Human-RRT Collaboration in UAV Mission Path Planning

by

Alina Griner

S.B. in Mathematics,
Massachusetts Institute of Technology (2010)
S.B. in Computer Science and Electrical Engineering,
Massachusetts Institute of Technology (2012)

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 2012

© Massachusetts Institute of Technology 2012. All rights reserved.

Author . . . .
Department of Electrical Engineering and Computer Science
May 21, 2012

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

Accepted by . . . .
Dennis M. Freeman
Professor of Electrical Engineering and Computer Science
Chairman, Masters of Engineering Thesis Committee

Human-RRT Collaboration in UAV Mission Path Planning

by

Alina Griner

Submitted to the
Department of Electrical Engineering and Computer Science

May 21, 2012

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

Abstract

Unmanned Aerial Vehicles (UAVs) are used for a variety of military and commercial purposes, including surveillance, combat, and search and rescue. Current research is looking into combining automation with human supervision to facilitate various lower-level cognitive tasks, such as path planning, in order to allow the human operator to focus on high-level mission strategy. Path planning in the presence of dynamic constraints necessitates extensive real-time replanning, which is a computationally intensive task, especially when there is a high density of obstacles or no-fly zones. Recently common choices of path finding algorithms have used variations of a randomized algorithm called Rapidly exploring Random Tree (RRT). This randomized sampling algorithm finds fairly short feasible paths, and it finds them efficiently, however human operators supervising UAV missions may have difficulty collaborating with a randomized algorithm. This thesis presents the experimental results of the second round in an iterative interface design project analyzing human collaboration with a RRT algorithm. In the experiment, operators completed simulated UAV missions in three modes with varying methods of interaction with the RRT algorithm. The collected data included performance and behavioral metrics, as well as subjective feedback. The results demonstrated a clear operator preference for certain methods of interaction with RRT over others. The mode of interaction that allowed these preferred methods had similar results in most metrics to the manual planning mode; the mode with the least preferred methods had significantly worse results. The outcome of the experiment did not conclusively answer the question of whether using RRT for path planning results in lower mission completion times or lower cognitive load on the operator, however the analysis of the data and observations of operator behavior lead to questions for future research.

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

Acknowledgements

I would like to thank my thesis supervisor, Missy Cummings, for offering me this project, guiding me along the way, and answering all my questions. My time in the Humans and Automation Lab has been a great experience – I met a lot of interesting people, had fun playing squash and going out for drinks, and learned about doing research in human-computer interaction. Thank you to everyone in the lab, and good luck with your future projects.

To Luca Bertuccelli, thank you for all your help with this thesis – for discussing designs, and checking to make sure I finished things on time, and reading my thesis draft, and having a good sense of humor. Thank you to Jamie Macbeth, for offering a lot of helpful advice and not getting angry at me for missing deadlines. This process would have been a lot harder if not for you two.

Thank you to Andrew for taking care of COUHES (human research) paperwork and discussing interface design ideas. Thank you to Christine for giving me an overview of Americo's code, and thank you to Americo for writing good, clear, modular code that was very easy to modify and add on to. I would also like to thank everyone who participated in my experiment, including pilot testers, who gave me feedback that helped improve my interface.

Finally, I would like to thank my family and friends for all their warmth and support. I am very grateful for having you, and I would not be where I am today if not for you.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

UAVs (Unmanned Aerial Vehicles) are unmanned aircraft that are used for a variety of different purposes, such as surveillance, combat, and search and rescue. The presence of UAVs is becoming more and more prominent in the military. Since 2005, the number of combat air patrols performed by UAVs has increased by 1200% [1]. UAVs are also used by civil and commercial organizations for applications such as fighting fires and fertilizing crops. Historically, UAVs have been either remotely piloted or flown semi-autonomously following pre-programmed paths and flight profiles. However, currently UAV missions are becoming increasingly more complicated, as future missions will involve multiple UAVs in dangerous and continually changing environments. To this end, a single operator will need to guide multiple UAVs involved in such a mission, but the dynamic environment makes pre-programming flight plans insufficient as unforeseen obstacles may come up and require changing the routes. One solution that could enable this capability is cooperation between a human operator and path planning algorithms. This thesis presents the experimental results of the second round in an iterative interface design project analyzing human cooperation with a Rapidly exploring Random Tree (RRT) algorithm.

1.1 Background

Unmanned aerial vehicles are being used for increasingly diverse and complicated missions. To perform these missions, the UAVs are equipped with increasingly advanced technology, but there is always a human in the loop in the UAV control system.

Nehme, Crandall, and Cummings [2] developed a taxonomy of UAV missions, dividing both military and commercial mission types into seven categories, and these further into sub-categories. The taxonomy includes both military missions and commercial missions, and the categories and subcategories are not mutually exclusive; a UAV mission can involve several different functions. The high-level mission categories are the following:

Intelligence/Reconnaissance: provide military intelligence, for example the mapping of an area, target designations, or information on battle damage and assessment.

Drones: used as decoys and as aerial gunnery units during combat.

Transport: can be used for both cargo and passengers.

Extraction: payload extraction from a target. Search and rescue is in this category.

Insertion: examples include weapons deliveries in the military, and crop dusting in commercial use.

Communication: the UAVs may need to communicate among themselves as well as with the central command.

Surveillance: used to monitor people, objects, or processes.

A UAV mission may fall into several of the above categories. For example, in a search and rescue mission, the UAV can be used to map out a damaged building, and if several UAVs are involved in the search and rescue mission, they may need to communicate between themselves to make sure they perform the task as efficiently as possible.

Nehme, Crandall, and Cummings listed functional and informational requirements for the different missions, and noted that path planning supervision was an important factor in nearly all the categories and sub-categories. They further argued that for a single operator to control multiple UAVs, the use of automation is required to help with navigation, because if operators devote too much time and cognitive capacity to path planning, they do not perform effectively in higher-level mission management.

1.1.1 Path Planning

The variety of roles that UAVs play and the complexity of the missions make path planning in missions a hard problem. The goal of the path finding problem is essentially to find the minimum-cost collision-free path from the UAV to its target, or to a sequence of targets. Many missions take place in dangerous environments that are continuously changing. Obstacles may move or new obstacles may appear, areas that were previously considered safe become no-fly zones, or perhaps new information is learned that requires a UAV to add a new waypoint to its route or change its destination. Paths may be planned for UAVs at the start of the mission, but due to frequent changes in the environment, the path planning system should allow quick replanning. If automation is used to plan paths, there is the question of which algorithm to use. In most cases, the closer a path is to the optimal, the longer it takes for an algorithm to find it, and in UAV missions results need to be generated efficiently enough for real-time planning. Constraints on the vehicles make the problem even harder. UAVs cannot make sharp turns in the air because of kinodynamic constraints, thus a path planning algorithm would need to take these constraints into consideration when creating paths so that the paths are physically possible for the UAVs to follow.

Optimal path planning in the presence of dynamic constraints requires extensive replanning. This is computationally intensive, and so often a *feasible* plan is acceptable because finding the optimal plan requires too much time and paths are frequently modified anyway.

Optimal planning in dynamic environments is a difficult combinatorial optimization problem, so usually suboptimal methods are used. Randomized algorithms are examples of such methods. Recent common choices of path finding algorithms have used variations of a randomized algorithm called Rapidly exploring Random Tree (RRT). The RRT algorithm is a type of Rapidly exploring Dense Tree (RDT) algorithm [3]. The idea behind these algorithms is to build a search tree from the start location by repeatedly sampling the search space and extending the current tree to include the sampled point. The tree is extended by connecting the sampled point to the nearest neighboring point in the tree. When the tree contains the goal location, the algorithm terminates. In an RRT algorithm, the sequence of samples used in the construction of the tree is random. RRTs work well with UAV path planning applications because it is straightforward to modify the algorithms to consider the kinodynamic constraints of the vehicles. The algorithm finds fairly short (though not optimal) paths [4], and it finds them quickly, enabling efficient real-time path planning. The randomized property of the produced paths is an advantage against opponents trying to track the UAVs; however it could be difficult for the operator supervising the UAVs to accept the unpredictable nature of the paths.

1.1.2 Human Supervision

UAV missions are too complicated to fully automate. There must always be a human supervisor in the loop in UAV control systems to have the final say on various tasks such as assigning and prioritizing UAV targets, firing a weapon, monitoring information from the UAVs, and designing the high-level strategy of the mission. These tasks cannot be entrusted to automation either because they are decisions that vary depending on the situation and should be made by humans, such as firing a weapon during combat, or because they are hard to encode using algorithms.

The human operator cannot be in charge of an entire mission involving multiple UAVs manually, however, because then the cognitive load on the operators is too high for them to perform well consistently. The use of automation can reduce the cognitive load on the operator by helping with some of the tasks, such as path planning. A lot of current research is trying to find ways of combining automation and human control for the most effective performance [5]. This problem involves finding algorithms that humans can trust, and developing usable methods of interaction with these algorithms for human operators.

For path planning, humans and computer algorithms each have their strengths in solving this problem. Humans are good at solving perceptual-based problems, while for computers these problems are very hard. For example, it is difficult for path planning algorithms to find paths that go through a narrow passage in between obstacles. For humans this is very easy. However, it is hard for humans to predict what kind of paths will be possible for UAVs to follow due to kinodynamic constraints, and what paths will be impossible. This is easy to encode using algorithms.

1.2 Research Questions

This thesis investigates the following research questions:

Research Question #1: Does collaborating with an RRT algorithm in path planning decrease the cognitive load of the operator over manual path planning?

There has been little research so far that shows how using RRTs in path planning impacts performance over human manual planning. Caves conducted an experiment with this purpose but had inconclusive results [6]. RRTs find solutions fairly quickly, and they generate entire paths, meaning that operators may end up doing less work and thus having a lower cognitive load. In addition they can take into consideration the kinodynamic constraints of the vehicles, taking that responsibility away from the operator. However, the reason that RRTs work quickly is due to randomization, and therefore the algorithm may output unexpected paths that frustrate the operators and lead them to spend more time trying to correct the generated solutions than would take them to generate their own, which would increase their cognitive load.

Research Question #2: Does a mode involving automation result in better performance, where performance is measured by mission completion time and obstacle avoidance, than the manual mode? Which mode involving automation performs better?

The interface has three modes of operation: one manual mode and two modes that involve automation. The modes involving automation allow the operator to interact with RRT in different ways. In one of the modes, the operator can influence the algorithm's solutions by indicating general areas that the paths should go through, but the operator cannot alter generated paths. In the other mode the algorithm suggests paths and the operator can then modify them manually. Allowing operator input into the algorithm may improve performance from pure automation because the human in the loop can set constraints that create shorter problems for the algorithm and speed up the search, and it may improve performance from manual path planning because the operator does not have to find the entire path. Allowing operator input after solutions have been proposed may improve performance from pure automation because the human in the loop can modify paths to make them more efficient, and it may improve performance from manual planning because the automation suggests feasible paths that avoid obstacles.

Research Question #3: How do human operators prefer to interact with RRTs to plan paths? How does using RRTs affect operators' perception of their performance?

Apart from objective performance metrics, it is important to understand how human operators subjectively feel about collaborating with an RRT algorithm, and how they prefer to interact with it. For successful human-automation collaboration, the human operator must feel comfortable with the automation.

Research Question #4: How does RRT path planning impact performance and subjective assessment as the number of UAVs increases?

Finally, it is important to understand how the effect of using an RRT changes with different numbers of UAVs. For example, it is possible that with only a few vehicles human operators prefer to have manual control but as the number of UAVs increases operators find it more efficient to automate paths. If, on the other hand, using RRTs brings frustration, operators may prefer to use RRTs with small numbers of UAVs but not with many.

The objective of this thesis was to investigate these questions through an experiment that analyzes human-RRT interaction in UAV path planning, and as a result develop a better understanding of the collaboration between a randomized path planner and a human in the loop. The remaining chapters of this thesis are outlined below:

- Chapter 2 describes relevant background information and previous work in detail to provide more context for this research.

- Chapter 3 describes the experiment, including the interface, the planning algorithm, and the experimental protocol.

- Chapter 4 answers the research questions of the experiment.

- Chapter 5 presents additional experimental results and observations.

- Chapter 6 gives a brief summary of this thesis and discusses future research directions.

