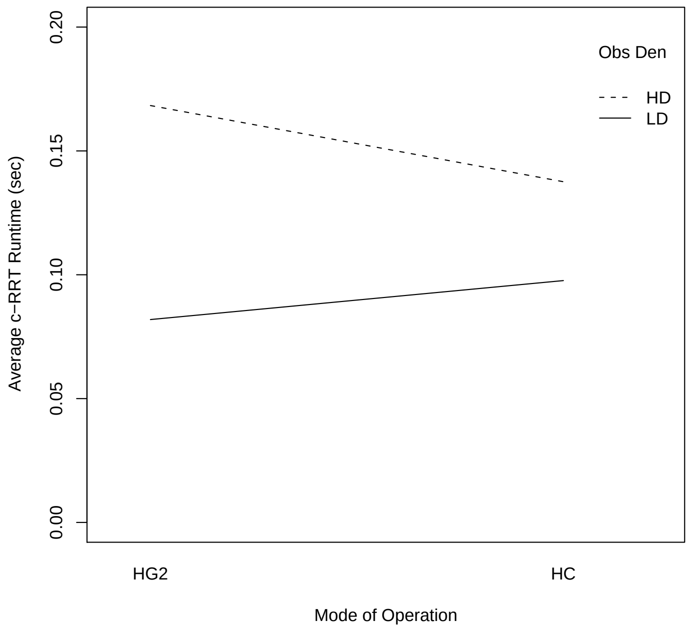


Figure 4-1: Interaction Plot for Average c-RRT Runtimes

only considering the **HG2** and **HC** modes. For the **HG1** mode, where participants were given the option of not using the c-RRT algorithm to replan, 28.3% of the participants chose not to use the algorithm at all. They instead chose to guide the UAVs by modifying the existing waypoints.

The results of the two-way repeated measures ANOVA, shown in Table 4.1, demonstrates a significant effect of obstacle density on the average runtime of the c-RRT (F(1,46)=34.531, $p<0.001$), and a significant interaction effect (F(1,46)=5.699, p=.02). However there was no significant effect for the mode of operation.

From Figure 4-1 it can be seen that the the average c-RRT run times for the **HD** missions tend to be higher than the average run times for the **LD** case. Since the c-RRT's performance was expected to suffer in higher complexity environments, this increase in the average c-RRT runtimes in the more complex high density obstacle environments is not a surprising result. However, the significant interaction effect is an interesting result.

Table 4.2: Repeated Measures ANOVA - Average c-RRT Path Length Ratio

	num df	den df	F	p
Mode of Operation	1	46	1.769	.1900
Obstacle Density	1	46	1.152	.2887
Interaction Effect	1	46	0.350	.5571

The interaction plot shown in Figure 4-1 suggests that guiding the c-RRT algorithm's search in a high density obstacle environment by specifying subgoals (as is done in the **HC** mode) reduces the runtime of the algorithm. For the scenarios chosen in this experiment, the reduction in the average c-RRT algorithm's runtime between the two modes for the **HD** environment is on the order of hundredths of a second, which may be of limited practical significance for these scenarios. However, these results are promising and hint at the fact that this trend could be more apparent in more complex environments. The increase in runtime between the **HG2** and the **HC** mode for **LD** obstacle environments is interesting and suggests that specifying subgoals is detrimental to runtime in low-complexity environments.

Table 4.2 shows that the repeated measures ANOVA indicates no statistically significant effect from any of the factors for path length ratio. Furthermore, Pearson's product-moment correlation test was performed on the average algorithm runtime and path length ratio to determine whether there was a correlation between algorithm runtime and path length ratio. Pearson's test returned r=.126 and p=.2201, thus the null hypothesis of r=0 cannot be rejected.

4.2.2 Algorithm's Impact on Human

To address the second research question (the c-RRT algorithm's impact on a human operator's performance), four aspects of the experiment were investigated: (1) average mission completion time, (2) collision count, (3) human interaction count, and (4) replan count. The average mission completion time was measured by taking the average of the completion times for each of the three UAVs, while the collision count is simply the total number of times the three UAVs collided with no-fly zones during a mission. The human interaction count is the sum of the number of times the human operator added, moved,

Table 4.3: Repeated Measures ANOVA - Average Mission Time

	num df	den df	F	p
Mode of Operation	3	138	1.872	.1371
Obstacle Density	1	46	10.16	.0026
Interaction Effect	3	138	0.255	.8580

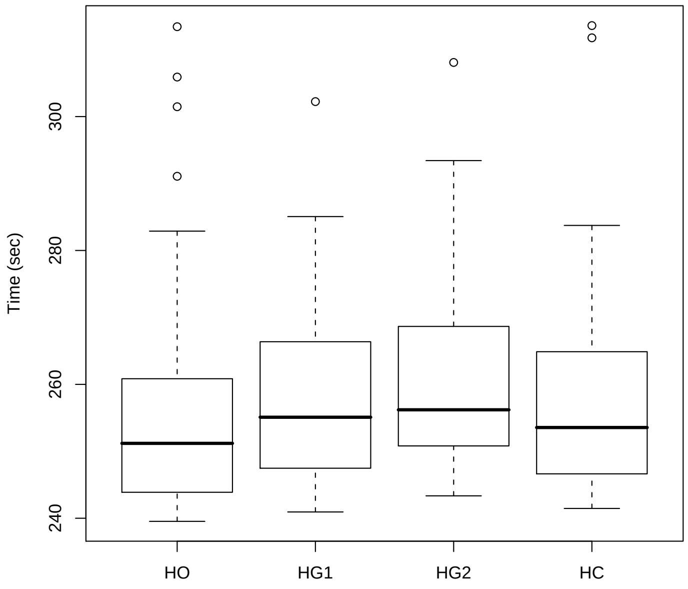


Figure 4-2: Box Plot of Average Mission Times

or deleted a waypoint or subgoal, and the replan count is the number of times that the human operator ran the c-RRT algorithm during a mission. Pearson's correlation test returned r=-.028 and p=.7483. Both metrics are further analyzed in the next sections.

The two-way repeated measures ANOVA results for the average mission completion time are shown in Table 4.3. The repeated measures ANOVA demonstrates that there is a highly significant main effect of obstacle density on the average mission completion time (F(1,46)=10.16, p=.003). This result is consistent with intuition since the UAVs have to go between and around more obstacles in the **HD** than the **LD** obstacle environments.

A box plot of the data separated by mode of operation is presented in Figure 4-2. From the box plot, it can be concluded that the effect is not practically significant since the difference between the average mission completion times is on the order of a few seconds.

For the collision count data, the repeated measures ANOVA, shown in Table 4.4, demonstrates a highly significant main effect from mode of operation (F(3,138)=7.648,

Table 4.4: Repeated Measures ANOVA - Number of Collisions

	num df	den df	F	p
Mode of Operation	3	138	7.648	0^{+}
Obstacle Density	1	46	0.621	.4348
Interaction Effect	3	138	0.507	.6779

Table 4.5: Tukey HSD Tests - Number of Collisions

	HO v HG1	HO v HG2	HO v HC	HG1 v HG2	HG1 v HC	HG2 v HC
num df	4	4	4	4	4	4
den df	141	141	141	141	141	141
q	2.106	1.248	4.135	0.858	6.242	5.384
p	.4464	.8138	.0208	.9297	.0001	.0012

p<.0001). Posthoc Tukey HSD Tests (shown in Table 4.5) indicate that the number of collisions were higher in the **HC** mode than in any other mode, while the null hypothesis could not be rejected for pairwise comparisons between the **HO**, **HG1**, and **HG2** modes. To replan a path for a UAV in order to prevent collisions, in the **HC** mode, participants were not only forced to use the c-RRT algorithm, but they were given the additional task of specifying subgoals. Observation and the subjective data from the survey suggests that the increased workload and participant's placement of the subgoals resulted in the higher collision count. Figure 4-3 shows the total number of collisions across the four different modes of operation.

Table 4.6: Repeated Measures ANOVA - Number of Human Interactions

	num df	den df	F	p
Mode of Operation	3	138	3.014	.0322
Obstacle Density	3	138	0.148	.7022
Interaction Effect	3	138	2.529	.0598

The two-way repeated measures ANOVA results for the number of human interactions are shown in Table 4.6. For the interaction effect and the main effect from mode of operation, the null hypothesis could not be rejected. The interaction plot from the human interaction count in presented in Figure 4-4.

The interaction plot indicates that there was an interaction effect present between the **HO** and **HG1** modes of operation, where the plot shows that the number of human interactions is higher for the **HD** obstacle environments in the **HO** mode of operation, but

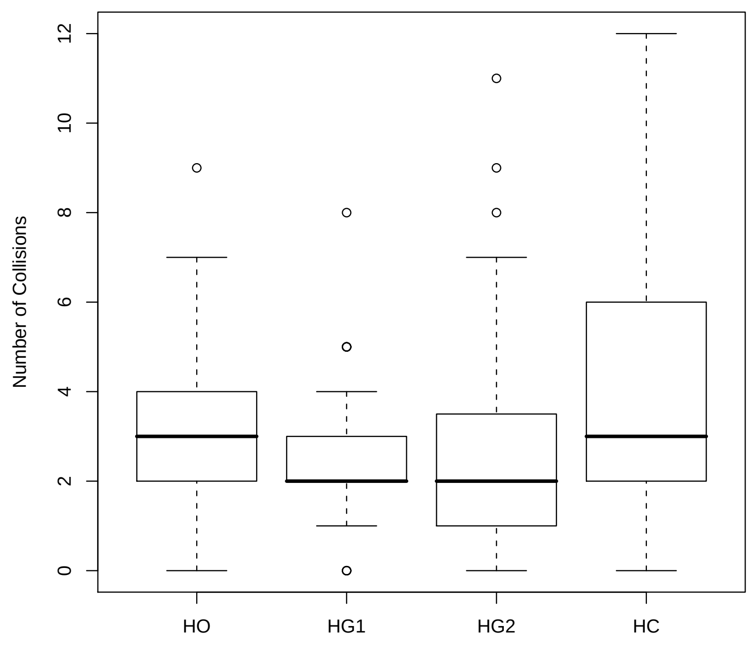


Figure 4-3: Box Plot of Number of Collisions

then switches and becomes higher for the **LD** obstacle environments in the **HG1** mode of operation. One possible explanation for this result is that participants required fewer interactions to move waypoints in the **HG1** mode rather than having to add waypoints in the **HO** mode in the **HD** obstacle environment. The increased number of interactions in the **HO** mode may be attributed to the inherent complexity of satisfying the Dubins path constraints, while for the **LD** obstacle environments, participants were inclined to interact with the interface due to lower environment complexity and remove the unnecessary waypoints that were automatically placed on the UAV paths in the **HG1** mode of operation. These observations were made from the video recordings of the interface during the experiments.

Table 4.7: Tukey HSD Tests - Number of Human Interactions

	HO v HG1	HO v HG2	HO v HC	HG1 v HG2	HG1 v HC	HG2 v HC
num df	4	4	4	4	4	4
den df	141	141	141	141	141	141
q	0.366	3.428	1.634	3.794	2.000	1.794
p	.9939	.0772	.6559	.0403	.4927	.5842

The results of Posthoc Tukey HSD tests, shown in Table 4.7, indicate that there is a significant difference in the number of human interactions between the **HG1** and **HG2**

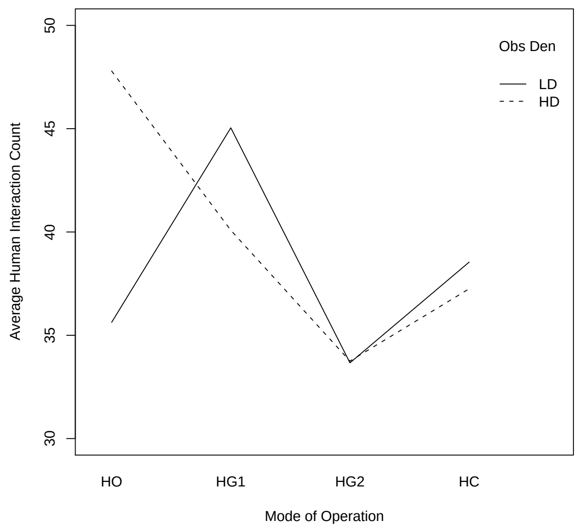


Figure 4-4: Interaction Plot for Human Interaction Count

Table 4.8: Repeated Measures ANOVA - Number of Replans

	num df	den df	F	p
Mode of Operation	2	92	43.90	0^+
Obstacle Density	2	92	2.237	.1416
Interaction Effect	2	92	0.021	.9792

modes and a marginally significant difference between the **HO** and **HG2** modes, where the number of interactions is higher for the **HG1** and **HO** modes. These differences can be explained by the fact that participants were not forced to use the c-RRT algorithm in the **HO** and **HG1** modes, hence choosing to manually steer the UAVs around the obstacles. The average number of human interactions for the **HC** mode was between the number of human interactions for the **HG2** mode and the **HO** and **HG1** modes (Appendix E). One possible explanation for this is that like the **HG2** mode, participants were forced to use the c-RRT algorithm to change a UAV's path, but each time the algorithm was called, participants had to specify at least one subgoal, thus increasing the number of interactions.

Since the **HO** mode does not allow the human operator to utilize the c-RRT algorithm to replan, there are only three levels to the mode of operation factor for the analysis of the replan count: **HG1**, **HG2**, and **HC**. The two-way repeated measures ANOVA results in Table 4.8 show a significant main effect from the mode of operation (F(2,92)=43.90, p<.001). The posthoc Tukey HSD tests, shown in Table 4.9, indicate a high level of

Table 4.9: Tukey HSD Tests - Number of Replans

	HG1 v HG2	HG1 v HC	HG2 v HC
num df	3	3	3
den df	94	94	94
q	13.138	5.077	8.062
p	0^+	.0015	0^+

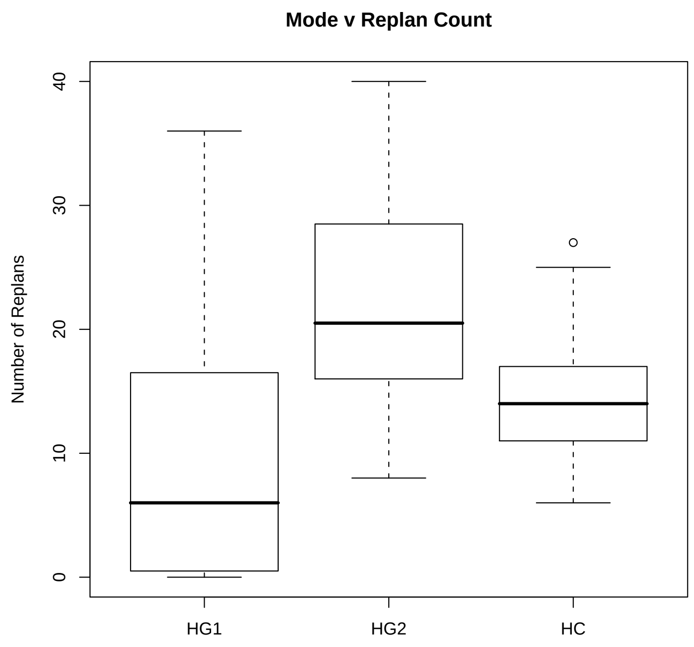


Figure 4-5: Box Plot of Replan Count

significance between all three pairs of modes (**HG1** v **HG2**, **HG1** v **HC**, and **HG2** v **HC**).

The box plot in Figure 4-5 reveals that the **HG1** mode has the lowest replan count, with **HC** next, and **HG2** trailing with the highest replan count. This is consistent with intuition that **HG1** has the lowest replan count since the **HG1** mode does not force the human operator to use the algorithm and almost one third of participants chose to not even replan once using the algorithm. Participants were forced to call the c-RRT algorithm to replan in both the **HG2** and the **HC** modes. First hand observation and subjective feedback suggest that the lower replan count for the **HC** mode as compared to the **HG2** mode can be attributed to the increased effort required to use the c-RRT in the **HC** mode. This increased effort means that participants can physically call the c-RRT algorithm a smaller number of times in the **HC** mode than in the **HG2** mode. The additional workload may have even discouraged participants from using the algorithm and either fixing the path manually or diverting their attention elsewhere and allowing UAVs to collide. This additional effort in using the c-RRT in the **HC** mode comes from the fact that human operators have to go through the additional trouble of going into subgoal mode and deciding where and how many subgoals to set before calling the algorithm.

4.2.3 Subjective Assessment of Collaborative RRT Planner

The responses given to three survey questions during the experiment were analyzed to address the research question regarding the subjective assessment of the c-RRT planner. The first question asked the participants to evaluate their level of performance during the mission, the second to rate their level of frustration, and the last asked the participants to assess the workload during the previous mission.

For performance, level of frustration, and self-assessed workload, the results of the Wilcoxon Signed Rank Tests, shown in Tables 4.10(a-c), indicate significant differences between the **HC** mode and the other three modes of operation. The responses to the survey questions suggest that participants did not prefer the **HC** mode of operation because of the increased workload from forcing participants to place subgoals before every replan, the increased number of collisions (which was likely a combined result of increased workload and subgoal placement), and frustration from the increased workload and the behavior of the c-RRT path planner.

Table 4.10: Non-Parametric Test Results of Survey Responses

(a) Performance

Wilcoxon Signed Rank Test		
	W	p
HO v HG1	186	.466
HO v HG2	159	.669
HO v HC	461	.003
HG1 v HG2	197	.847
HG1 v HC	465	.001
HG2 v HC	343	.001
Wilcoxon Rank Sum Test		
	U	p
LD v HD	4366	.503

(b) Frustration

Wilcoxon Signed Rank Test		
	W	p
HO v HG1	116.5	.310
HO v HG2	120	.127
HO v HC	91.5	0^+
HG1 v HG2	105	.713
HG1 v HC	93.5	.002
HG2 v HC	125.5	.006
Wilcoxon Rank Sum Test		
	U	p
LD v HD	5355.5	.041

(c) Workload

Wilcoxon Signed Rank Test		
	W	p
HO v HG1	54	.074
HO v HG2	54	.236
HO v HC	37.5	0^+
HG1 v HG2	124.5	.436
HG1 v HC	44	.002
HG2 v HC	56	0^+
Wilcoxon Rank Sum Test		
	U	p
LD v HD	4409	.562

The only other significant result shown in the analysis of the three survey questions is demonstrated by the result of the **LD** v **HD** Wilcoxon Rank Sum Test on the frustration level data, the results of which are also shown in Table 4.10. Higher density obstacle environments generally caused higher frustration, but curiously, did not cause any noticeable increase in subjective workload.

4.3 Discussion

The results addressing the first research question (dealing with the humans impact on the algorithm) showed that in the more complex **HD** environment, constraining the solution space of the c-RRT algorithm by specifying subgoals reduced the algorithm's runtime, while neither human input or obstacle density had a significant effect of the quality of the c-RRT solution. The data indicates that constraining the solution space of the c-RRT algorithm may result in detrimental performance and negative subjective feedback. However, the improvement in the runtime and reduction in replan count may have applications in other domains.

The results addressing the second research question (dealing with the algorithm's impact on the human operator's performance) do not provide evidence to conclusively answer the research question. The average mission runtime data does not demonstrate a statistical or practical difference between the different modes of operation. Since the missions were about five minutes each, some difference in the mission times between the **HO** mode and the c-RRT modes was expected due to the sub-optimality of the c-RRT solutions. It is unclear why this difference was not present, but one possibility is that the presence of down time during the missions allowed operators to manipulate solutions to the point that they were similar enough to the manually generated solutions. The collision count data indicates that there is no significant difference across the modes, save for the **HC** mode. This is probably a result of the experiment design since the use of the c-RRT algorithm is still a fairly hands-on manual task requiring the human operator to recognize upcoming collisions, then replan by invoking the c-RRT algorithm.

While the data obtained from the experiment provides a significant amount of in-

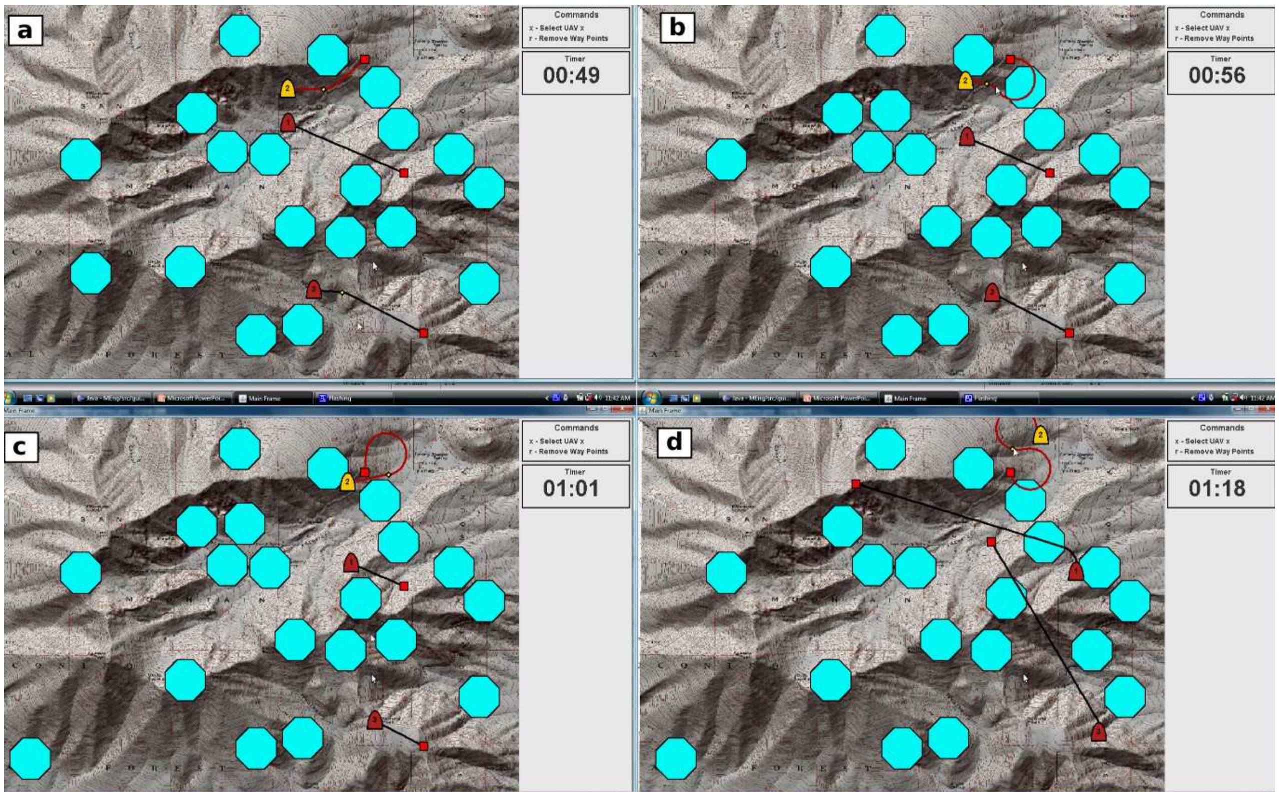


Figure 4-6: Experiment Screenshot - Human Only Mode Delay
(a) UAV 1 on course to reach its next target (on top of the figure), (b) human moves waypoint and no longer has collision-free trajectory due to curvature constraints, (c) human moves waypoint again to obtain collision-free path to target, (d) human tries adding another waypoint and further delays UAV 1's arrival at target.

formation for assessing the usefulness of an RRT-based planner for UAV mission path planning, there are several aspects of the human-automation interaction that the data cannot express. To address these aspects that are not captured by the data, the following section discusses a few common behaviors that were observed during the experiment.