Chapter 2: Background and Previous Work

2.1 Path Planning

There are many variations of the path planning problem. Sometimes the goal is merely to find a *feasible* path, which means any path from the starting location and initial conditions, together called a start state, to the goal location and conditions, together called a goal state. Other times the goal is to find the *optimal* path, the path that optimizes some criterion – usually either time, distance, or energy. Solving the problem in discrete environments is different from solving it in continuous spaces.

In order to plan paths for an environment, the environment has to be accurately represented. Environments are usually modeled as discrete spaces or as continuous spaces. Continuous spaces are defined in $\mathbb{R}^n$, where $\mathbb{R}$ is the set of real numbers and n is the number of dimensions in the space. Any real coordinates are possible in this representation. Discrete representations model the space as a lattice of acceptable coordinates. An example would be an environment where only integer coordinates are possible. It is easier to find optimal paths in discrete environments but harder to ensure that those paths are physically feasible for UAVs, because the transition between coordinates is difficult to model in discrete spaces, as there are no intermediate states.

2.1.1 Discrete Path Planning

One approach to solving path planning problems is to discretize the search space. In discrete environments, each distinct situation is called a *state*, and the set of all possible states is called a *state-space*. A state-space may be infinite, though usually it will be finite, but for discrete problems it must be countable. For a given state x, there is a set of actions that can be performed on x and which result in a new state x' according to a *state transition function*, f. In other words, for a possible action u, $x' = f(x,u)$. The problem defines a start state and a goal state or set of goal states. The objective is to find a finite sequence of actions that lead from the start state to a goal state. This sequence of actions is a feasible plan.

One example of a problem that can be described in this way is a Rubik's Cube. Here the state could be described by the colored squares and their positions, an action could be an allowable rotation of the cube, and the new state would result from the rotation. The goal state would be the state where each side of the cube has only one color on it, and the solution would be the sequence of rotations to achieve the goal state from the initial configuration of the cube. Another example of a discrete feasible search problem would be the exploration of a maze. The state would be the coordinates of the current cell; possible actions are stepping into an adjacent cell given that it does not have any obstacles; and the goal state would be the exit of the maze. The solution would be the sequence of steps to get there from the initial position.

Path planning problems in discrete spaces can be formalized using graph theory. In graph problems, there are given a set of vertices and a set of edges, where an edge connects two vertices *u* and *v*. The graph can be directed, which means that an edge (*u*,*v*) can be traversed in only one direction, *u* to *v*, and not in the opposite direction – *v* to *u*, or undirected, where an edge can be traversed in either direction. The graph can be weighted, where each edge has a cost *w* associated with it, or unweighted, where all the edges have the same cost. For a given node, the possible actions are to traverse an edge leaving that node to one of the neighboring nodes. The search problem is to get from a given start node *s* to a given end node *t*. A feasible solution for this problem is a sequence of edges that can be followed from *s* to reach *t*. The optimal solution to this problem is the sequence of edges with the lowest possible total cost out of all feasible solutions. For unweighted graphs, the optimal path has the smallest number of edges. The next few sections describe algorithms typically used to solve various path finding problems in graph theory.

Breadth First Search and Depth First Search

If the goal of the path finding problem is to simply find a path, a solution can be provided by breadth first search or depth first search. Breadth first search and depth first search are two similar algorithms that find feasible paths in graphs. They both keep a queue of unvisited nodes that they gradually explore and add on to. Each algorithm initializes with only the start state in the queue. Then, until it reaches the goal state, it continues to execute the following steps: it extracts the first node in the queue, checks if it is the goal state, marks it as visited, and adds the node's unvisited neighbors to the queue. The algorithms differ in the way that nodes are added to the queue. Breadth first search adds the unvisited neighbors to the end of the queue in a First In First Out (FIFO) manner ; depth first search adds them to the beginning of the queue in a Last In First Out (LIFO) manner. Breadth first search explores all nodes k steps away before it explores any node k+1 steps away. As a result, it finds the path with the fewest transitions. If all edges are weighted equally, this is the optimal path. Depth first search chooses a direction from the start and explores it exhaustively before trying a different direction. Breadth first search works quickly if the goal state is relatively close to the start state, such as in a Rubic's Cube. Depth first search works well if the solution path is longer than most paths, such as when exploring a maze. Both algorithms have a running time of O(|V| + |E|), where V is the set of vertices or states in the state-space, and E is the set of edges or actions.

Dijkstra's Algorithm

A common problem in graph theory is the single-source shortest paths problem; the task is to find the shortest paths from a given start node *s* to all the nodes in the graph. Dijkstra's algorithm solves the single-source shortest paths problem on a weighted, directed graph with non-negative edge weights. The algorithm keeps a set S of vertices for which the shortest path distance has already been found, and a queue of vertices for which it has not. It extracts from the queue the vertex that has the shortest current path length from the start node and relaxes the outgoing edges from that vertex. It then puts that vertex into S, because its current path is necessarily a shortest path to it. The process of relaxing an edge (*u*,*v*) checks to see if going along that edge would result in a shorter path to vertex *v* than is currently known. In the code below, `d[x]` keeps track of the shortest distance to node *x* and `π[x]` holds a pointer to the previous node along the shortest known path to node x.

```
RELAX(u, v, w)
   if d[v] > d[u] + w(u, v)
      then d[v] ← d[u] + w(u, v)
           π[v] ← u
```

The running time of Dijkstra depends on the implementation of the queue. Its fastest time is achieved with the use of a Fibonacci heap and is O(|E| + |V| log |V|). Dijkstra's algorithm finds optimal paths from the start node to every node in the graph if it is allowed to finish. However, if only a feasible path is desired, the algorithm can be terminated as soon as `d[t]`, where *t* is the goal node, becomes less than infinity. The algorithm can be sped up by using bidirectional search – running Dijkstra's from both the start node and the goal node in parallel, and terminating when the two trees meet. Bidirectional search can greatly reduce the amount of exploration required [7].

A* (A-star) Algorithm

The A* algorithm is an extension of Dijkstra's algorithm [8]. It tries to reduce the number of states searched by adding a heuristic potential function that estimates the distance from each node to the goal node. It runs exactly like Dijkstra's, but instead of sorting the queue based on distances from the start node, it sorts it based on the distance from the start node plus the estimated distance to the goal node. Equivalently, the edges of the graph can be reweighted using the heuristic potential function λ: $w^*(u,v) = w(u,v) - \lambda(u) + \lambda(v)$. Then Dijkstra's algorithm can be run with the new edges w*. The purpose of the heuristic potential function is to make nodes closer to the end goal more "attractive" so that the algorithm follows them earlier in the process and finds the shortest path faster. An example of a common heuristic function used is straight-line distance. It works well if for example the graph is representing a geographic area or a maze. If a directed edge (u,v) is heading in the direction of the goal, then the potential function for v will be smaller than the potential for u, so $w^*(u,v)$ will be smaller than $w(u,v)$, making the edge more attractive. If, on the other hand, the edge is heading away from the goal, then $\lambda(v)$ will be larger than $\lambda(u)$, and thus $w^*(u,v)$ will be larger than $w(u,v)$, making the edge less attractive. The requirement for the potential function is that it is *feasible*: for every edge (u,v) in the graph, $w^*(u,v) = w(u,v) - \lambda(u) + \lambda(v) \geq 0$. Dijkstra's cannot be used with negative edges, so the reweighted edges must still have non-negative weight. In discrete spaces, A* is one of the most commonly used algorithms [9].

Discrete path finding algorithms work well for small search spaces, such as the Rubik's Cube, or in environments where the objects' motion along the paths does not need to follow physical constraints, such as in video games. However, for the problem of UAV path planning, discrete algorithms are not optimal for two reasons: large computation times and unsatisfied vehicle constraints. First, in order to approximate continuous paths well, the discretization of the search space must be very fine. However, the finer the discretization, the more states must be searched, and so computation time increases. For the environments involved in UAV missions, discrete algorithms that provide good approximations of continuous paths usually take so much time that they are impractical. Second, it is difficult to modify discrete algorithms to take vehicle constraints into consideration. Discrete paths are represented by sequences of adjacent states; however in the real world, there are infinitely many in-between states to account for. For example, a UAV cannot change its heading instantaneously; the path finding algorithm must account for a turning radius, and the turning radius may depend on other factors such as speed. In order to find physically feasible paths in discrete spaces, additional information such as heading must be included in the states, but this greatly complicates the search space and increases computation time to the point of impracticality.

2.1.2 Continuous Path Planning

Continuous planning, often referred to as motion planning [3], does not have the same limitations as discrete planning and therefore is a better choice for UAV mission path planning. The problem is to find the motions required for the robot, human, vehicle, or other moving entity to get from the starting location to the end goal without colliding with obstacles in a 2D or 3D world. There are several different approaches to motion planning, including combinatorial motion planning and sampling-based motion planning.

Combinatorial Motion Planning

Combinatorial motion planning algorithms are generally *complete*, which means that the algorithms always find solutions if they exist or report that there is no solution when they do not exist in a finite amount of time [3]. Combinatorial algorithms are often impractical: either they have very high asymptotic running times or they are too complicated to implement. Most combinatorial algorithms construct a roadmap of the continuous space. A roadmap is a topological graph with paths that reflect the connectivity of the space. It should be reachable from both the start state and the end state. Combinatorial algorithms generally consist of building the roadmap and subsequently searching it for a solution.

Sampling-Based Motion Planning

Unlike combinatorial motion planning algorithms, sampling-based algorithms are not complete; they do not always find a solution if a solution exists [3]. Weaker notions of completeness are used with sampling-based algorithms. If an algorithm samples *densely*, this means that as the number of iterations approaches infinity, the samples come arbitrarily close to any possible sequence. If this is the case, the algorithm is said to be *resolution complete*, which means that if a solution exists, it will find it in a finite amount of time, however if no solution exists the algorithm may never stop running. The algorithm used for this thesis performs random sampling and is therefore *probabilistically complete*, which means that as the number of samples approaches infinity, the probability that the algorithm finds a solution, if it exists, approaches one.

Sampling-based planning generally separates the geometry of the space and obstacle collision detection from the planning problem. Collision detection is done by its own algorithm and is used in combination with the sampling to help search for feasible paths. The sampling algorithm can use either discrete searching or continuous sampling to find the paths.

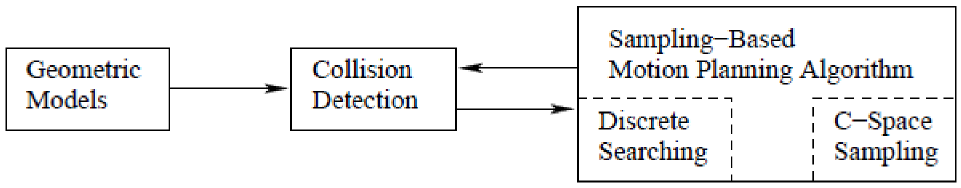


Figure 2.1: The Separation between Sampling-based Motion Planning Algorithms, Collision Detection, and the Geometric Models of the Environment [3]

The algorithm used in this thesis is a sampling-based path planning algorithm called RRT (Rapidly exploring Random Tree). The RRT algorithm is a type of rapidly exploring dense tree algorithm (RDT). The idea behind RDT algorithms is to build search trees that progressively improve the resolution and eventually densely cover the space, as defined above. The sampling can be either random or deterministic. If it is random, then the algorithm is an RRT. Below is the basic RDT algorithm in the case when there are no obstacles.

```
RDT($q_0$)
    G.init($q_0$);
    for i = 1 to k do:
        G.add_vertex($\alpha$ (i));
        $q_n \leftarrow$ nearest(S(G),$\alpha$(i));
        G.add_edge($q_n$,$\alpha$(i));
```

G is the tree that is grown. It consists of vertices and edges. It initially contains only the starting vertex q_0. S is the *swath* of the graph G, and is defined as the set of points that can be reached by G: it includes all the points on the edges of G as well as the vertices. The algorithm runs for k iterations. On each iteration, the newly sampled point is α(i). The nearest point in S to α(i) is q_n. If q_n is not a vertex in G, it becomes a vertex. The new edge (q_n, α(i)) is added to G.