4.3.1 Common Behaviors

Several common behaviors were observed in the experiment, and are shown in this section. The UAV that is being referred to in each of the following figures is the selected UAV in each of the screenshots, and is depicted in yellow.

It was hypothesized that humans would have difficulty in planning with the curvature constraints since they may be difficult to conceptualize. In real life, curvature constraints

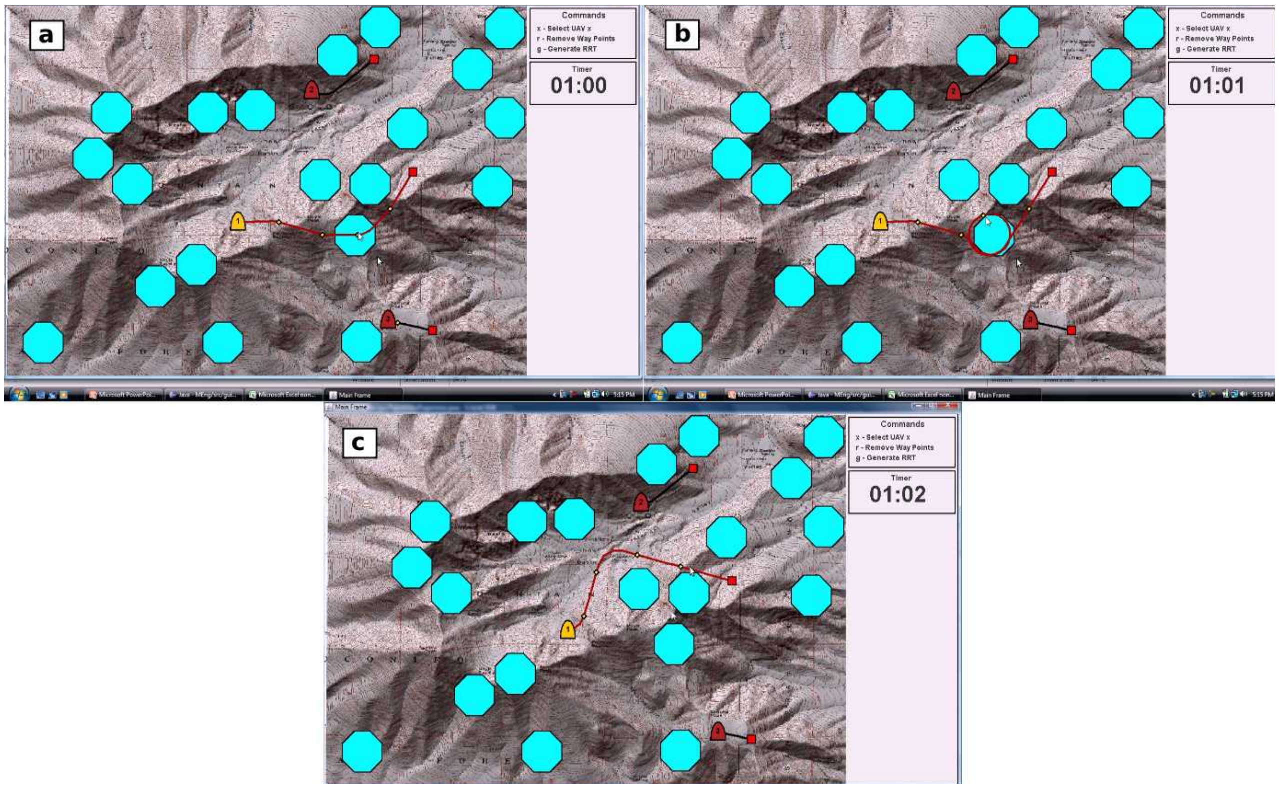


Figure 4-7: Experiment Screenshot - Using the c-RRT
(a) Obstacle appears on UAV 1's current path, (b) participant has trouble manually replanning, (c) participant uses c-RRT algorithm to find a path for UAV 1 that goes around the obstacles to the goal.

limit how quickly a UAV can turn, and present non-trivial difficulties when trying to avoid obstacles under time constraints. An example of such behavior is shown in Figure 4-6, which shows a participant in the **HO** mode having trouble adding waypoints due to curvature constraints. In frame (a), the participant seems to have set UAV 1 on a path that will lead it to the target soon, but in an apparent attempt to get UAV 1 to its target faster, the participant moves the waypoint. In frame (b), at the waypoint's new position, UAV 1 can no longer go to the target due to curvature constraints, and the new path goes through an obstacle. To avoid a collision, the participant again moves the waypoint, but can only obtain a path that requires the UAV to loop before reaching the target. Still unsatisfied, the participant adds a waypoint in frame (d), causing UAV 1 to now be on the strange "S"-shaped trajectory. Note that the participant could have avoided the whole problem by simply removing the waypoint around the 0:58 second point (after frame (b)).

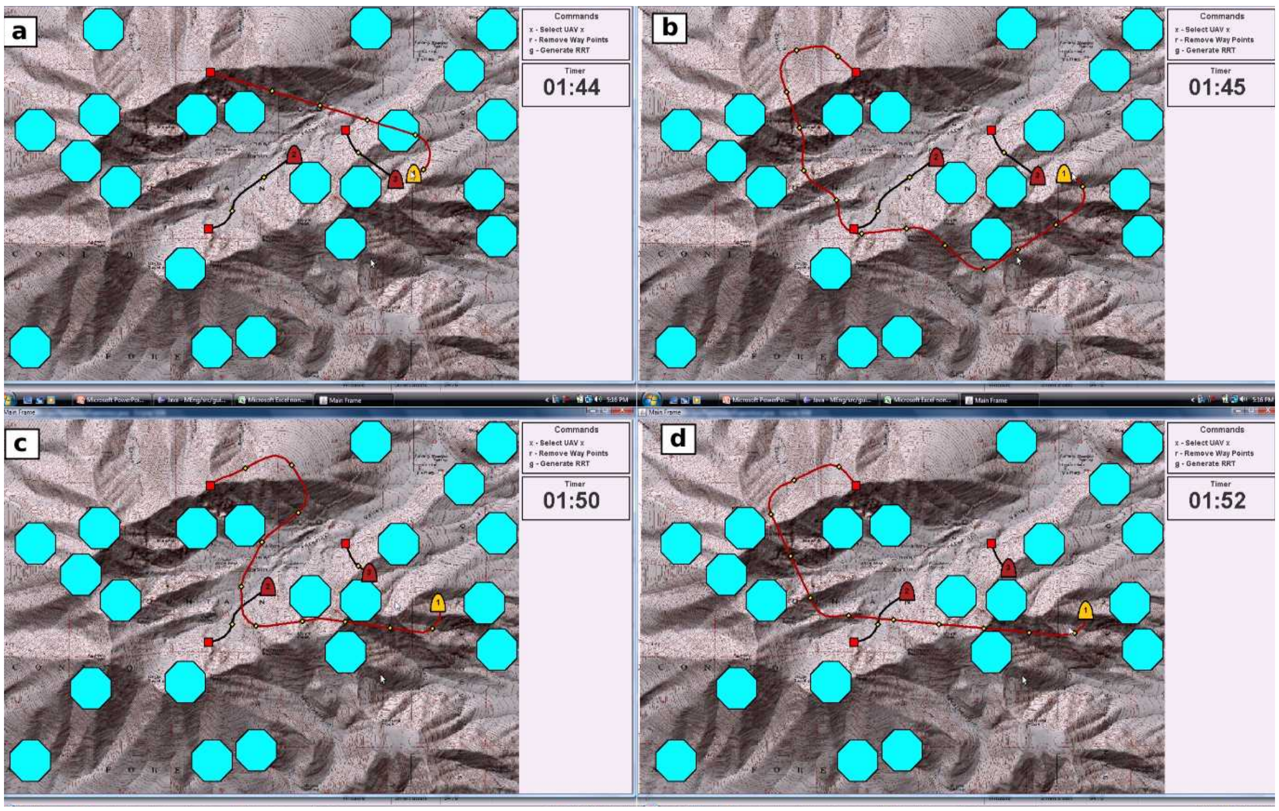


Figure 4-8: Experiment Screenshot - Sequence of c-RRT Replans
(a) Participant is unhappy with UAV 1's current path, and (b),(c),(d) replans with the c-RRT algorithm three times.

While we see that a lack of consideration of the curvature constraints when adding waypoints led to a dramatic increase in mission time for UAV 1, Figure 4-7 tells a very different story. In frame (a) of Figure 4-7, an obstacle appears on UAV 1's current path, then in frame (b), the participant tries to modify the path by moving a waypoint and runs into trouble due to curvature constraints. Finally, in frame (c), the participant gives up on fixing the path manually and asks the c-RRT to find a solution. The c-RRT solution shown in frame (c) could have been more direct by going between the obstacles as opposed to around.