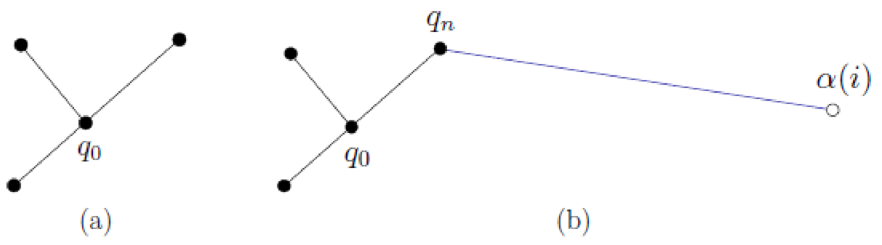


Figure 2.2: Extending an RDT Tree when the Closest Point is a Vertex [3]
(a) The tree at the beginning of the current iteration. (b) Point α(i) is sampled, and the tree is extended to include it by making an edge between α(i) and the closest point to it in the swath of the tree, q_n.

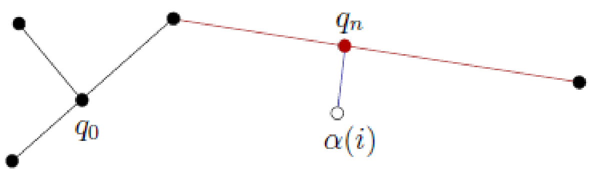


Figure 2.3: Extending an RDT Tree when the Closest Point is on an Edge [3]
If q_n – the closest point to α(i) – is on an edge, the edge is split in two and q_n is added as a new vertex to the tree.

If there are obstacles in the environment, then a step needs to be added when extending the tree. The algorithm has to check whether the edge between q_n and α(i) runs through an obstacle. If it does, there are several approaches to dealing with this situation. One approach is to ignore the sampled point α(i) and try sampling again until it finds a point that is reachable without obstacle collisions. Another approach is to extend the edge up to the obstacle boundary as close as the collision detection algorithm will allow, and add the end point of this edge to the tree instead of the sampled point α(i). This is the approach of the implementation used by this thesis.

There are several methods of finding the nearest point in S to α(i). The nearest neighbor can be computed exactly, or an approximation can be used. If the edges consist of straight line segments, the shortest distance between the sample point and an edge can be calculated easily using vector calculus. The algorithm can iterate over all the edges and find the closest one. Something similar can be done if the edges are curves, for example by using parameterization or by projecting the curves onto a cube or sphere. However, depending on the complexity of the curves, there may be some distortion introduced in this solution. Approximate solutions may involve adding a parameter Δq, such that no two vertices in G are allowed to be more than a distance of Δq apart from each other. When a new edge is inserted into the tree, it is broken into segments of length Δq, such that at the ends of each segment is a vertex. Then to find the approximate nearest neighbor, the algorithm only needs to iterate over the vertices of the tree, which greatly reduces computation time. There are other ways to decrease computation time even further, for example the use of Kd-trees, which are multi-dimensional extensions of binary search trees. Kd-trees divide a space into a hierarchical search tree, and allow the algorithm to locate the nearest neighbor by recursively searching the tree in time logarithmic in *k*, where *k* is the total number of vertices, instead of iterating over all *k* points.

The algorithm terminates either when the tree includes the goal state, or when it has run through all the iterations and not found a solution. The algorithm must periodically check if it is possible to connect the goal to G. If the sampling is deterministic, one way to achieve this is to sample the goal every *n* iterations, for some positive integer *n*. If the sampling is random, as with RRTs, then on every iteration, there can be some probability *p* that the algorithm will attempt to connect the goal, and probability 1-*p* that it will sample the space randomly. This is the method used by the implementation of RRT for this thesis.

The RRT algorithm has many advantages that make it a good candidate for path planning in UAV missions: it finds fairly short (though not optimal) paths, it finds paths quickly, enabling efficient real-time path planning, and it is straightforward to modify the algorithms to consider the kinodynamic constraints of the vehicles. Humans can collaborate with RRT by manipulating the search space. In addition, the randomized property of the produced paths makes it difficult for opponents to track the UAVs. However, a disadvantage is that operators supervising the UAVs may have difficulty collaborating with a randomized algorithm. Human-automation collaboration is very important in UAV missions, so to demonstrate the effectiveness of RRT this collaboration must be studied carefully. The next section provides an overview of relevant previous work in human supervision of unmanned vehicles and motivates human-automation collaboration.

2.2 Human Supervision of UAVs

In the future, human supervisors of UAVs are expected to be responsible for high level management during missions, while lower level tasks are expected to be carried out largely by automation [10]. Various aspects of UAV mission supervision and human-automation collaboration must be studied to determine how best to design such systems.

Operator workload is an important factor in UAV missions, as it affects the operator's ability to perform tasks such as path planning. Cummings and Mitchell [11] investigated the cognitive constraints on the number of vehicles that a human operator can control simultaneously. They modeled several sources of wait time during human-UAV interaction, including interaction wait time, wait time in a queue to interact with multiple vehicles, and situation awareness wait time. They found that the primary source of wait time delays was situation awareness wait time, defined as time during which at least one vehicle needed the operator’s attention, but the operator did not realize it. They also proposed an upper limit number of vehicles that a single human operator can control simultaneously, formulated in terms of the amount of time a vehicle can be ignored and the amount of interaction time required to raise a vehicle’s performance to the minimum acceptable level.

Marquez [12] studied the effects of automated path planning on high level mission planning. She researched human-automation collaboration in the domain of planetary surface exploration. She tested the effects of varying ratios of human and automation input and different visualizations in path planning in a dynamic environment. She found that the use of automation-generated paths gave better path performance but reduced situation awareness. She also determined that allowing the human operator to modify generated paths resulted in a decrease in path cost errors, which are percent errors calculated by comparing the path generated by the operator with the automation's optimal path. The health and status monitoring capabilities helped reduce task times. Her research shows that systems involving human-automation collaboration should help situation awareness, especially at times when automation is used more heavily.

Some path planning tools have been developed specifically to facilitate human-automation collaboration. For example, Carrigan [13] developed an automation tool called the Mobile Situation Awareness Tool (MSAT) to aid submarine commanders. MSAT provided automated path planning, in addition to health and status monitoring capabilities to increase the commanders' situation awareness. The planning algorithm used by the MSAT was A*; the algorithm would find paths by optimizing the constraints given by the commander, and taking into account minimum depth and visibility. Carrigan tested human subjects, and the experiments showed that compared to manual planning using paper charts, the auto-planning tool performed better in terms of both the lengths of the paths and the amount of time it took to find them [14].

Lesh et al [15] studied how human input can help computers solve problems that would otherwise be very difficult. They developed an automation system that can perform plan recognition, which is the understanding of the users' intentions from their actions. They used input from the user and occasional requests for clarification to make the plan recognition problem tractable, and demonstrated that adding their algorithm to an implemented collaborative system in the email domain decreased the amount of communication required from the user to accomplish various tasks. This thesis has a similar approach; it attempts to use human input to reduce the complexity of path planning problems.

Navigation is an important factor in UAV missions, yet it is a high workload, cognitive task. The goal is to find the balance between human control and automation in path planning, such that the role of the operators is small enough that they can focus on high-level mission management, yet significant enough that the paths are efficient. RRT has promising potential for allowing such a balance.

Previous Human-RRT Collaboration Work

This project is a continuation of previous Masters of Engineering research conducted by Caves ("Human-Automation Collaborative RRT for UAV Mission Path Planning") [6]. In his work, Caves implemented the RRT algorithm with a set of kinodynamic constraints known as Dubins flight constraints, and developed an interface for humans to interact with the program to guide vehicles to their targets while avoiding no-fly zones, which are regions such that if UAVs fly through them, the vehicles' likelihood of damage increases. Caves' experiment presented several modes of human-RRT interaction (with varying levels of human operator involvement), and tested each mode for performance in simulated UAV missions and subjective feedback. The experiment did not convincingly determine whether the use of an RRT in path planning has better performance than planning manually. One of the reasons for this was that the interface was unintuitive and hard to use, according to numerous test subject responses. The purpose of this thesis, therefore, is to redesign the interface and the modes of human-RRT interaction, and perform a similar experiment to determine whether the combined efforts of the human operator and the RRT algorithm produce better results in terms of mission completion times, total time spent in no-fly zones, and performance on a secondary task, than entirely manual control of the UAVs.

In Caves' experiment, the test users had to control several UAVs simultaneously, to try to guide them to their targets in as little time as possible while flying through as few obstacles (no-fly zones) as possible.

Caves designed two ways to influence RRT algorithm results – subgoal modification (before path generation with RRT) and waypoint modification (after path generation with RRT). Subgoals were operator-defined points which the generated path was required to include on the way to the goal. After a path was generated by the algorithm, it appeared with points equally-spaced along the line – waypoints; the operator could drag these waypoints to modify the generated path. In the experiment, there were four modes of operation: Human-Only (HO), Human-Guided 1 (HG1), Human-Guided 2 (HG2), and Human-Constrained (HC).

In the HO mode, the human operators had no interaction with the algorithm: they guided the UAVs to their targets by manually creating waypoint-defined paths (as they clicked each next waypoint, the path was extended from the previous waypoint with the shortest Dubins-constrained path). In the HG1 mode, human operators were still allowed to guide the UAVs manually, but they were also given the option to ask the RRT algorithm to generate a path for them. Once the path was generated (with no input from the operators), they could modify it by moving the waypoints. The HG2 mode forced the operators to use the algorithm. In this mode, if the operators wanted to generate or modify a path, they had to first use the RRT algorithm to search for a solution, and only after that they were able to change the produced solution. The HC mode was similar to HG2, except that it required additional user input: in order to run the RRT algorithm, the operator needed to define at least one subgoal.

Apart from modes of operation, the other parameter in the experiment was obstacle density. Obstacle density refers to the complexity of the environment, and is represented by the number of obstacles present at any given time. The obstacles did not move, but every five seconds a randomly chosen obstacle disappeared and reappeared in a different location. There were two levels of density – low density and high density. Low density missions contained 12 obstacles, while high density missions contained 20. In every mission, the operator controlled 3 UAVs.

The operator interacted with the interface largely by using keyboard commands. The mouse was used only for waypoint modification – a user could add a waypoint by left-clicking, delete a waypoint by right-clicking the desired waypoint, and move a waypoint by left-clicking it and dragging the mouse. In order to select a UAV, users would have to type in the number of that UAV; they could not select it by clicking it. In order to enter subgoals a user would click “s” to go into subgoal mode. In order to generate a path with the RRT algorithm a user would click “g”. The remaining information that was displayed in the interface was the time left in the mission and a list of commands and their corresponding keys. This interface did not have good learnability and visibility, two basic factors in good interface design [16].

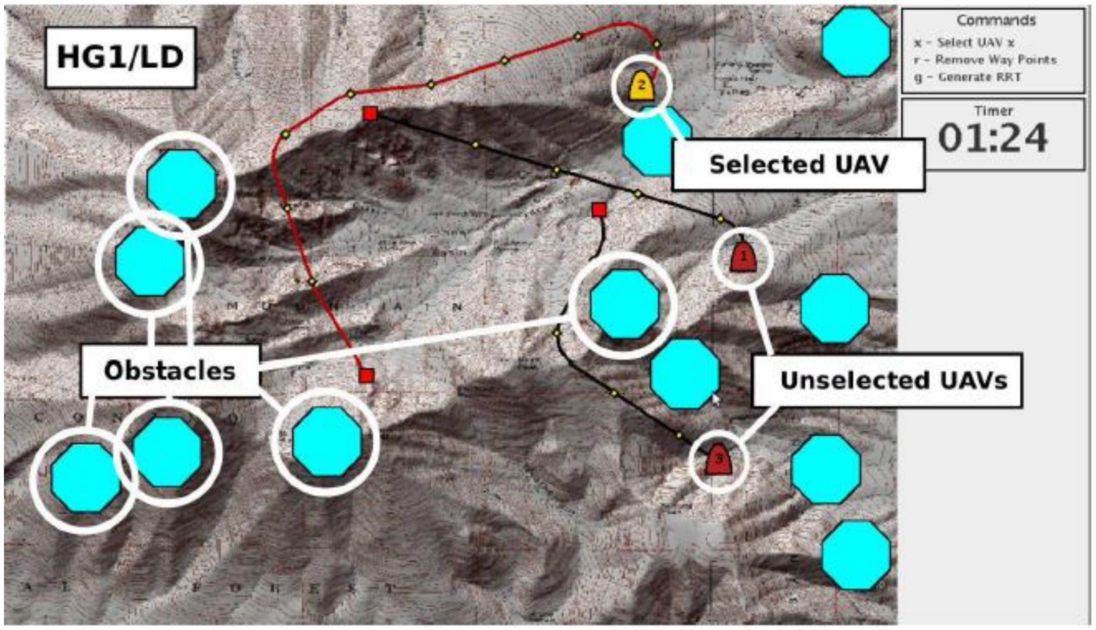


Figure 2.4: Caves' Interface: HG1 Mode with Low Density Obstacles

The results of Caves' work found that although the run time of RRT was the shortest in the HC mode due to the use of subgoals, this mode had the worst overall performance, while there was no significant difference between the other three modes. The user feedback showed a subjective dislike for the HC mode, but was similar for the other three modes, which indicates that the test users were indifferent between using RRT and controlling the UAVs manually, but did not like interacting with RRT by setting subgoals.

The purpose of this thesis was to develop a new interface for the simulation, with the objectives of good learnability and usability, and to design an experiment based on the new interface to test human interaction with RRT. The experiment and the interface are described in Chapter 3.

Chapter 3: Experimental Design

This chapter presents the experimental design used to test various human interactions with an RRT algorithm. The first section explains the implementation of the RRT algorithm used for the experiment. The second section gives an overview of the interface. The third section describes the experiment and the variables measured during the experiment. The fourth section explains how the experiment and the measurements made will help answer the research questions of this thesis.

The simulation program was written entirely in Java, version 1.6. The only external resource used was a TableLayout jar file, necessary for creating the layout of the side panels of the interface (described later in this chapter). The program was run in Eclipse, release 3.7.0. The experiment was conducted on a 64-bit Lenovo ThinkPad W500 laptop running Windows 7, with a 15.4 inch screen, and a screen resolution of 1280 by 768 pixels.

3.1 RRT Implementation

Caves programmed the first version of the simulation in a very modular way, completely separating the back end RRT algorithm from the front end interface. This facilitated easy changes in future versions. Most of the work for this thesis was done on the front end, while the implementation of the RRT algorithm was kept basically the same as Caves' implementation [6]. This section gives details of Caves' implementation of the path planning algorithm, as well as the possible methods of operator interaction with it that were developed and studied for this thesis.

3.1.1 Collaborative Human-Automation RRT

The algorithm used in the previous and current efforts differs from the basic RRT algorithm [3] in several ways. First, it finds Dubins-constrained paths: curvature constrained paths that are feasible for UAVs. Second, the user can constrain the solution by using "wayareas" (explained in section 3.1.2). Third, the user can modify the solution found by manipulating waypoints along the paths (explained in section 3.1.2). This type of RRT implementation is called a collaborative human-automation RRT, or c-RRT. Figure 3.1 shows the pseudo-code for RRT.

```
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($q_a$,$q_b$,$T$);
```

Figure 3.1: RRT Pseudo-code [6]

The environment represented in the interface is a 2-dimensional continuous space; it is a subset *C* of $\mathbb{R}^2$. Obstacles can be any connected enclosed subsets of $\mathbb{R}^2$. For the purpose of the experiment, it is necessary to be able to quickly determine whether a sampled point is inside an obstacle, therefore Caves chose to represent obstacles as regular octagons [6] without loss of generality, and this representation was kept in this iteration of the interface. The search space is then the area that lies inside *C* but not inside *O*, where *O* is the set of obstacles. This space can be mathematically denoted as the set difference $C \setminus O$.

The c-RRT algorithm initializes its tree with only the start node, q_{start} in the pseudo-code above. Then it continues to grow its tree until the tree contains the goal, q_{goal}. To grow the tree, it performs three functions: *randomState()*, which samples randomly over the search space $C \setminus O$ and returns a sampled point q_a, *bestNode(q_a, T)*, which returns the node q_b in the tree T that is closest to the sampled point q_a, and *extend(q_a, q_b, T),* which extends the tree T from q_b to include q_a.

The sampling method *randomState* is fairly simple. With probability *r* it returns a random point from the search space $C \setminus O$. With probability *1-r* it returns the goal state. The value of *r* needs to be set high enough so that a large portion of the search space is explored. In addition, if *r* is set high then by the time that *randomState* returns the goal state, there will likely be a relatively short collision-free path available from the tree to the goal; this will result in better paths. However, the value of *r* should not be set so high that the algorithm takes a long time to terminate or consumes too much memory due to the size of the tree. Caves extensively experimented with the value of *r* and empirically decided to set *r* to *r*=0.95 [6]. This value was kept in the updated version of the interface.

The method *bestNode* returns the best node in the tree to extend to the sampled point returned by *randomState*. The best node is generally considered to be the node with the smallest Euclidean distance to the sampled point. However, since in this environment the paths are planned for UAVs, they have curvature constraints. Therefore in his implementation, Caves used a distance metric that takes into account these constraints. Studies have shown that using a metric that is based on system constraints instead of Euclidean distance to select the best node in RRT produces smoother solutions and explores the search space more thoroughly [17].

The *extend* method extends the tree by generating a Dubins path from q_b to q_a. The tree is not continuous; this means that not all points on the path from q_b to q_a are added as nodes in the tree. Instead, the algorithm makes nodes at regular intervals along the path. If the path from q_b to q_a goes through an obstacle, nodes are added to the tree along the path up until the obstacle boundary. As soon as a point that is to become a node is determined to be inside an obstacle, the method returns and the tree is not extended any further. In addition to location, each node in the tree has a heading – the direction that the UAV is moving in at that node. The heading is determined by the node's spatial relationship with its parent in the tree. It is necessary to know the headings of the nodes in order to calculate feasible paths. For example, if a UAV is currently flying east, in order to reach a point *p* that lies to the west of the UAV, a feasible path will need to turn around; a path that does not account for heading will stretch from the current location to *p* in a straight line, assuming there are no obstacles in the way. A few iterations of the algorithm are shown in Figure 3.2.

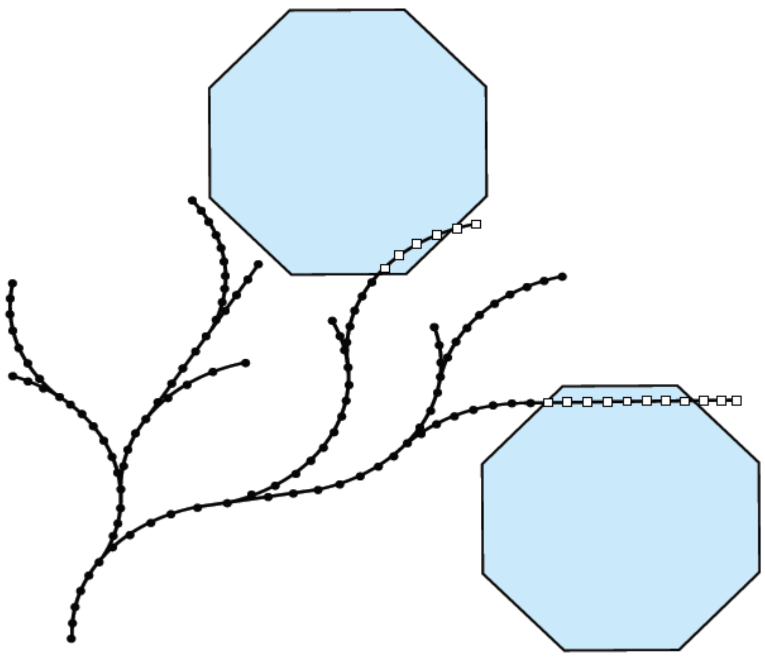


Figure 3.2: The RRT Algorithm [6]

The tree consists of nodes (black circles) spaced equidistantly along Dubins paths. If a path collides with an obstacle, points beyond the collision are not included in the tree (white squares).

3.1.2 Human Input

There are several types of human input to path planning in this interface. Human operators can constrain the search space of the c-RRT by defining wayareas. Operators can define paths manually and modify RRT solutions by manipulating waypoints. Waypoints are a method of interaction that was retained from Caves' implementation, while wayareas were developed for the new version of the interface.

Wayareas

There are two main reasons for allowing human operators to collaborate with the RRT path planning algorithm: 1) using human input can reduce the complexity of the problem and decrease the computation time required to solve it, and 2) human input guides the algorithm toward finding a solution that the human operator will be satisfied with.

In this interface, there is one mode – human guided mode – where operators are given the ability to make wayareas, represented as semi-transparent yellow circles overlaid on the display. The purpose of wayareas is to indicate general areas that the operator wants the path to go through. Operators have control over the center and the radius of the wayareas, so if they want the path to go through a precise area, they can draw a small wayarea there, and if there is a wide space they want the path to pass through they can draw a large circle that covers the majority of the space. Operators can draw as many wayareas as they want.

The wayareas help break down the path finding problem into smaller problems. After making wayareas, when the operator requests an RRT path, the algorithm orders the wayareas by distance from the UAV's current location from closest wayarea to farthest wayarea, where distance is calculated from the center of the wayarea. This method of ordering wayareas was chosen over the order in which the operator made the wayareas so that if the operator decided to add a wayarea in between two other wayareas, he or she would not have to delete all following wayareas to produce the correct order. In addition, it was determined that in this experiment, there are no practical paths for which this ordering of wayareas would not be effective. After ordering the wayareas, the algorithm then samples each wayarea randomly to produce a point inside that wayarea. If that point is inside an obstacle, it samples again. If after ten tries the algorithm has not sampled a point that is outside of any obstacles, that wayarea is ignored. This way, if an operator makes a wayarea that overlaps with an obstacle or if an obstacle appears inside a wayarea after its creation, the algorithm does not take too long looking for a feasible point in the wayarea. The limit of ten sampling attempts was chosen to balance responsiveness to user input and computational efficiency. After sampling the wayareas, the automation has a sequence of subgoals; it iteratively looks for a path from each subgoal to the next subgoal, starting with the UAV's location and ending with the goal state.

When sequentially growing the trees, the heading of the root node of each tree is assigned to be the heading of the goal node of the previous tree. This way, the curvature constraints are guaranteed to hold from the start to the end of the overall path. Wayareas are a way to reduce the randomness of RRTs and make them more acceptable to human operators. If they are used efficiently, they can reduce computation time because they can break a hard problem into several much easier problems.