When participants were unsatisfied with c-RRT generated paths, it was common to see the participant ask the algorithm to generate several paths, one right after the other. An example of this is demonstrated in the sequence of screenshots in Figure 4-8. This mission was carried out in **HG2** mode, so no subgoals were specified in each of the replans. In frame (a) of Figure 4-8, UAV 1 has a path that goes through an obstacle so the participant

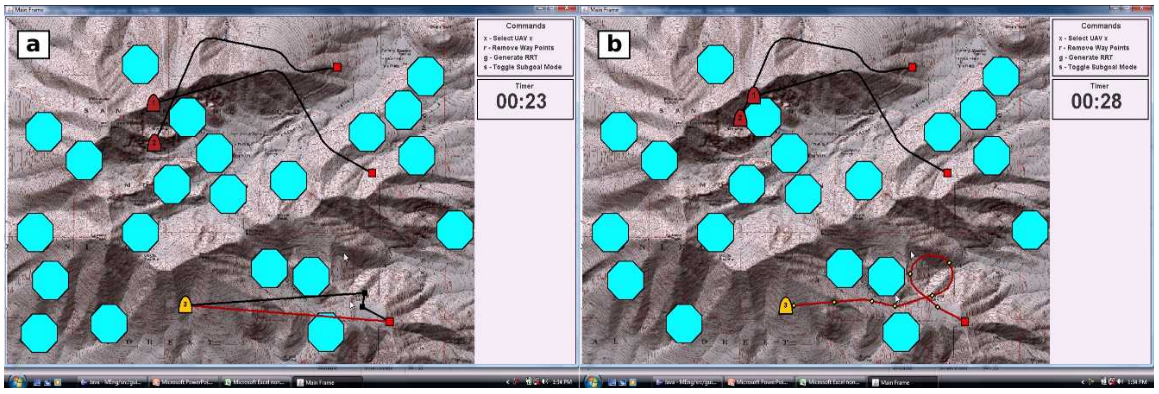


Figure 4-9: Experiment Screenshot - Poor Subgoal Sequence
(a) Participant sets subgoals for UAV 3 too close to each other, and (b)obtains strange c-RRT due to curvature constraints.

decides to use the c-RRT algorithm to replan. The first replan is shown in frame (b). The participant is not satisfied with the solution and asks the c-RRT algorithm to replan twice, shown in frame (c) and (d). Note that the solution shown in frame (c) is actually shorter than the solution in (d). This illustrates one of the problems of the c-RRT in which a solution may be good, but a participant decides to use the c-RRT to replan, obtaining a less desirable solution. When participants were unsatisfied with a c-RRT solution, they would either keep calling the c-RRT algorithm or would replan manually. For the modes in which participants were allowed to use the c-RRT algorithm, an average of 30% of the interactions were replans, as opposed to waypoint modifications and subgoal interactions.

Detrimental behavior due to a lack of consideration of the curvature constraints was most apparent in the **HC** mode when participants specified sequences of subgoals that had no feasible solution or produced bad solutions. An example of such a behavior can be observed in Figure 4-9. Notice that in frame (a), the participant places two subgoals for UAV 3 and that one is right below the other. In frame (b), the consequences of placing the two subgoals too close to each other becomes apparent as the c-RRT algorithm finds a path that goes through the two subgoals, but curvature constraints force the feasible path to have a loop in the path between the first and the second subgoal.

Even when subgoals do not cause the c-RRT algorithm to produce strange solutions, certain sequences of subgoals produce better solutions. An example of different sequences producing different length paths is shown in the comparison made in Figure 4-10. The two

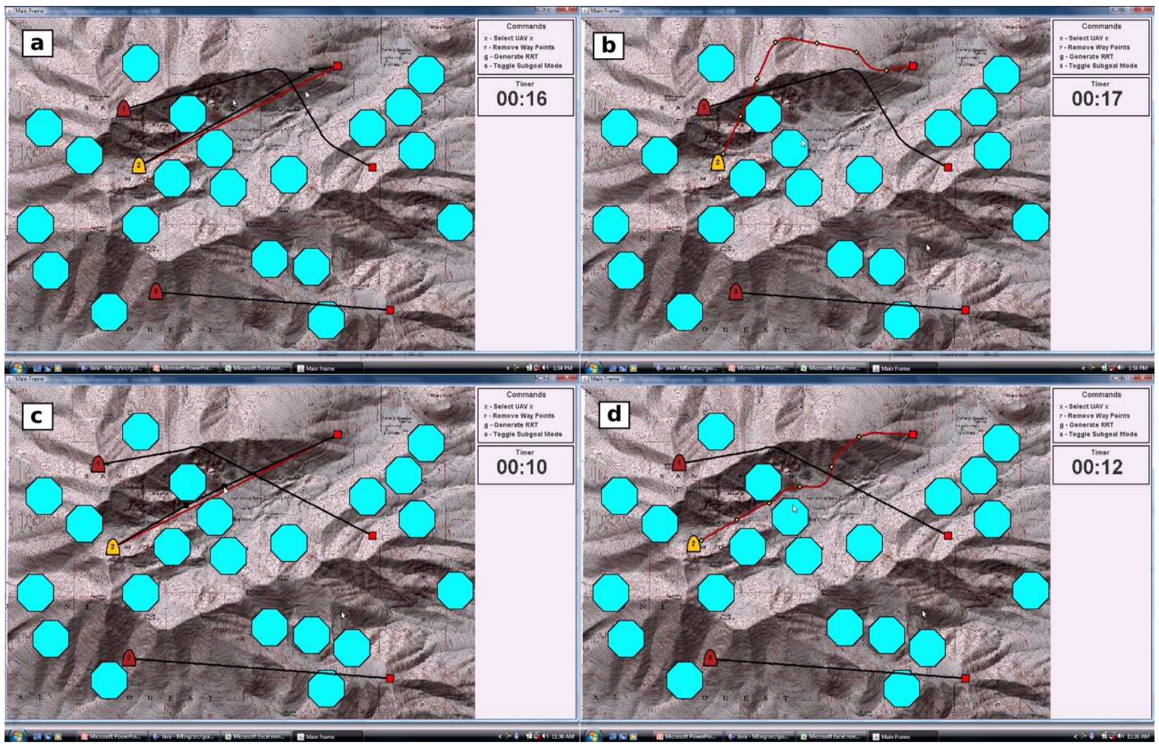


Figure 4-10: Experiment Screenshot - Taking Advantage of Subgoals
(a),(b) Participant sets a subgoal close to target and gets c-RRT solution that takes UAV 2 the long way around the obstacles. (c),(d) Another participant sets subgoals for UAV 2 in between obstacles and c-RRT returns a path that takes the shorter path to the target between the obstacles.

examples are carried out in the same **HD** obstacle map in **HC** mode. Frame (a) shows a single subgoal for UAV 1 placed the target, while frame (c) shows the subgoals for UAV 1 placed between obstacles. The solution the c-RRT algorithm produced given the subgoal shown in frame (a) is shown in frame (b), while the solution the c-RRT algorithm produced given the subgoals in frame (c) is shown in frame (d). The solution in frame (d) goes between the obstacles, and is therefore shorter than the solution shown in frame (b) which goes around the obstacles. While the subgoal shown in frame (a) could have produced a path that goes between the obstacles, the subgoals from frame (c) force the solution to go between the obstacles. Frames (c) and (d) demonstrate the use of subgoals to guide the algorithm and produce a shorter path.

There is one other behavior worth discussing that was evident across the **HG1**, **HG2**, and **HC** modes. Regardless of whether the c-RRT solutions were good or not, many participants would move and delete as many waypoints as possible without creating a path that went through an obstacle.[2] This behavior seems to be a product of the fact that participants were given only one task, and there were times where there was nothing productive to do, so they would try and make the paths look better (i.e., shorter) by deleting and moving waypoints. This behavior could explain the similarity in the average mission time across the modes.

The behaviors just described illustrate some of the behaviors that were commonly observed. As expected, there were cases when participants struggled with curvature constraints, and in many cases, in the modes where the algorithm was available, participants used the c-RRT algorithm to help them navigate through the obstacle fields. These behaviors were expected and observed, but due to the design of the experiment, the data does not reflect the beneficial and/or detrimental behavior of using an RRT-based path planner. The examples pertaining the the **HC** mode of operation support the data demonstrating the subjective dislike of the mode by showing that specifying subgoals takes time and can produce results that are not necessarily better than the algorithm running without subgoals or with different subgoals specified.

4.4 Summary

This chapter revisited the three primary research questions of this thesis by looking at the relevant data from the experiment. The data regarding the first research question addressing the human operator's impact on the c-RRT algorithm showed that the c-RRT algorithm runtime is affected by high complexity environments, and that setting subgoals, as was done in the **HC** mode of operation, reduces the average runtime. For the second research question regarding the algorithm's impact on the human operator, the data showed that the **HC** mode of operation was detrimental to performance, while there was no significant difference between the remaining three modes of operation. The absence of a significant difference in performance between the **HO**, **HG1**, and **HG2** modes means that the second research question could not be conclusively answered. The analysis of the

[2]Statistics on the number of waypoint and subgoals added, deleted, and moved are shown in Appendix E for human interaction count.

subjective responses did not show a significant difference between the **HO**, **HG1**, and **HG2** modes, but did show a dislike for the **HC** mode of operation. In other words, the subjective responses did not indicate a preference for using the c-RRT algorithm, but did indicate a dislike for the setting of subgoals required in the **HC** mode. The next chapter draws final conclusions and discusses directions for future work.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis has presented the collaborative Rapidly exploring Random Tree (c-RRT) algorithm to analyze how a human operator interacts with an RRT path planning algorithm in complex dynamic environments. An experiment was designed to test how participants behaved and performed in four different modes of operation and in one of two different obstacle densities. The four modes of operation varied in the level of automation and type of interaction with the automation. Participant behavior and the data obtained from the experiment was analyzed to investigate the human operator's impact on the algorithm's performance, the effect of the algorithm on the human, and the human's subjective evaluation of planning with the c-RRT algorithm.

While the results of the experiment did not conclusively answer the primary research questions of this thesis, a number of observations were made from carrying out the experiment and analyzing the data. While allowing human operators to guide the c-RRT algorithm's search by giving the algorithm subgoals did show some benefits, such as reduced replan count and decreased runtime in high density obstacle environments, the experiment conclusively showed that having humans constrain the algorithm's solution space by specifying subgoals requires improvements as it resulted in a lower actual and self-assessed performance, increased self-assessed workload, and increased level of frustration.

5.2 Future Work

The work done for this thesis is only the beginning in research in the use of RRTs for human-automation collaborative path planning, and the lack of a significant difference across a number of different metrics suggest more work is needed to understand the benefits of using an RRT algorithm in the UAV path planning domain. Two possible confounds in this experiment are that (1) participants had to manually instantiate the c-RRT algorithm for each UAV for which the participant desired algorithm to replan, and (2) participants only had one task to which they had to dedicate their attention (steering the UAVs).

Requiring participants to manually call the c-RRT algorithm seems to have detracted from one of the main benefits of automation: giving a human operator the luxury of not having to supervise and micro-manage each UAV all the time. The eventual goal is to use an RRT algorithm to make UAV path planning more autonomous, and it would therefore be beneficial to test the usefulness of the RRT for UAV path planning when the UAVs automatically replan using the RRT algorithm. In addition, participants had the sole task of supervising the path planning for three UAVs, which resulted in some down time during the mission, giving participants plenty of time to modify UAV paths regardless of whether the modifications were necessary. With this in mind, it would be interesting direction for future work to investigate how this result changes when human supervisors are asked to carry out one or more tasks (in addition to path planning), via the addition of secondary tasks.

The experiment showed that having human operations guide the c-RRT algorithm's search by specifying subgoals negatively impacted performance, self-assessed workload, and frustration level. Several of the factors that are likely to have caused decreases in performance and increased frustration can be addressed by better training the human operator or providing easier interface interactions. In a planner that utilizes an RRT in a more autonomous fashion, giving the human operator the option to constrain the solution space of the path planner in low workload can be beneficial and is one area of possible future research. Furthermore, any extensions to this work wishing to address the issue of human operators' subjective assessment of the RRT algorithm for UAV path planning should design more detailed survey questions pertaining to specific aspects of the mission planning.

This thesis fixed the number of UAVs a human operator was simultaneously responsible for supervising to 3. Thus, another possible direction for future work involves comparing the number of UAVs a human operator can successfully manually guide to the number of UAVs an RRT-based path planner can successfully guide through an obstacle field.

Once an RRT-based path planner is shown to be suitable for human-automation collaboration, extensions to the planner need to be made and studied before such a planner is put to practical use. The c-RRT algorithm was developed to find paths for UAVs in a two dimensional space, but it will be useful to eventually extend the algorithm for path planning for UAVs in three dimensions. The c-RRT algorithm uses two dimensional Dubins paths to generate solution, and the algorithm can extended to three dimensions by using the three dimensional Dubins paths for the Dubins airplane described in [50]. In addition, it may be useful to study and take advantage of other aspects of the RRT, for example, the random nature of the algorithm for stealth applications.

Appendix A

Consent Form

CONSENT TO PARTICIPATE IN
NON-BIOMEDICAL RESEARCH

Human Interaction with RRTs for UAV Mission Path Planning

You are asked to participate in a research study conducted by Professor M. L. Cummings, Ph.D., from the Aeronautics and Astronautics Department at the Massachusetts Institute of Technology (M.I.T.). You were selected as a possible participant in this study because the population this research will influence is expected to contain researchers who have experience with human interaction with path planners. You should read the information below, and ask questions about anything you do not understand, before deciding whether or not to participate.

- **PARTICIPATION AND WITHDRAWAL**

Your participation in this study is completely voluntary and you are free to choose whether to be in it or not. If you choose to be in this study, you may subsequently withdraw from it at any time without penalty or consequences of any kind. The investigator may withdraw you from this research if circumstances arise which warrant doing so.

- **PURPOSE OF THE STUDY**

The objective of this experiment is to investigate the effect of human interaction with randomized planners in path planning applications.

- **PROCEDURES**

If you volunteer to participate in this study, we would ask you to do the following things:

- Fill out a demographic survey.
- Perform an experiment lasting approximately 1 hour.
- Total time: 1 hour.

- **POTENTIAL RISKS AND DISCOMFORTS**

There are no anticipated physical or psychological risks in this study.

- **POTENTIAL BENEFITS**

While there is no immediate foreseeable benefit to you as a participant in this study, your efforts will provide critical insight into the development of a methodology that can help researchers select a set of human-automation performance metrics.

- **PAYMENT FOR PARTICIPATION**

You will be paid $10/hr to participate in this study. This will be paid upon completion of your debrief. Should you elect to withdraw in the middle of the study, you will be compensated for the hours you spent in the study. Additionally, a $100 gift certificate to Best Buy will be awarded to the top scorer of all participants of the study (averaged across trials).

- **CONFIDENTIALITY**

Any information that is obtained in connection with this study and that can be identified with you will remain confidential and will be disclosed only with your permission or as required by law.

You will be assigned a subject number which will be used on all related documents to include databases, summaries of results, etc. Only one master list of subject names and numbers will exist that will remain only in the custody of Professor Cummings.

- **IDENTIFICATION OF INVESTIGATORS**

If you have any questions or concerns about the research, please feel free to contact the Principal Investigator, Mary L. Cummings, at (617) 252-1512, e-mail, missyc@mit.edu, and her address is 77 Massachusetts Avenue, Room 33-311, Cambridge, MA 02139. The graduate student investigator is Americo Caves at (617) 258-5046, email, americo@mit.edu.

- **EMERGENCY CARE AND COMPENSATION FOR INJURY**

If you feel you have suffered an injury, which may include emotional trauma, as a result of participating in this study, please contact the person in charge of the study as soon as possible.

In the event you suffer such an injury, M.I.T. may provide itself, or arrange for the provision of, emergency transport or medical treatment, including emergency treatment and follow-up care, as needed, or reimbursement for such medical services. M.I.T. does not provide any other form of compensation for injury. In any case, neither the offer to provide medical assistance, nor the actual provision of medical services shall be considered an admission of fault or acceptance of liability. Questions regarding this policy may be directed to MIT's Insurance Office, (617) 253-2823. Your insurance carrier may be billed for the cost of emergency transport or medical treatment, if such services are determined not to be directly related to your participation in this study.

- **RIGHTS OF RESEARCH SUBJECTS**

You are not waiving any legal claims, rights or remedies because of your participation in this research study. If you feel you have been treated unfairly, or you have questions regarding your rights as a research subject, you may contact the Chairman of the Committee on the Use of Humans as Experimental Subjects, M.I.T., Room E25-143B, 77 Massachusetts Ave, Cambridge, MA 02139, phone 1-617-253 6787.

SIGNATURE OF RESEARCH SUBJECT OR LEGAL REPRESENTATIVE

I understand the procedures described above. My questions have been answered to my satisfaction, and I agree to participate in this study. I have been given a copy of this form.

____
Name of Subject

____
Name of Legal Representative (if applicable)

____	______________
Signature of Subject or Legal Representative	Date

SIGNATURE OF INVESTIGATOR

In my judgment the subject is voluntarily and knowingly giving informed consent and possesses the legal capacity to give informed consent to participate in this research study.

____	______________
Signature of Investigator	Date

Appendix B

Demographic Questionaire

Demographic Questionnaire for
"Human Interaction with RRTs for UAV Mission Path Planning"

Please indicate your sex:
o Male
o Female

Please indicate your age: ___________

Please indicate your occupation (if student, indicate your year and degree)?
____

Are you currently or have you ever served in the armed forces of any country?
____
If yes,
o Country: ____________
o Service: __Army __Navy __Air Force
o Years of service:______________

Do you have any experience with remotely piloted vehicles (land, sea, sub-sea, air)?
o NO
o YES
If yes, please state what vehicle types and number of hours:
Vehicle types: __Land __Sea __Sub-Sea __Air
Number of hours? ________

Have you participated in a controlled experiment with unmanned vehicle simulators before?
o Yes
o No

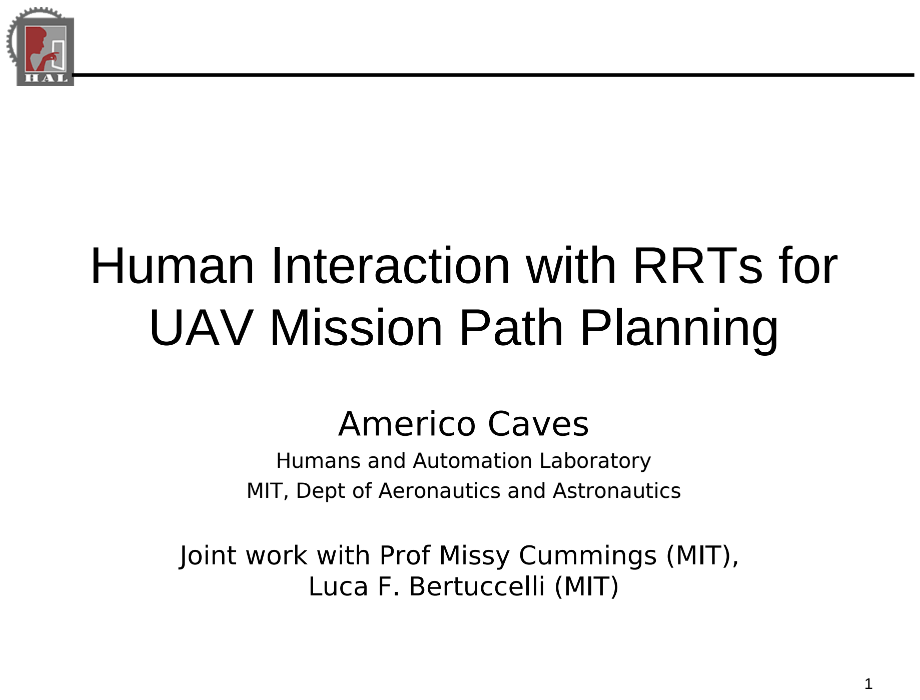
Do you currently have or do you have a history of color blindness?
o Yes
o No

How much experience do you have playing video games? _______ hr per week (on average)

Have you ever interacted with a randomized path planning algorithm?
o Yes
o No

Appendix C

Training Slides



Human Interaction with RRTs for UAV Mission Path Planning

Americo Caves

Humans and Automation Laboratory

MIT, Dept of Aeronautics and Astronautics

Joint work with Prof Missy Cummings (MIT), Luca F. Bertuccelli (MIT)

1

Outline

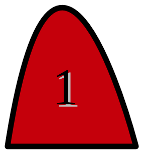
- Context
- Experiment
- Environment
- RRT – The planning algorithm
- Four Modes of Operation
  - Human Only
  - Human Guided RRT 1
  - Human Guided RRT 2
  - Human Constrained RRT
- The Training Interface
- Transitions

Context

- Unmanned Aerial Vehicles (UAVs) are used for carrying out a variety of tasks.
  - Combat, Surveillance, Firefighting
- Pathplanning is required for carrying out any of these tasks.
- Will be focusing on pathplanning for these UAV missions
- Pathplanning: give the UAV a collision free path to follow from its current position to its target

Context

- You, the participant, will be supervising UAV in a simulation of a mission
- Objectives:
  - Guide Unmanned Aerieal Vehicles (UAVs) to sequence of targets.
  - Get UAVs to targets as quick as possible
  - Avoid obstacles as best as possible

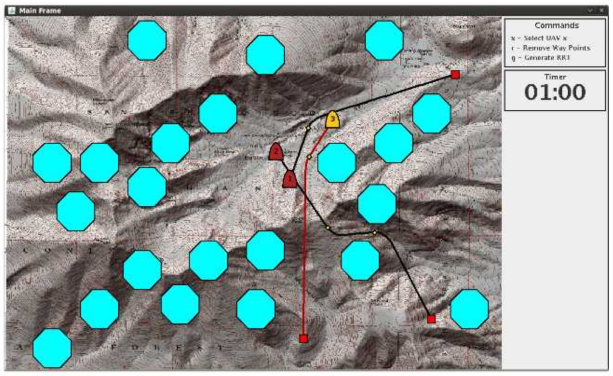


Experiment Outline

- Training (10 min)
- Demographic Survey
- Mission (5 min)
- Feedback Survey
- Mission (5 min)
- Feedback Survey
- Mission (5 min)
- Feedback Survey
- Mission (5 min)
- Feedback Survey