Waypoints

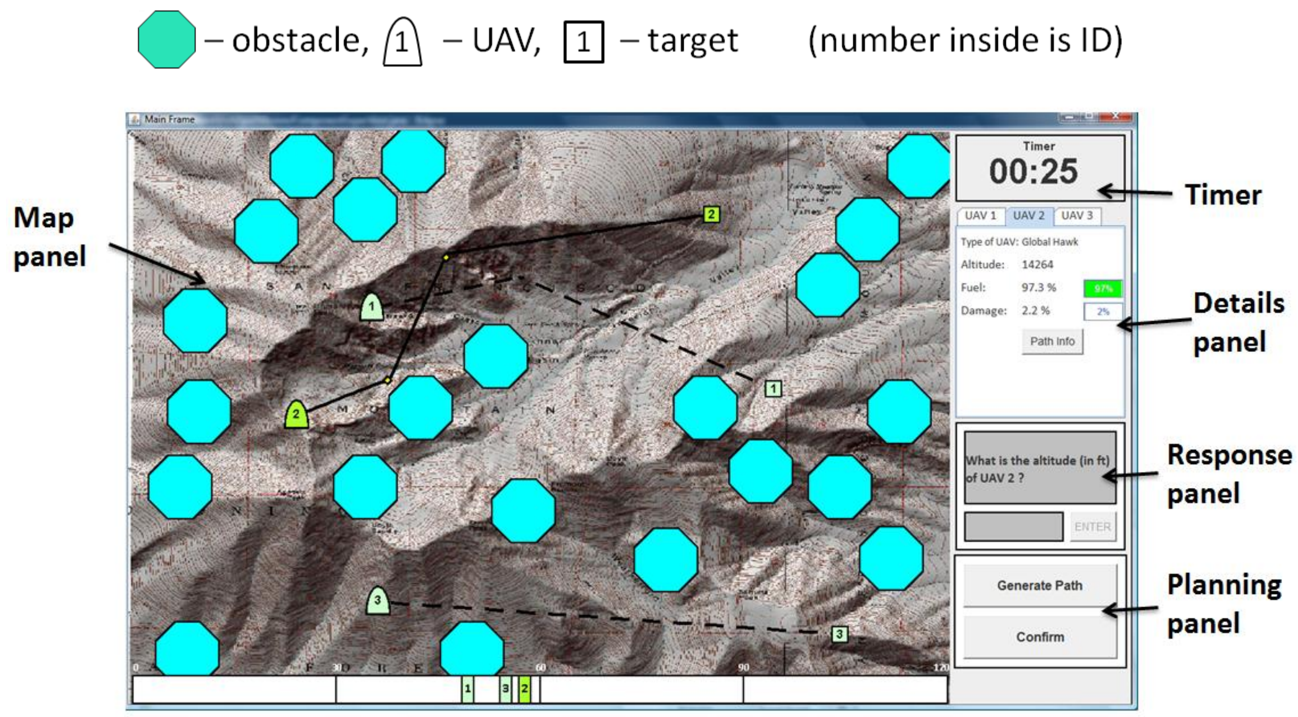
When planning paths manually, operators can define a sequence of waypoints. Besides the waypoints, all that is necessary to specify the waypoint-defined path is the UAV's start location, its target, and its initial heading. The path can then be formed sequentially by finding the shortest Dubins path from each point to the next point in the sequence, the same way that *extend* grows the RRT tree. Just like in the algorithm used with wayareas, the UAV's heading at each point is assigned to be the heading at the goal state of the last Dubins path. This way, the entire path is guaranteed to meet curvature constraints, and only the initial heading needs to be known.

The path found in this way is not guaranteed to be the optimal path from the start state to the end state through the specified waypoints. The reason for this sub-optimality is that the heading at each waypoint is calculated only based on its predecessors [18]. Calculating the optimal headings at all the points is computationally too intensive for real-time path planning.

In the interface, there are two modes where operators interact via waypoints (the third mode involves wayareas) – human only and automation. In one of the modes they use waypoints to initially define paths and to modify paths. In the other mode, the paths suggested by the c-RRT algorithm display movable waypoints at regular intervals along the path. If operators choose to add, delete, or move a waypoint, the path is modified in the way described above: the shortest Dubins paths are found between each sequential pair of affected waypoints.

3.2 The Interface

An interface was designed for the experiment to allow human interaction with the RRT algorithm to plan paths for multiple UAVs in simulated missions. The interface was designed to be as intuitive as possible, so that participants could focus on guiding UAVs and not on learning to work the interface. Figure 3.3 shows the interface.



Timer
00:25

UAV 1 | UAV 2 | UAV 3

Type of UAV: Global Hawk
Altitude: 14264
Fuel: 97.3 % 97%
Damage: 2.2 % 2%

What is the altitude (in ft) of UAV 2 ?

Figure 3.3: The Environment Used in the Experiment

UAVs are represented as bullets; targets are represented as squares. A UAV and its target have the same number ID, shown inside the shape. Unselected UAVs are light green in color. When a UAV is selected (only one can be selected at a time), the UAV and its target change to bright yellow-green. Paths are shown in black. Yellow points on paths are movable waypoints (present only in human only and automation modes). The path of the selected UAV is a solid line; the paths of unselected UAVs are displayed as dashed lines and the waypoints on them, if any, are hidden. This design allows operators to distinguish the selected path from unselected paths and eliminates distractions by hiding the waypoints on other paths.

The map panel displays UAV paths and obstacles. An operator can manipulate paths by interacting with the map panel. The timer shows how much time has elapsed since the beginning of the current mission. The details panel presents UAV status, including altitude, fuel, and percent damage. There is also a button that can be clicked to display path information, consisting of path time, path length, and path fuel. The response panel occasionally poses questions about the status of a particular UAV. This is necessary to measure the cognitive load on the operator via a secondary task (explained in section 3.3.1). The response panel flashes when a new question appears. The path planning panel is used for generating paths. In every mode of operation, clicking "Generate Path" will output a path that connects the selected UAV to its target. Clicking "Confirm" will let the selected UAV start moving, if it was waiting for a path. If the UAV is already moving, it will automatically switch to the newly generated path. Unselected UAVs that have a target but are not moving flash orange to attract the operator's attention. The timeline at the bottom of the map panel shows the arrival time in seconds of each moving UAV at its next destination. It is a useful tool for learning which UAV will be the next to arrive at its goal and so will need the operator's attention soon.

Obstacles are represented as octagons, without loss of generality. They are no-fly zones; the UAVs can fly through them but participants are warned that there is a large penalty for doing so. Obstacles do not move around, but they occasionally appear or disappear. If a UAV's path goes through an obstacle, that obstacle turns yellow to warn the operator. Participants are informed that if an obstacle appears and a UAV is inside it at the time of its appearance, this does not count as an incursion.

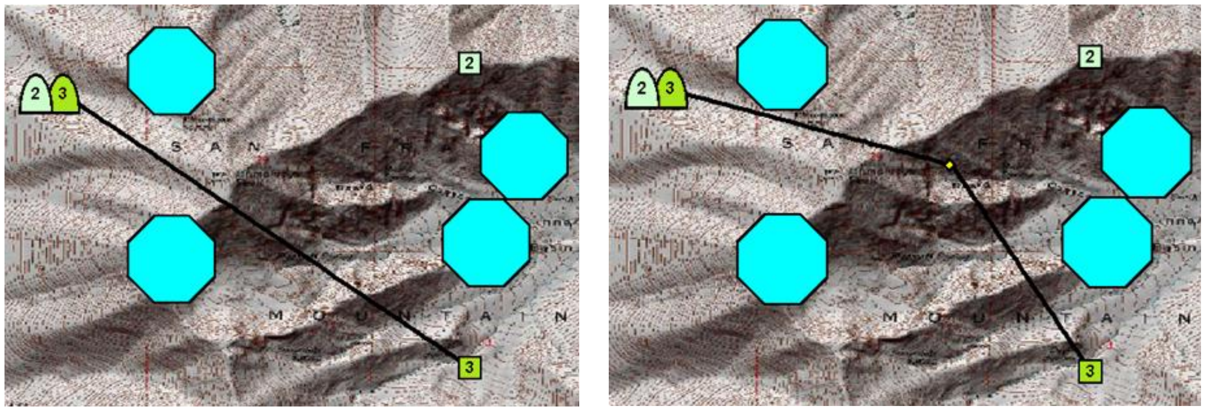
3.2.1 Modes of Operation

There were three modes of operation in the experiment. Each mode allowed the user to interact with the c-RRT path planner in different ways. In the **Human Only (HO)** mode, the human operators plan paths manually without relying on the RRT algorithm at all. They click waypoints on the map, and to create the path, the waypoints are connected with the shortest Dubins-constrained trajectory. In the **Human Guided (HG)** mode, operators can only use RRT to plan and replan paths, but they can influence the solution found by the algorithm through inputting wayareas. Wayareas are circles drawn by the operator on the map to indicate general areas to take the path through. A solution found by the RRT will go through each wayarea. In the **Automation (A)** mode, operators must use the RRT algorithm to plan paths initially, but they can modify suggested paths and replan either manually or by using the automation. In both HG mode and A mode, operators can continue to request new paths until they find one that they approve.

The modes of operation were designed to allow three different types of interaction with the c-RRT algorithm: 1) no interaction (in HO mode), 2) the ability to influence the solution found by the algorithm (in HG mode), and 3) the ability to manually modify solutions found by the algorithm (in A mode). In the human guided and automation modes, the operators were forced to use the algorithm to plan paths. They were not required to input wayareas in HG mode or to modify the paths manually in A mode, so they could only use the automation for path planning if they preferred. These methods of interaction were separated in the two modes to study their performance and operators' behavior with them more carefully.

Human Only Mode

Human only mode has manual path planning, with no involvement of the RRT planning algorithm. In order to generate a new path, operators may click on the map to make waypoints. When they are done making waypoints, they click "Generate Path"; this connects the waypoints with a Dubins-constrained path. If the operator does not make any waypoints before clicking "Generate Path", the UAV will be directly connected to the goal with a Dubins-constrained straight line path. To modify a path either while a UAV is waiting or when it is moving, operators can add waypoints by left-clicking the location of the desired waypoint, delete waypoints by right-clicking the waypoints, and move waypoints by clicking and dragging the existing waypoints. Adding a waypoint connects the two closest waypoints to it on the path with Dubins-constrained shortest paths (basically straight line paths). Deleting a waypoint connects the preceding and following waypoints with a Dubins-constrained shortest path.



a) Path before adding waypoint b) Path after adding waypoint

Figure 3.4: The Process of Adding a Waypoint to the Path of UAV 3

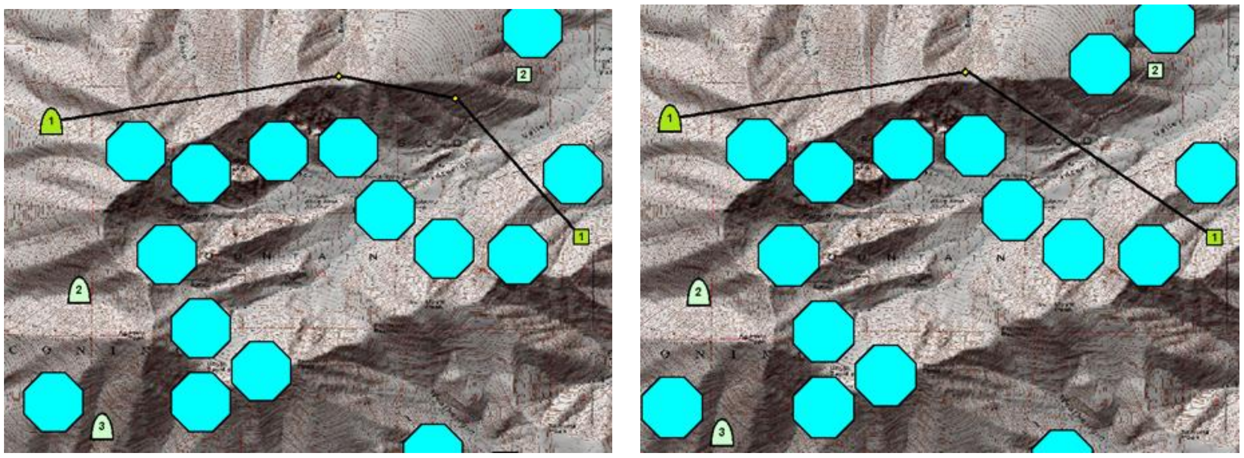

a) Path before deleting waypoint b) Path after deleting second waypoint

Figure 3.5: The Process of Deleting a Waypoint from the Path of UAV 1

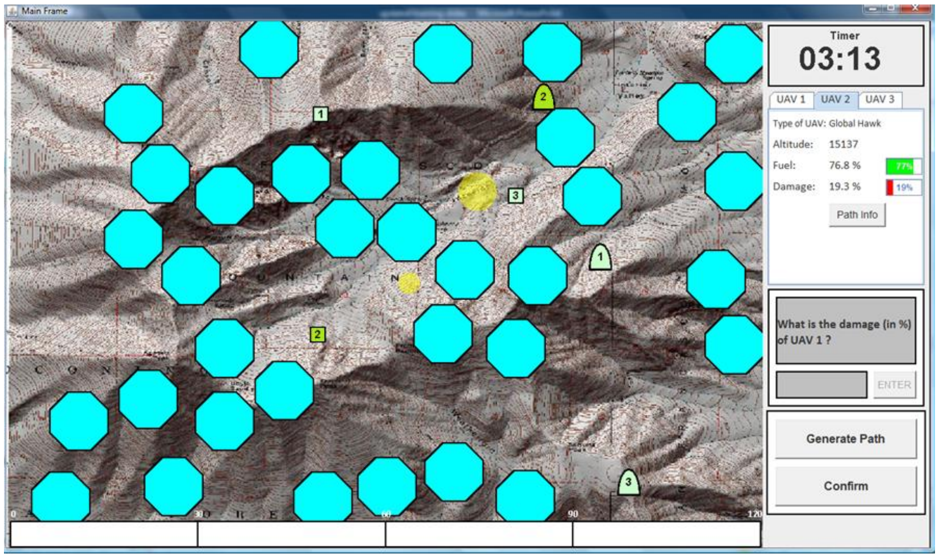
Human Guided Mode

In the human guided mode, operators must use the RRT algorithm to plan paths, but they may influence its solutions. Human operators can influence the c-RRT path planner by making wayareas, shown as semi-transparent yellow circles in the interface. To create a wayarea, the operator clicks on the map for the center of the circle, and drags the mouse to increase the radius. The wayarea indicates a general area that the RRT should guide paths through, but the idea is that inside the area the operator has no preference where the path goes. Therefore, if operators want the UAV to go through a specific point, they should draw a small wayarea, whereas if there is a large space they want the UAV to go through, they can draw a large wayarea.