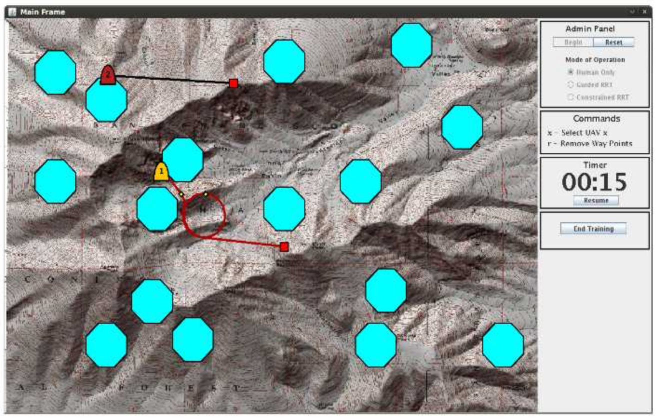
The Environment

- Blue octagons – obstacles
- Bright red squares – next target
- Red/orange shape w/ number – UAV (number is id)
- Selected UAV and Path – orange and red, respectfully
- Yellow points on paths– way points
- Timer and commands panels are on the right



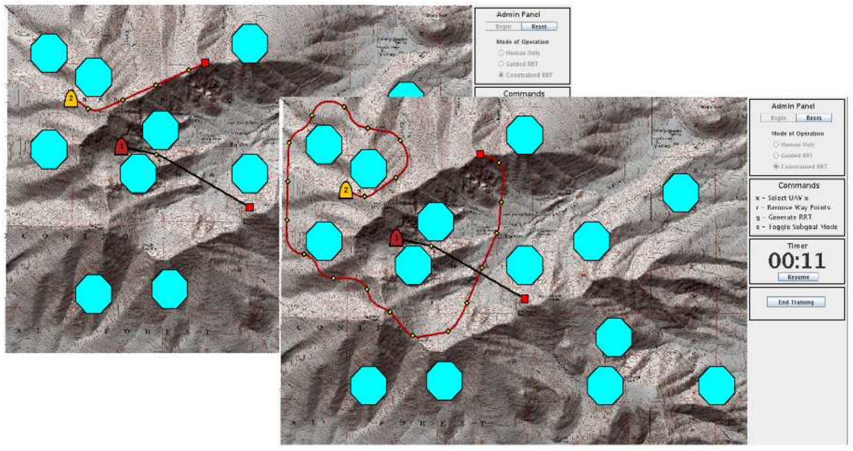
The Environment

- Obstacles appear and disapear – avoid as best you can
- If an obstacle appears and UAV is in the obstacle, it does not count as collision
- Adding a waypoint too close to another way point or to the UAV will result in a loop in the path (see UAV 1 path below).



RRT – The planning algorithm

- The solutions output by the algorithm are random
- The solution and the time it takes to find that solution will vary due to the random nature of the algorithm



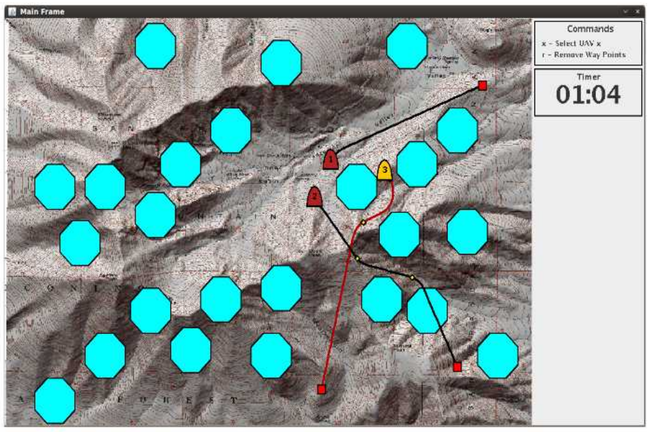
Mode of Operation: Constrained RRT

x – Select UAV x
r – Remove Way Points
g – Generate RRT
s – Toggle Subgoal Mode

Timer 00:11

Human Only Mode

- Select UAV by pressing number on keybord that corresponds to number of UAV
- Can add, delete or move waypoints for selected UAV
  - Add a way point with left mouse click
  - Delete by right clicking the desired way point

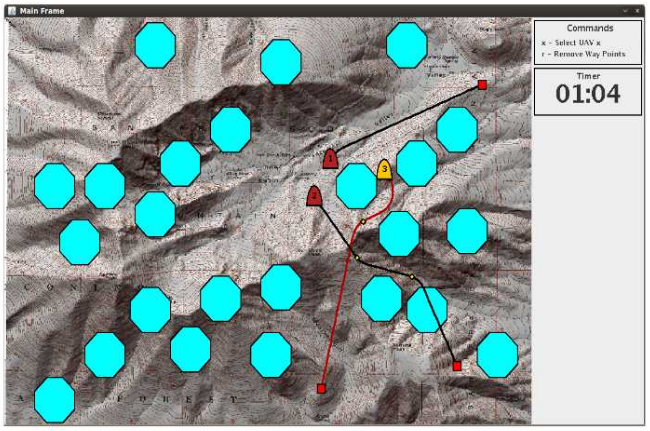


x – Select UAV x
r – Remove Way Points

Timer 01:04

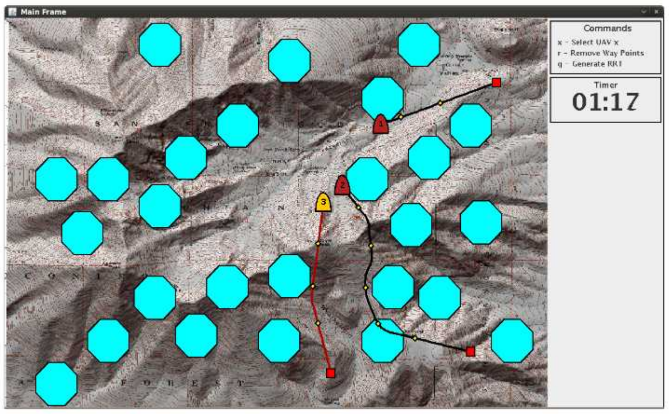
Human Only Mode

  - Move by holding left mouse button and dragging way point. Release when at desired location.
- Remove all waypoints for selected UAV by pressing the “r” key on keyboard



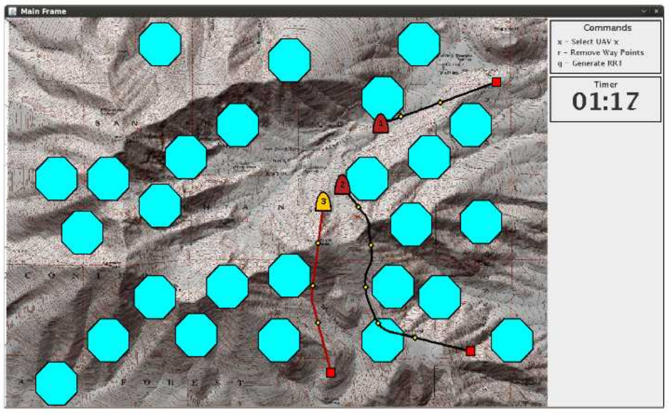
Human Guided RRT Mode 1

- Can only move and delete way points
- Can now ask RRT planning algorithm to generate solution for selected UAV by pressing the “g” key on the keyboard
- If RRT cannot find solution in a few seconds, the path will not change. Try again later or fix path manually.



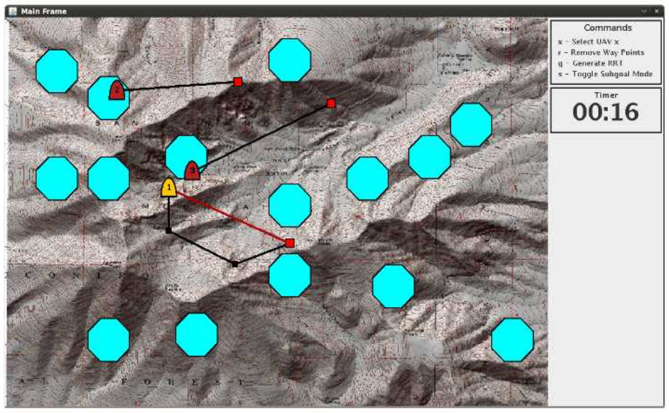
Human Guided RRT Mode 2

- Still have ability to move and delete way points, and use RRT algorithm by pressing the “g” key
- Waypoints appear only after calling for an RRT solution
- If another UAV is selected, the waypoints disappear



Human Constrained RRT Mode

- Waypoints appear and disappear the same way they do in the Human Guided RRT mode 2.
- Must now specify at least one subgoal for the RRT algorithm to include on the path to the goal.
- Subgoals meant to guide the algorithm to hopefully find a better solution faster

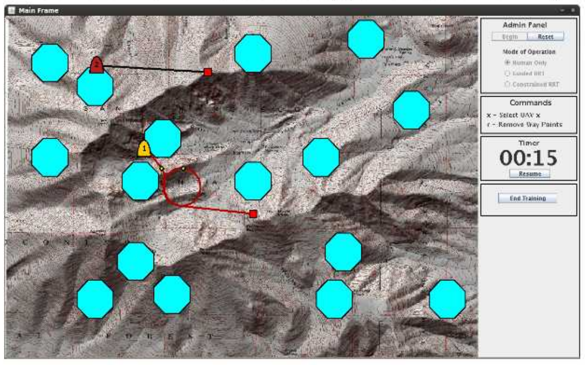


Human Constrained RRT Mode

- Toggle subgoal mode by pressing “s” on the keyboard
- When in subgoal mode, a black line will appear from the selected UAV to its next target
- To go back to changing waypoints, toggle out of subgoal mode
- Subgoals are added sequentially by left clicking w/ mouse
  - To insert a subgoal, hold the “ctrl” key
- Subgoals can be deleted and moved in the same manner waypoints are deleted and moved
- Once desired subgoals are set, ask RRT to generate solution with the “g” key
- **Must be in subgoal mode and set at least one subgoal for RRT algorithm to search for solution**
- After the RRT algorithm is called, subgoal mode is exited.

Training Interface

- Allows practice in all 4 modes with 2 UAVs
  - Select mode (in admin panel)
  - Click start button
  - Pause by pressing button on timer panel
  - Reset by pressing reset button on admin panel (can try another mode by repeating process)
- When done or 10 minutes are up, click end training button



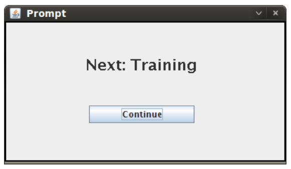
Mode of Operation
[✓ Human Only]
Guided RRT
Constrained RRT

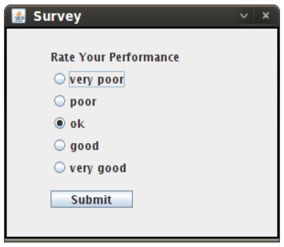
x - Select UAV x
r - Remove Way Points

00:15

Transitions

- Before every mission, will get a screen like the on shown below to the left denoting what mode of operation the next mission will be in
- After every mission (not including training) will get a survey window (below to the right). Select response, then click submit.
- After the electronic survey, you will be given a paper survey to fill out before proceeding to the next mission.





Questions??

If not then proceed to training

Appendix D

Post-Mission Survey

Feedback Survey

1. What did you like (if any) about the system (interface, UV behaviors, etc.)?

2. What did you NOT like (if any) about the system (interface, UV behaviors, etc.)?

3. What other capabilities would you like (if any) the system to have to assist you?

4. On the scale 1 to 5 (with 5 being the highest), how did you feel you understood what was happening in the game, what you were doing and what you needed to do next?

   1 2 3 4 5

   Comments (if any):

5. H*ow did you feel you performed?*

   Very Poor *Poor* *Satisfactory* *Good* *Excellent*

6. *How busy did you feel during the mission?*

   Extremely Busy *Busy* *Not Busy* *Idle*

7. Any other comments you might have

Appendix E

Supportive Statistics

This appendix contains tables showing the descriptive statistics for the data. The statistics are shown for the data when it is divided by the modes of operation, then the obstacle densities, then all of the data.

Table E.1: Descriptive Statistics: Average RRT Runtime (sec)

		mean	median	mode	st. dev.	max	min
Mode of Op.	HG2	.12512	.10665	.03395	.06822	.33123	.03395
(N=48)	HC	.11760	.10407	.04109	.05100	.25509	.04109
Obs. Density	HD	.15294	.14421	.07327	.05996	.33123	.07327
(N=48)	LD	.08978	.08548	.03395	.04064	.25510	.03395
Overall (N=96)		.12136	.10665	.03395	.06003	.33123	.03395

Table E.2: Descriptive Statistics: Average RRT Path Length Ratio

		mean	median	mode	st. dev.	max	min
Mode of Op.	HG2	1.3582	1.3539	1.1488	.1200	1.7081	1.1488
(N=48)	HC	1.4742	1.3223	1.0724	.5680	4.6688	1.0724
Obs. Density	HD	1.4595	1.3396	1.0724	.5574	4.6688	1.0724
(N=48)	LD	1.3728	1.3388	1.1073	.1711	2.0362	1.1074
Overall (N=96)		1.4162	1.3388	1.0724	.4125	4.6688	1.0724

Table E.3: Descriptive Statistics: Average Mission Time (sec)

		mean	median	mode	st. dev.	max	min
Mode of Operation (N=48)	HO	255.68	251.19	239.54	17.25	313.44	239.54
	HG1	257.17	255.09	240.94	12.54	302.24	240.94
	HG2	261.19	256.20	243.34	15.34	308.09	243.35
	HC	259.31	253.55	241.46	16.62	313.60	241.46
Obs. Density (N=96)	HD	263.15	257.52	239.55	17.93	313.60	239.55
	LD	253.53	251.03	239.54	10.91	305.91	239.54
Overall (N=192)		258.34	253.67	239.54	15.57	313.60	239.54

Table E.4: Descriptive Statistics: Number of Collisions

		mean	median	mode	st. dev.	max	min
Mode of Operation (N = 48)	HO	3.146	3	3	1.856	9	0
	HG1	2.583	2	2	1.582	8	0
	HG2	2.812	2	2	2.515	11	0
	HC	4.250	3	3	2.531	12	0
Obs. Density (N=96)	HD	3.365	3	2	2.616	12	0
	LD	3.031	3	3	1.780	8	0
Overall (N=192)		3.198	3	2	2.238	12	0

Table E.5: Descriptive Statistics: Number of Human Interactions

		mean	median	mode	st. dev.	max	min
Mode of Operation (N=48)	HO	41.708	34.5	22	23.009	133	15
	HG1	42.562	45.0	27	20.735	117	8
	HG2	33.708	32.0	15	18.007	73	3
	HC	37.896	35.0	35	15.566	79	11
Obs. Density (N=96)	HD	39.719	37.5	27	20.290	133	3
	LD	38.219	35.0	25	19.157	117	4
Overall (N=192)		38.969	35.5	25	19.694	133	3

Table E.6: Descriptive Statistics: Number of Replans

		mean	median	mode	st. dev.	max	min
Mode of Operation (N=48)	HG1	9.354	6.0	0	9.506	36	0
	HG2	21.917	20.5	15	8.274	40	8
	HC	14.208	14.0	14	4.740	27	6
Obs. Density (N=96)	HD	16.361	16.0	14	9.356	38	0
	LD	13.958	14.5	0	9.155	40	0
Overall (N=192)		15.160	15.0	0	9.302	40	0

Table E.7: Descriptive Statistics: Performance

		mean	median	mode	st. dev.	max	min
Mode of Operation (N=48)	HO	3.500	3	3	0.9225	5	2
	HG1	3.583	4	3	0.7390	5	2
	HG2	3.562	3	3	0.8482	5	2
	HC	3.021	3	3	0.7852	5	2
Obs. Density (N=96)	HD	3.458	3	3	0.8574	5	2
	LD	3.375	3	3	0.8491	5	2
Overall (N=192)		3.417	3	3	0.8521	5	2

Table E.8: Descriptive Statistics: Frustration

		mean	median	mode	st. dev.	max	min
Mode of Operation (N=48)	HO	2.292	2.0	3	0.8241	4	1
	HG1	2.438	3.0	3	0.9204	4	1
	HG2	2.500	2.5	2	0.7718	4	1
	HC	2.917	3.0	3	1.0071	5	1
Obs. Density (N=96)	HD	2.406	2.0	2	0.9578	5	1
	LD	2.667	3.0	3	0.8419	4	1
Overall (N=192)		2.536	3.0	3	0.9088	5	1

Table E.9: Descriptive Statistics: Workload

		mean	median	mode	st. dev.	max	min
Mode of Operation (N=48)	HO	2.395	2	2	0.5739	3	1
	HG1	2.583	3	3	0.6790	4	1
	HG2	2.500	2	2	0.5458	4	2
	HC	2.938	3	3	0.5614	4	2
Obs. Density (N=96)	HD	2.635	3	3	0.6508	4	1
	LD	2.573	3	3	0.5937	4	1
Overall (N=192)		2.604	3	3	0.6221	4	1

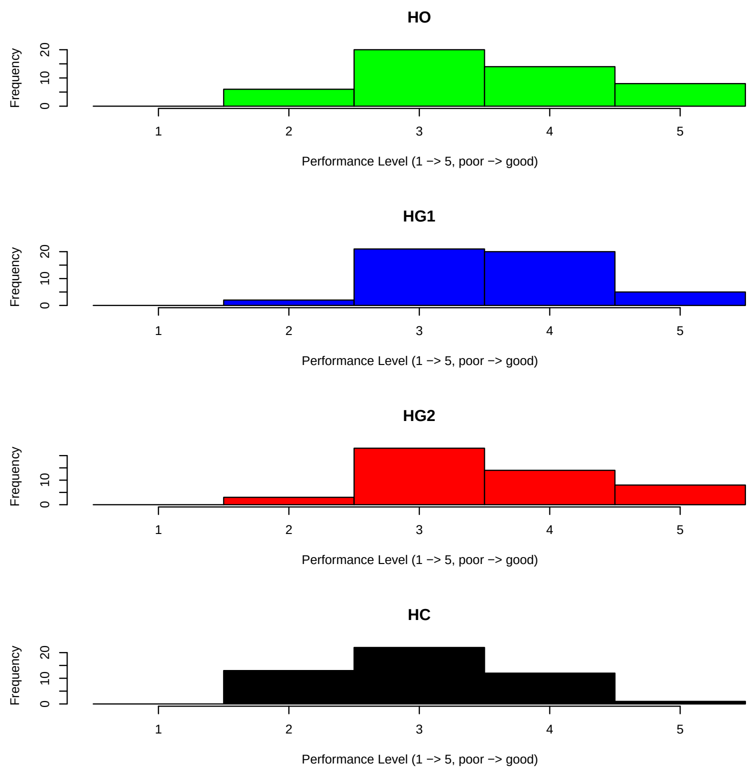


Figure E-1: Histogram - Performance

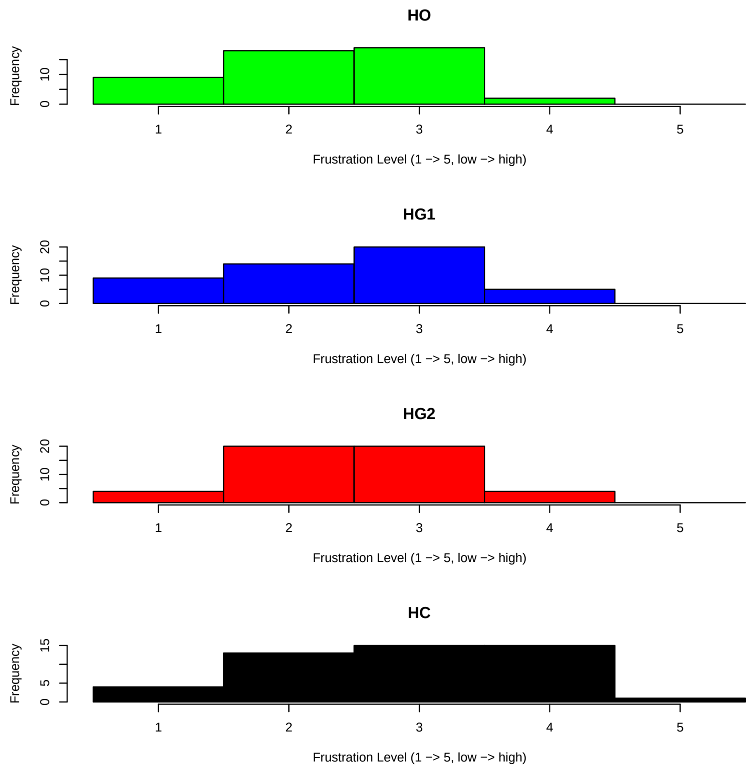


Figure E-2: Histogram - Frustration

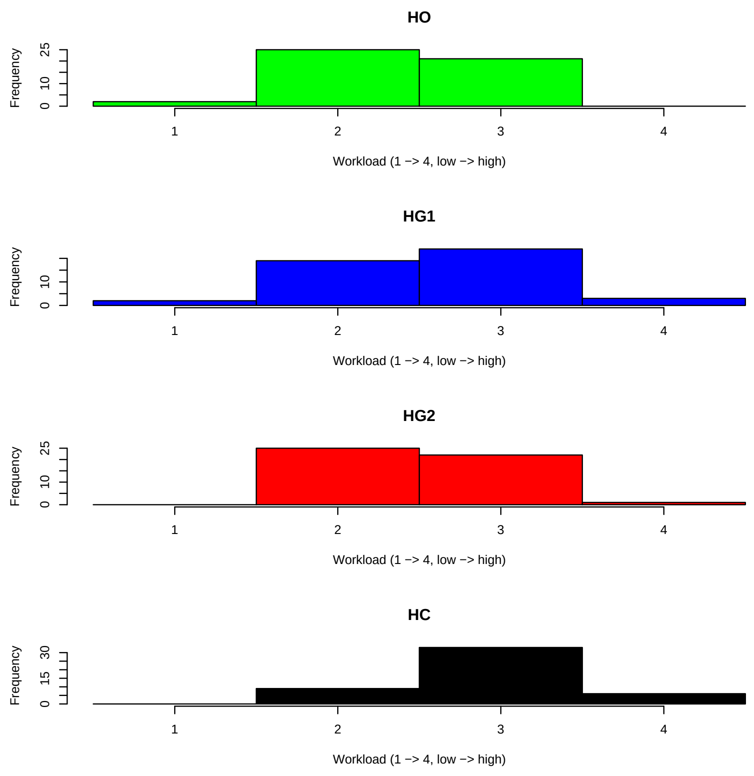


Figure E-3: Histogram - Workload

Bibliography

[1] M. Cummings, S. Bruni, S. Mercier, and P. Mitchell, "Automation architecture for single operator, multiple UAV command and control," *International Command and Control Journal*, vol. 1, no. 2, pp. 1–24, 2007.