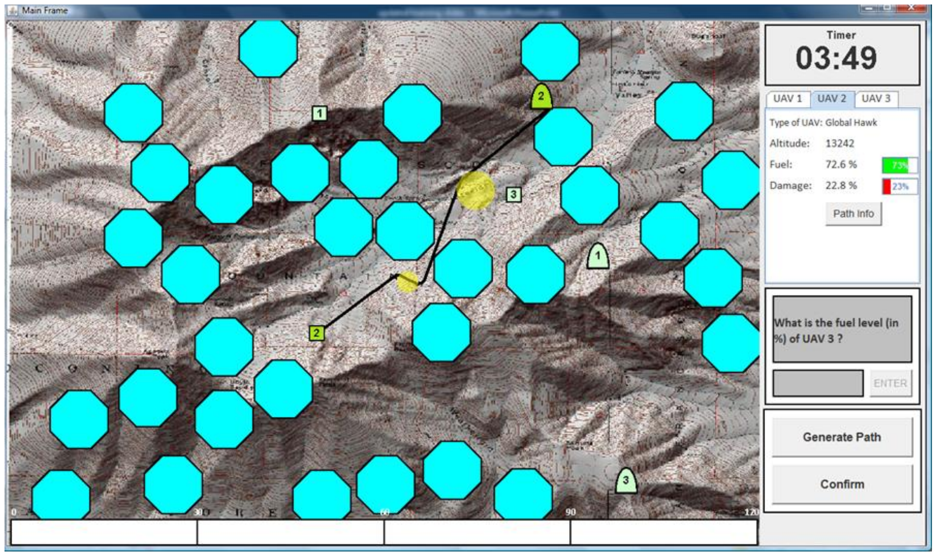
When the operator is done making wayareas, clicking on "Generate Path" starts the search. The RRT algorithm searches for a path that passes through each wayarea, from closest to farthest from the UAV on the way to the target. If a large portion of a wayarea is contained inside an obstacle, that wayarea may be ignored in the search; the algorithm uniformly samples over the area inside each wayarea a limited number of times, and if in those times it does not find a point not inside any obstacles, it will skip that wayarea. If the c-RRT algorithm does not find a feasible path that passes through each wayarea in the iteration limit, it outputs a straight-line Dubins-constrained path from the UAV to its target. The operator may change the wayarea configuration by adding or deleting wayareas (by right-clicking, as in HO mode), and then may try to generate a path again. If the operator does not make any wayareas before clicking

"Generate Path", the RRT will perform a normal unconstrained search from the UAV to its target.

While a UAV is waiting to start moving, when a path is generated the wayareas remain so that if the operators are unhappy with the path, they may change the wayarea configuration or simply click the button again. However, if a UAV is moving, then when a path is generated the UAV's path is automatically updated to the generated path, and all wayareas for that UAV are removed. All generated paths contain no waypoints; no manual modification of the paths is allowed. If operators need to replan, they must follow the process outlined above: make wayareas for the selected UAV and click "Generate Path". As a result, it is impossible to modify only a section of a path when replanning; the entire path is changed.



a) Wayareas are input for UAV 2



b) The path generated goes through the wayareas

Figure 3.6: The Path Planning Process in HG Mode

Automation Mode

In automation mode, operators must use the RRT algorithm to generate initial paths, but they can modify the solutions manually and can replan manually as well. To generate a path, the operator must simply click "Generate Path" and the automation will output the solution found by c-RRT for the selected UAV. Waypoints are displayed along the generated paths at regular intervals. The operator can modify the paths as in HO mode, by adding, moving, and deleting waypoints. If the RRT algorithm does not find a feasible path before the iteration limit, it will output a Dubins-constrained straight-line path directly from the UAV to the goal. If there are obstacle incursions (most likely there will be in this case), the operator may then either let the algorithm attempt again, or may modify the path manually. The same holds true when an operators are replanning for a UAV that is moving: they may either click "Generate Path" and let the algorithm output a solution that avoids obstacles, or they may manipulate waypoints themselves to change the path.

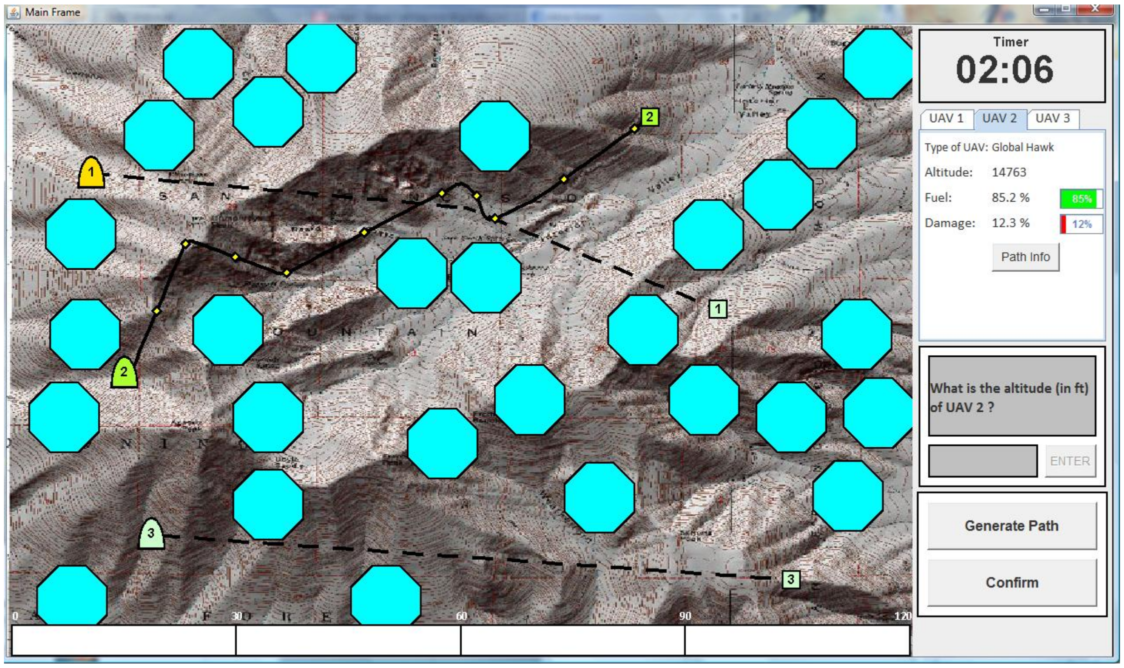
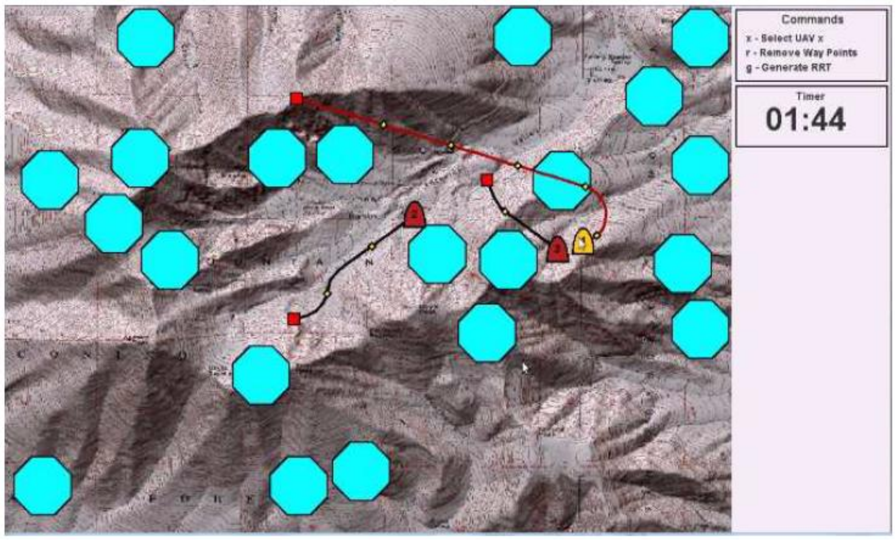


Figure 3.7: Automation Mode

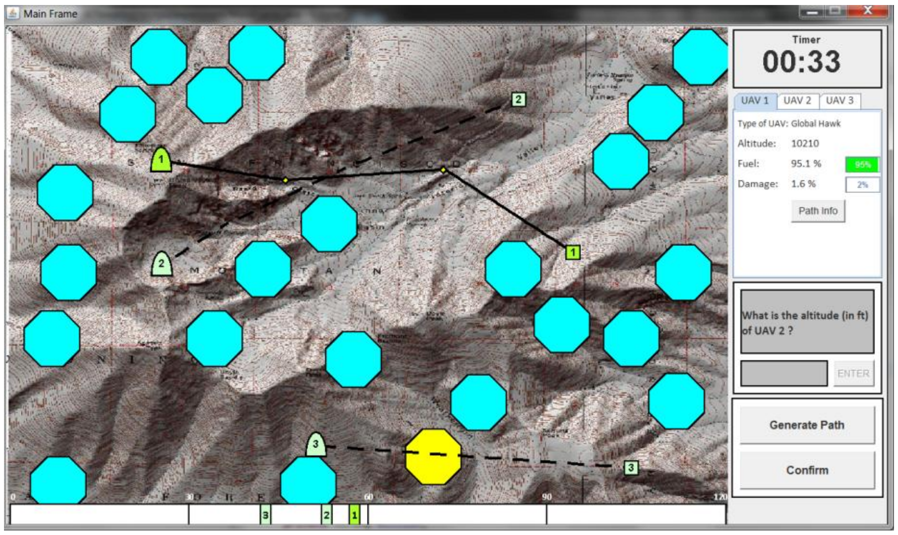
In the two modes involving RRT – human guided mode and automation mode, operators can continue to click "Generate Path" to attempt to find paths they are more satisfied with; the algorithm is randomized and so will output a different path each time. If the algorithm does not find a feasible solution, it will output a Dubins-constrained shortest path from the UAV to its target (straight-line path) if the UAV is stopped and does not have a path; if the UAV is moving, it will keep its current path. In the RRT modes, each UAV has its own thread; while some of the UAVs are searching for solutions, the operator can plan paths for other UAVs. When the algorithm is in the process of finding a path for the selected UAV, the "Generate Path" button is disabled, to let the operator know that the algorithm is still searching.

3.2.2 Changes from Caves' Version

This section summarizes the major changes made to the interface in the current version of the program. Figure 3.8 shows Caves' interface and the current version.



a) Caves' interface



b) Current interface

Figure 3.8: Interface Versions

First of all, operator interactions with the interface were made more intuitive and easier to use. In the current version of the interface, operators select UAVs by clicking on them, and generate and confirm paths by clicking the large buttons "Generate Path" and "Confirm" in the path planning panel. In Caves' interface operators used keyboard shortcuts, which can be more efficient than pointing and clicking, but take more time to learn and are easy to forget. In Caves' interface there was a default path displayed for each UAV; it was the shortest curvature-constrained path from the vehicle to its goal. As soon as a mission started, the UAVs immediately started moving along this path toward their targets. In the current version, the UAVs start out stationary so that operators can take time to adjust their paths as necessary, and are not overwhelmed at the beginning. This also means that operators can generate a new path multiple times until they find a satisfactory one. Only the path of the selected UAV is displayed with its regions of interaction (waypoints or wayareas), so as not to distract operators from the path they are currently modifying. Instead of distinguishing the selected path from unselected paths by changing its color to red, the selected path is displayed by a solid line, while unselected paths are displayed by dashed lines.

Several new features were added to the interface. One such feature is the timeline at the bottom of the map panel that shows when each UAV will arrive at its next destination. This visualization helps operators decide which UAV to focus on next, either because it will be in need of a new path soon, or because its path is relatively long compared to the other UAVs and could be modified to become shorter. Another such feature is obstacle warnings; obstacles turn yellow when UAV paths go through them in order to warn the operator. A third example is the feature that when UAVs are in need of a path, they continually flash orange to attract the operator's attention. These features were added to help operators with the aspects of UAV missions that are not directly related to the process of path planning.

An important addition to the simulation is the secondary task. In Caves' experiment, in addition to subjective ratings of performance and frustration, participants were asked to rate workload, on a four-level Likert scale. For this experiment it was decided that objective measures of workload, which can be acquired by having a secondary task, would provide more useful data.

This interface has three modes instead of four, and they allow different interactions with RRT than Caves' experiment had. The human only mode in this version allows the same methods of interaction as Caves' human only mode, however the modes involving RRT are fairly different. The modes in the current version separate operator input before and operator input after running the algorithm, to study their effects independently. The differences in the workload, performance, and subjective feedback for the two types of interaction will guide the development of future iterations of the interface. Although Caves had waypoint manipulation in some of his modes, wayareas were developed for this version to replace subgoals. The idea behind wayareas is to allow operators to select general areas for UAV paths to travel through. Instead of varying obstacle density between participants in this experiment, the second independent variable (the first was the mode of operation) was the number of UAVs controlled by participants. All missions were completed in a high density environment. The reason for this change is that in a low density environment, paths could often be defined with only one or two waypoints, and so the full potential of using the algorithm would not be apparent. The effect of increasing the number of UAVs was decided to be a more interesting research topic.

Finally, another important change is in the use of the RRT algorithm. In the previous version, the interface and the RRT algorithm were on one thread; while the algorithm was searching for a solution for a UAV, the interface would freeze, and the operator could continue interacting with it only after the algorithm returned. In the current program, the UAVs are multithreaded; each UAV has a thread with its own instance of RRT. While RRT is searching for a path for one UAV, the interface continues to update and operators can plan paths for other UAVs. This change should decrease mission completion times and frustration because operators can now plan paths for separate UAVs concurrently instead of having to wait for the algorithm to output a path for one UAV before moving on to the next UAV.

3.3 The Experiment

The experiment involved three missions with multiple UAVs. Each mission was completed in a different mode of operation. In all the missions the operator's objective was to guide each UAV to a sequence of four targets as quickly as possible while avoiding obstacles.

Each operator performed three missions, one in each mode. The other variable besides mode of operation is the number of UAVs in the missions. Each of the three missions performed by an operator had either 3, 5, or 7 UAVs.

3.3.1 Missions

In all missions, the UAVs have four targets each. The UAVs start without a default path. A UAV waits to start moving until the operator finds a path for it and confirms it, and then the UAV starts moving at constant speed along the path toward its target. When it reaches its target, it stops again. Only one target is visible at a time for each UAV; as soon as a UAV reaches a target, its next target appears. Operators must guide the UAVs to their targets while avoiding obstacles as best they can. A mission ends when every UAV has reached its last target.

Apart from guiding UAVs, there is a secondary task: approximately every 30 seconds, the response panel displays a question for the operator to answer. The time interval between when two consecutive questions appear is a random value selected from the range 25 seconds to 35 seconds. The questions ask for the altitude, fuel level, or percent damage of a particular UAV. The operator can find this information in the details panel above the response panel. Participants are told that they should try to answer the questions, but the UAVs are their first priority: if they have a choice between answering a question and helping a UAV avoid a collision with an obstacle, they should help the UAV avoid a collision. The purpose of the secondary task is to measure the operator's cognitive load during a mission. Presumably, when operators are idle, they will answer the questions, whereas if they are busy manipulating UAVs, they will not answer the secondary task.

Each mission has a mission file corresponding to it. The mission file specifies the starting location and sequence of target locations for each of the UAVs, and the locations and appearance and disappearance times for each obstacle. A few times a second, the computer executes a "timer task", which causes the UAVs to update their positions and checks whether it is time to add or remove any obstacles. For a given number of UAVs, all participants carry out the same missions in the same order. However, the order of the modes of operation (e.g. HO, HG, A or HO, A, HG) is varied between participants. For different numbers of UAVs, the missions are different. Therefore there was a total of nine missions in this experiment. For a given number of UAVs, the missions were designed to be of approximately the same level of difficulty; the author of this thesis repeatedly completed all three missions in human only mode with mission completion times within 10% of the average mission completion time.

3.3.2 Experimental Protocol

For each participant the experiment lasted an average of 45 minutes, and at most an hour. First, participants signed a consent form (Appendix A). Next, they went through a two-part training. The first part consisted of reading through training slides that explained the context of the experiment and how to use the interface, as well as the differences between the modes of operation (Appendix B). The second part was practice using the actual interface. Participants were given up to ten minutes to practice controlling UAVs in a training mission. The training mission had three UAVs, each with two targets. It was simpler than the actual missions. Training always started in human only mode, continued to human guided mode, and finished with automation mode. When participants finished the mission in one mode, they would click "Restart Training" to choose the next mode and restart the mission. After carrying out the training mission in all three modes, the actual experiment started.

During the experiment, participants completed three missions, one in each mode of operation (human only, human guided, and automation). Participants were instructed to attempt to guide the UAVs to their targets as quickly as possible while avoiding obstacles as well as they could, and answering secondary task questions whenever they were not occupied with the UAVs. The participants controlled either 3, 5, or 7 UAVs in each mission. In order to avoid unknown outside variables, the order of the modes of operation presented in the experiment and the number of UAVs in the missions was randomized and counter-balanced. After each mission, the participants filled out a short survey about the mission that asked for their subjective assessment of their performance and frustration level, on a 5 level scale. The possible answers for performance were "very poor", "poor", "ok", "good", and "very good". The possible answers for frustration were "very high", "high", "medium", "low", and "none" (Appendix C). After completing the three missions, participants were asked further questions, including their order of preference for the three modes, feedback on the interface, whether they had enough training at the beginning of the experiment, whether they have video game experience and what kind of games they usually play, and their field of study (Appendix E). The questions were asked and answered orally, so that the participants would be able to talk freely and elaborate on their comments. Their answers were typed as they spoke.

3.3.4 Variables and Measurements

The independent variables in the experiment are the mode of operation and the number of UAVs. The experiment was a 3 (HO, HG, A) x 3 (3, 5, 7) repeated measures analysis of variance (ANOVA), with one within factor (mode) and one between factor (number of UAVs). The dependent variables that were measured include total mission completion time, average mission completion time, number of obstacle incursions, total time spent inside obstacles, number of interactions for various types of interactions, percentage of secondary tasks answered, percentage of secondary tasks answered correctly (within a margin of 10% of the correct answer), time it took to answer secondary tasks, algorithm runtime, algorithm solution length, self-rated performance, operator frustration, and subjective order of preference of the modes.

The dependent variables were measured based on data logged during the experiment, with the exception of the subjective order of preference of the modes, which was given during the subjective feedback after the experiment. For each participant, data was logged in three log files: an event file, which wrote event-triggered data, a snap file, which recorded a snapshot of the interface every two seconds, and a survey file, which recorded the participant's self-rated performance and frustration after each mission. Events that were logged in the event log file include operator interactions such as adding, deleting, or moving waypoints, adding or deleting wayareas, clicking on "Generate Path" or "Confirm", or answering a secondary task question. Other logged events include the secondary task appearing, a UAV reaching a target, the success or failure of the RRT algorithm, and the appearance or disappearance of obstacle warnings. All logged events contain the name of the event, a timestamp, and the ID and location of the relevant UAV, as well as other relevant factors depending on the event. Snap files contain data on obstacle proximity – the distance to the closest obstacle for each UAV, obstacle locations, and the current path of each UAV, specified by a list of points. Appendix D has a full list of the data that was logged in the event and snap files.

Total mission completion time is the time it took until all the UAVs reached their last targets. Average mission completion time is the mean amount of time it took for each of the UAVs to reach its last target in the mission. The number of obstacle incursions is the number of times that a UAV flew through an obstacle, given that the obstacle did not appear with the UAV inside it (those incursions were not counted because participants had no chance to avoid them).

Participants were instructed to try to get out of obstacles as quickly as they could, as the total time spent inside obstacles was also measured. The total number of interactions counts the number of times that participants added, deleted, or moved a waypoint, added or deleted a wayarea, or pressed a button in the path planning panel. Different modes have different types of interactions; for example participants can manipulate waypoints in HO and A modes, but they work with wayareas in HG mode. The time taken to answer a secondary task is the time between when the task appeared and when the operator answered it. The self-rated performance and operator frustration were collected from the surveys filled out by the participants after every mission. In addition, test users answered several questions after the experiment asking for further feedback on the modes of operation and the interface.

3.4 Relation to Research Questions

The measurements made during the experiment will help to answer the research questions presented in chapter 1.

Research Question #1: Does collaborating with an RRT algorithm in path planning decrease the cognitive load of the operator over manual path planning?

Cognitive load is measured by the secondary task in the experiment. The relevant dependent variables are the percentage of secondary tasks answered and the average time taken for participants to respond to secondary tasks. The question can be answered by comparing these variables for the modes involving RRT – HG and A – and the human only mode.

Research Question #2: Does either mode involving automation result in better performance, where performance is measured by mission completion time and obstacle avoidance, than human-only mode? Which mode involving automation performs better?

The dependent variables relevant to this question are total mission completion time, average mission completion time, the number of obstacle incursions per UAV, and total time spent inside obstacles per UAV. Looking at the data for these variables can show which of the three modes performs better in these metrics, for a given number of UAVs. In addition, RRT runtime and number of failures were measured for HG and A modes to determine whether human input into the algorithm improves the performance of the algorithm.

Research Question #3: How do human operators prefer to interact with RRTs to plan paths? How does using RRTs affect operators' perception of their performance?

The operators' subjective feedback after the experiment, as well as their answers to the survey questions after each mission will show whether humans find that RRTs are helpful, and how they prefer to interact with RRTs in path planning.

Research Question #4: How does RRT path planning impact performance and subjective assessment as the number of UAVs increases?

For each of the above three questions, the results can be compared for 3 UAVs, 5 UAVs, and 7 UAVs, to see if either cognitive load, performance, or subjective feedback changes with the number of UAVs.

Chapter 4: Experimental Results

4.1 Experimental Design and Analysis

A total of 36 people participated in the experiment, 25 males and 11 females. Participants were required to be between the ages of 18 and 50, however 86% were undergraduate and graduate students (participant ages were not recorded). The participants signed an informed consent form before participating in the experiment. They then went through training slides at their own pace. The training slides are in Appendix B. After reading the slides, they were given 10 minutes to practice controlling UAVs using the actual interface in each of the three modes of operation: human-only, human-guided, and automation.

Every participant completed three missions during the experiment, a mission in each of the three different modes. A participant controlled either 3, 5, or 7 UAVs in all missions. For each number of UAVs, there were 12 participants that were assigned to control that many. For a given number of UAVs, all six possible mode orders (HO HG A, HO A HG, etc.) were represented exactly twice. The experiment was a repeated measures experiment where the number of UAVs is the between subjects factor and the mode of operation is the within subject factor.

The data was analyzed using the statistical software package R [19]. First the data was checked for homogeneity in variances to make sure there were no additive effects from any of the factors or from interactions of the factors. Analysis of Variance (ANOVA) was used to see if any factor had a statistically significant effect on the results, and if it showed an effect Tukey Honest Significant Difference (HSD) pair-wise tests were used to see which means were significantly different. The subjective survey questions were answered on a discrete scale and so do not fit the assumptions of ANOVA. The Wilcoxon Signed Rank and Wilcoxon Rank Sum non-parametric tests were used to analyze the subjective data. All statistical tests were performed at the $\alpha = 0.05$ significance level. Apart from statistical tests, detailed in Appendix F, the data was visualized with box plots, and in some cases scatter plots with linear regression to demonstrate effects on the data.