[2] Simba Aviation (2009), *Unmanned Aerial-System [Online].* Available: http://www.unmanned-aerial-system.com/.

[3] G. Sirigineedi, A. Tsourdos, R. Zbikowski, and B. White, "Modeling and verification of multiple UAV mission using SMV," *Electronic Proceedings in Theoretical Computer Science*, vol. 20, pp. 22–33, 2010.

[4] Y. Yang, M. M. Polycarpou, and A. A. Minai, "Multi-UAV cooperative search using an opportunistic learning method," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 716–728, 2007.

[5] Y. Yang, A. Minai, and M. Polycarpou, "Cooperative real-time search and task allocation in UAV teams," in *Proceeding of IEEE conference on Decision and Control*, (Maui, Hawaii, USA), pp. 7–12, Dec 2003.

[6] J. C. Russo, M. Amduka, B. Gelfand, K. Pedersen, R. Lethin, J. Springer, R. Manohar, and R. Melhem, "Enabling cognitive architectures for UAV mission planning," in *Proceedings of the Tenth Annual High Performance Embedded Computing Workshop*, (Cambridge, Massachusetts, USA), Sept 2006.

[7] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms.* MIT Press and McGraw Hill, 1990.

[8] J. Reif and H. Wang, "Non-uniform discretization approximations for kinodynamic motion planning and its applications," *SIAM Journal on Computing*, vol. 30, pp. 161–190, 2000.

[9] J. Reif, "Complexity of the mover's problem and generalizations," in *Proceedings from IEEE Symposium on Foundations of Computer Science*, (San Juan, Puerto Rico, USA), pp. 421–427, Oct 1979.

[10] S. Lazard, J. Reif, and H. Wang, "The complexity of the two dimensional curvature-constrained shortest-path problem," in *Proceedings of the Third International Workshop on the Algorithmic Foundations of Robotics*, (Houston, Texas, USA), pp. 49–57, March 1998.

[11] J. Reif and M. Sharir, "Motion planning in the presence of moving obstacles," *Journal of the Association for Computing Machinery*, vol. 41, no. 4, pp. 764–790, 1994.

[12] A. Richards, T. Shouwenaars, J. How, and E. Feron, "Spacecraft trajectory planning with avoidance constraints using mixed integer linear programming," *AIAA Journal of Guidance, Control and Dynamics*, vol. 25, no. 4, pp. 755–764, 2002.

[13] D. Hsu, R. Kindel, J. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–235, 2002.

[14] D. Hsu, J. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings from IEEE International Conference on Robotics and Automation*, (Albuquerque, New Mexico, USA), pp. 2719–2726, April 1997.

[15] S. LaValle and J. Kuffner, J.J., "Randomized kinodynamic planning," in *Proceedings from IEEE International Conference on Robotics and Automation*, vol. 1, (Detroit, Michigan, USA), pp. 473–479, May 1999.

[16] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep. TR: 98-11, Computer Science Department, Iowa State University, Oct 1998.

[17] S. LaValle, *Planning Algorithms.* Cambridge University Press, 2006.

[18] Y. Kuwata, G. Fiore, J. Teo, E. Frazzoli, and J. How, "Motion planning for urban driving using RRT," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Nice, France), pp. 1681–1686, Sept 2008.

[19] D. Ferguson and A. Stentz, "Anytime RRTs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Beijing, China), pp. 5369–5375, Oct 2006.

[20] J. Lee, C. Pippin, and T. Balch, "Cost based planning with RRT in outdoor environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Nice, France), pp. 684–689, Sept 2008.

[21] A. Yershova, L. Jaillet, T. Simeon, and S. LaValle, "Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, (Barcelona, Spain), pp. 3856–3861, April 2005.

[22] M. Strandberg, "Augmenting RRT-planners with local trees," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, vol. 4, (New Orleans, Louisiana, USA), pp. 3258–3262, May 2004.

[23] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite RRTs for rapid replanning in dynamic environments," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, (Roma, Italy), pp. 1603–1609, April 2007.

[24] A. Alves Neto, D. Macharet, and M. Campos, "On the generation of trajectories for multiple UAVs in environments with obstacles," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1-4, pp. 123–141, 2010.

[25] K. Yang and S. Sukkarieh, "Real-time continuous curvature path planning of UAVs in cluttered environments," in *Proceedings of the 5th International Symposium on Mechatronics and Its Applications*, (Amman, Jordan), pp. 1–6, May 2008.

[26] A. P.-B. Issco and A. Popescu-belis, “An experiment in comparative evaluation: Humans vs. computers,” in *Proceedings of Machine Translation Summit IX*, (New Orleans, Louisiana, USA), pp. 307–314, Sept 2003.

[27] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski, “Building segmentation based human-friendly human interaction proofs (HIPs),” in *Proceedings of the Second International Workshop on Human Interactive Proofs*, (Bethlehem, Pennsylvania, USA), May 2005.

[28] M. L. Cummings and S. Bruni, “Collaborative human-automation decision making,” in *Handbook of Automation* (S. Y. Nof, ed.), pp. 437–447, Springer, 2009.

[29] V. J. Lumelsky and A. A. Stepanov, “Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape,” in *Autonomous robot vehicles*, pp. 363–390, Springer-Verlag New York, Inc., 1990.

[30] A. Datta and K. Krithivasan, “Path planning with local information,” in *Proceedings of the Conference on the Foundations of Software Technology in Theoretical Computer Science*, vol. 338, (Pune, India), pp. 108–121, Dec 1988.

[31] M. Sipser, *Introduction to the Theory of Computation.* International Thomson Publishing, 1996.

[32] K. G. Shin and N. D. McKay, “Robot path planning using dynamic programming,” in *The 23rd IEEE Conference on Decision and Control*, vol. 23, (Las Vegas, Nevada, USA), pp. 1629–1635, Dec 1984.

[33] M. Peot, T. Altshuler, A. Breiholz, R. Bueker, K. Fertig, A. Hawkins, and S. Reddy, “Planning sensing actions for uavs in urban domains,” in *The International Society of Optical Engineering (SPIE) Unmanned/Unattended Sensors and Sensor Networks II*, (Bruges, Belgium), Sept 2005.

[34] M. Alighanbari, Y. Kuwata, and J. How, “Coordination and control of multiple uavs with timing constraints and loitering,” in *Proceedings of the 2003 American Control Conference*, vol. 6, (Denver, Colorado, USA), pp. 5311–5316, June 2003.

[35] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[36] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of A*," *Journal of the Association for Computing Machinery (ACM)*, vol. 32, no. 3, pp. 505–536, 1985.

[37] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime search in dynamic graphs," *Artificial Intelligence*, vol. 172, pp. 1613–1643, 2008.

[38] G. Nemhouser and L. Wolsey, *Integer and Combinatorial Optimization.* New York: John Wiley and Sons, 1988.

[39] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco: Freeman, 1972.

[40] Y. Kuwata and J. How, "Receding horizon implementation of MILP for vehicle guidance," in *Proceedings of the 2005 American Control Conference*, vol. 4, (Denver, Colorado, USA), June 2005.

[41] J. Bellingham, A. Richards, and J. P. How, "Receding horizon control of autonomous aerial vehicles," in *Proceedings of the American Control Conference*, (Anchorage, Alaska, USA), pp. 3741–3746, May 2002.

[42] T. Schouwenaars, J. How, and E. Feron, "Receding horizon path planning with implicit safety guarantees," in *Proceedings of the 2004 American Control Conference*, vol. 6, (Boston, Massachusetts, USA), pp. 5576–5581, June 2004.

[43] N. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *IEEE International Conference on Robotics and Automation*, (Minneapolis, Minnesota, USA), pp. 113–120, May 1996.

[44] L. Kavraki, F. Svetska, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transaction of Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[45] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *Proceedings of the International Conference on Robotics and Automation*, (Kobe, Japan), May 2009.

[46] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[47] M. Shanmugavel, A. Tsourdos, B. White, and R. Zbikowski, "Co-operative path planning of multiple UAVs using Dubins paths with clothoid arcs," *Control Engineering Practice*, 2009.

[48] C. Froissart and P. Mechler, "Online polynomial path planning in cartesian space for robot manipulators," *Robotica*, vol. 11, pp. 245–251, 1993.

[49] P. Gallina and A. Gasparetto, "A technique to analytically formulate and to solve the 2-D constrained trajectory planning problem for a mobile robot," *Journal of Intelligent and Robotic Systems*, vol. 27, pp. 237–262, 2000.

[50] H. Chitsaz and S. LaValle, "Time-optimal paths for a dubins airplane," in *Proceedings of 46th IEEE Conference on Decision and Control*, pp. 2379–2384, Dec 2007.

[51] T. Sheridan, *Telerobotics, Automation, and Human Supervisory Control.* Cambridge, MA: MIT Press, 1992.

[52] M. Cummings and P. Mitchell, "Predicting controller capacity in remote supervision of multiple unmanned vehicles," *IEEE Systems, Man, and Cybernetics, Part A Systems and Humans*, vol. 38, no. 2, pp. 451–460, 2008.

[53] C. Nehme, J. Crandall, and M. Cummings, "An operator function taxonomy for unmanned aerial vehicle missions," in *12th International Command and Control Research and Technology Symposium*, (Newport, Rhode Island, USA), June 2007.

[54] J.-H. Hwang, R. Arkin, and D.-S. Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control," in *Proceedings of the*

IEEE/ RSJ International Conference on Intelligent Robots and Systems, (Las Vegas, Nevada, USA), Oct 2003.

[55] G. Carrigan, “The design of an intelligent decision support tool for submarine commander,” Master’s thesis, Massachusetts Institute of Technology Engineering Systems Division, June 2009.

[56] M. Cummings, M. Buchin, G. Carrigan, and B. Donmez, “Supporting intelligent and trustworthy maritime path planning decisions,” *International Journal of Human Computer Studies.* in press.

[57] J. Marquez, *Human-Automation Collaboration: Decision Support for Lunar and Planetary Exploration.* PhD thesis, Massachusetts Institute of Technology Department of Aeronautics and Astronautics, June 2007.

[58] H. Kautz and J. Allen, “Generalized plan recognition,” in *Proceedings of 5th National Conference on Artificial Intelligence*, (Philadelphia, Pennsylvania, USA), pp. 32–37, Aug 1986.

[59] N. Lesh, C. Rich, and C. Sidner, “Using plan recognition in human-computer collaboration,” in *Proceedings of the 7th International Conference on User Modeling*, (Banff, Canada), pp. 23–32, June 1999.

[60] R Development Core Team (2008), *R: A Language and Environment for Statistical Computing [Online].* Available: http://www.R-project.org.