4.2 Answering the Research Questions

This section analyzes data from the experiment to answer the four main research questions that were presented in Chapter 1.

4.2.1 Cognitive Load on the Operator

Research Question #1: Does collaborating with an RRT algorithm in path planning decrease the cognitive load of the operator over manual path planning?

For this question, the measurements to consider for the three modes are the percentage of the secondary task questions answered and the average response time to secondary tasks for the queries answered. Response times were calculated by reading the event log of each participant, and for each question the participant answered, subtracting the time that question appeared from the time the person answered it. The percentage of secondary tasks answered was calculated for each participant by finding the ratio of the number of questions the participant answered to the number of questions that appeared during the mission.

The boxplots below show the average response times and percentages of secondary tasks answered for missions in the three modes. The boxplots plot mode against response times in seconds and against percentages, respectively.

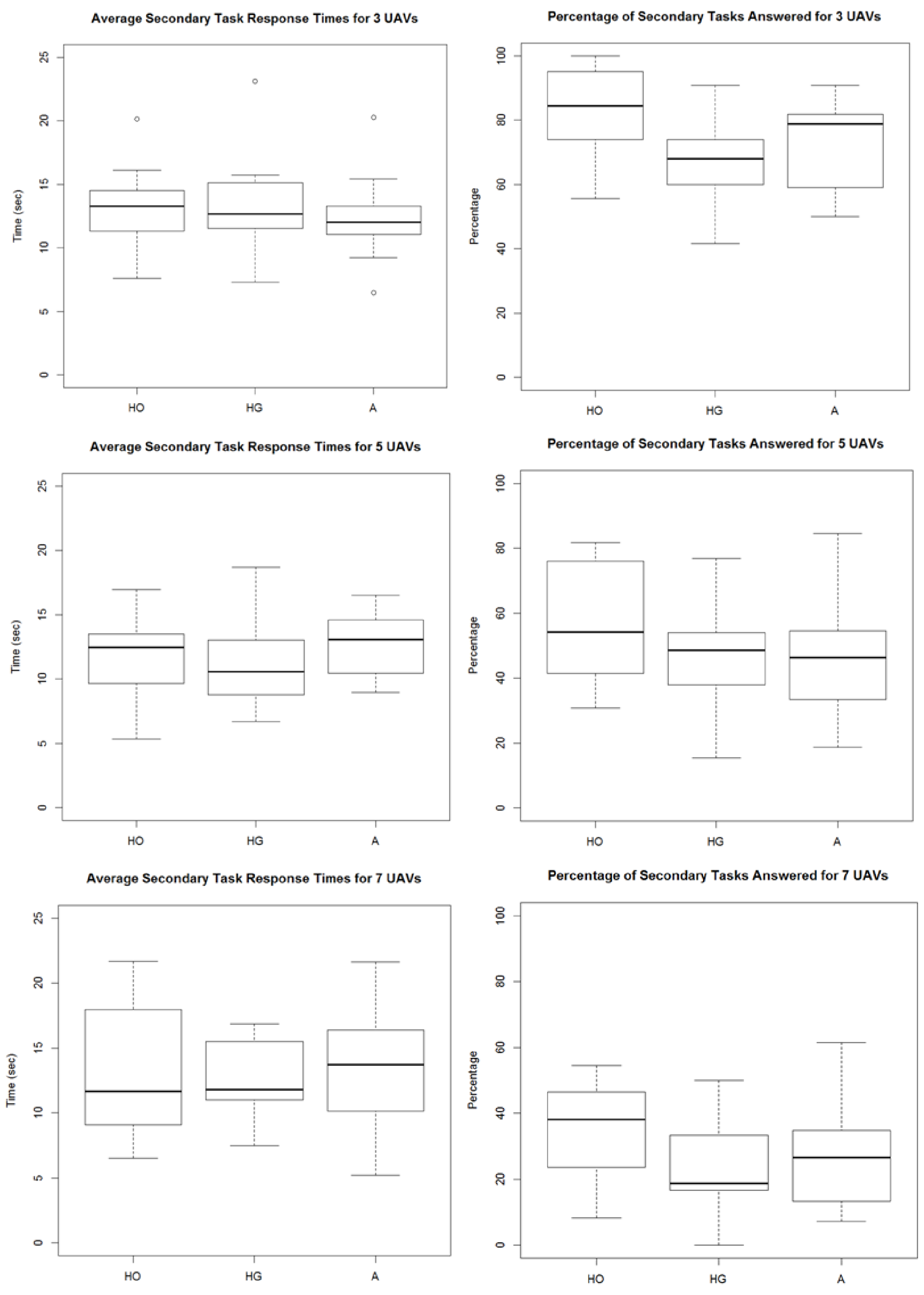


Figure 4.1: Secondary Task Response Times and Percentage of Secondary Tasks Answered for 3, 5, and 7 UAVs

The repeated measures two-way ANOVA did not show a statistically significant impact of either the mode of operation or the number of UAVs on the response time. The effects of the number of UAVs and the mode of operation had p-values of 0.31 and 0.9, respectively. When people did respond to a question, it usually took them around the same amount of time regardless of mode or number of UAVs. This may have been caused partly by the relatively frequent change of questions. New questions appeared on average every 30 seconds, and this does not allow for a wide variation in response times.

However, repeated measures two-way ANOVA did show a statistically significant impact of both mode ($F(2, 99) = 6.25$, $p = 0.0028$) and number of UAVs ($F(2, 99) = 75.63$, $p < 0.0001$) on the percentage of secondary tasks answered by a participant. It did not show a significant interaction between the two independent variables. ANOVA and Tukey HSD test results on the data are presented in Tables F.1 and F.2 in Appendix F.

The results of the tests show that participants who controlled 3 UAVs answered significantly more questions than participants who controlled 5 UAVs, who answered significantly more questions than participants who controlled 7 UAVs. This makes sense; the more UAVs operators must pay attention to, the higher their cognitive load.

The results also show that participants answered more questions in human only mode than in either of the modes involving RRT; human guided mode and automation mode did not have statistically different results. This means that according to this measure, operators had a lower cognitive load when operating in human only mode than when operating in either human guided or automation mode; adding c-RRT interaction increased the operators' cognitive load. Discussion on possible explanations will follow in the response to research question #3.

4.2.2 RRT Impact on Performance

Research Question #2: Does either mode involving automation result in better performance, where performance is measured by mission completion time and obstacle avoidance, than human-only mode? Which mode involving automation performs better?

For this question, the dependent variables involved measure mission completion times, obstacle avoidance, and algorithm runtime: total mission completion time, average mission completion time, number of obstacle incursions normalized for the number of UAVs, total time spent inside obstacles normalized for the number of UAVs, average RRT runtime, and number of RRT failures per UAV. All measurements were extracted from the experiment data logs. Total mission completion time is defined as the time it takes for the last UAV to reach its last target; average mission completion time is the average time over the number of UAVs for a UAV to reach its final target in a mission; the number of obstacle incursions per UAV is the number of times that some UAV flies through an obstacle during a mission, divided by the number of UAVs; the total time spent inside obstacles per UAV is the time in seconds that a UAV spent flying through an obstacle, summed over all UAVs, and then divided by the number of UAVs. Average RRT runtime is the average time that it takes for RRT to output a path after starting a search during a mission. The number of RRT failures per UAV is the number of times during a mission that the algorithm fails to find a path within its time limit, divided by the number of UAVs. The boxplots below show the data collected for mission completion time variables plotted against mode of operation for 3, 5, and 7 UAVs.

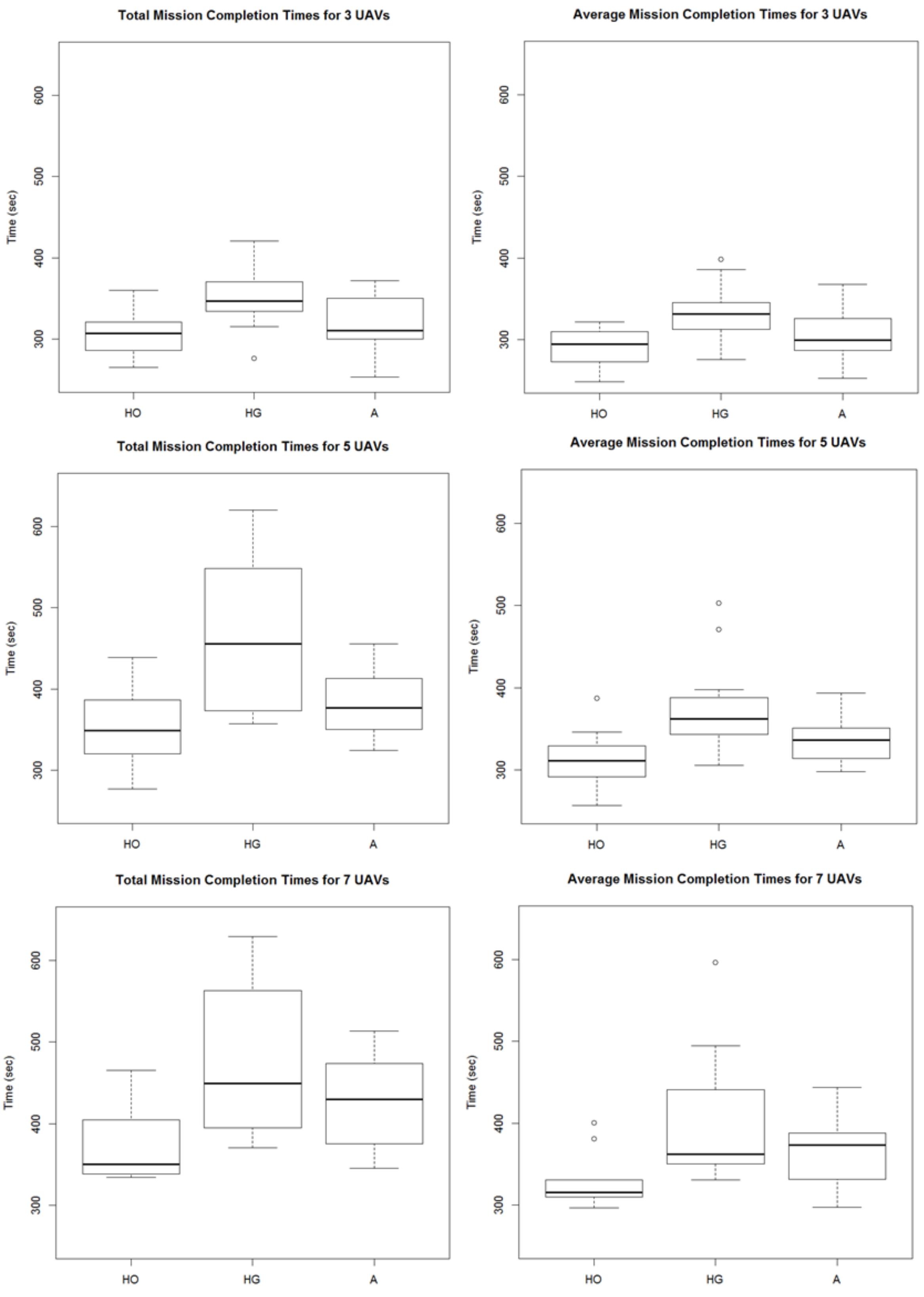


Figure 4.2: Total and Average Mission Completion Times for 3, 5, and 7 UAVs

Total mission completion time and average mission completion time have the same results: both the mode of operation and the number of UAVs affects mission time. The two factors do not have a statistically significant interaction. Supportive statistics can be found in Appendix F. For both variables, missions with 3 UAVs are completed faster than missions with 5 or 7 UAVs; missions with 5 UAVs and missions with 7 UAVs take statistically approximately the same amount of time. There is a statistically significant difference between every pair of modes, however the distinction between human only mode and automation mode is only marginally statistically significant, with a p-value of 0.046. Human only mode performs the best in terms of completion time; automation mode has second-highest performance, and human guided mode is significantly worse than the other two modes.

The boxplots present these results visually. In the missions with 3 UAVs and 5 UAVs, human only mode and automation mode have relatively close results, while human guided mode is significantly higher. In missions with 7 UAVs, automation mode is in between human only and human guided modes, but it looks clearly distinct from human only mode.

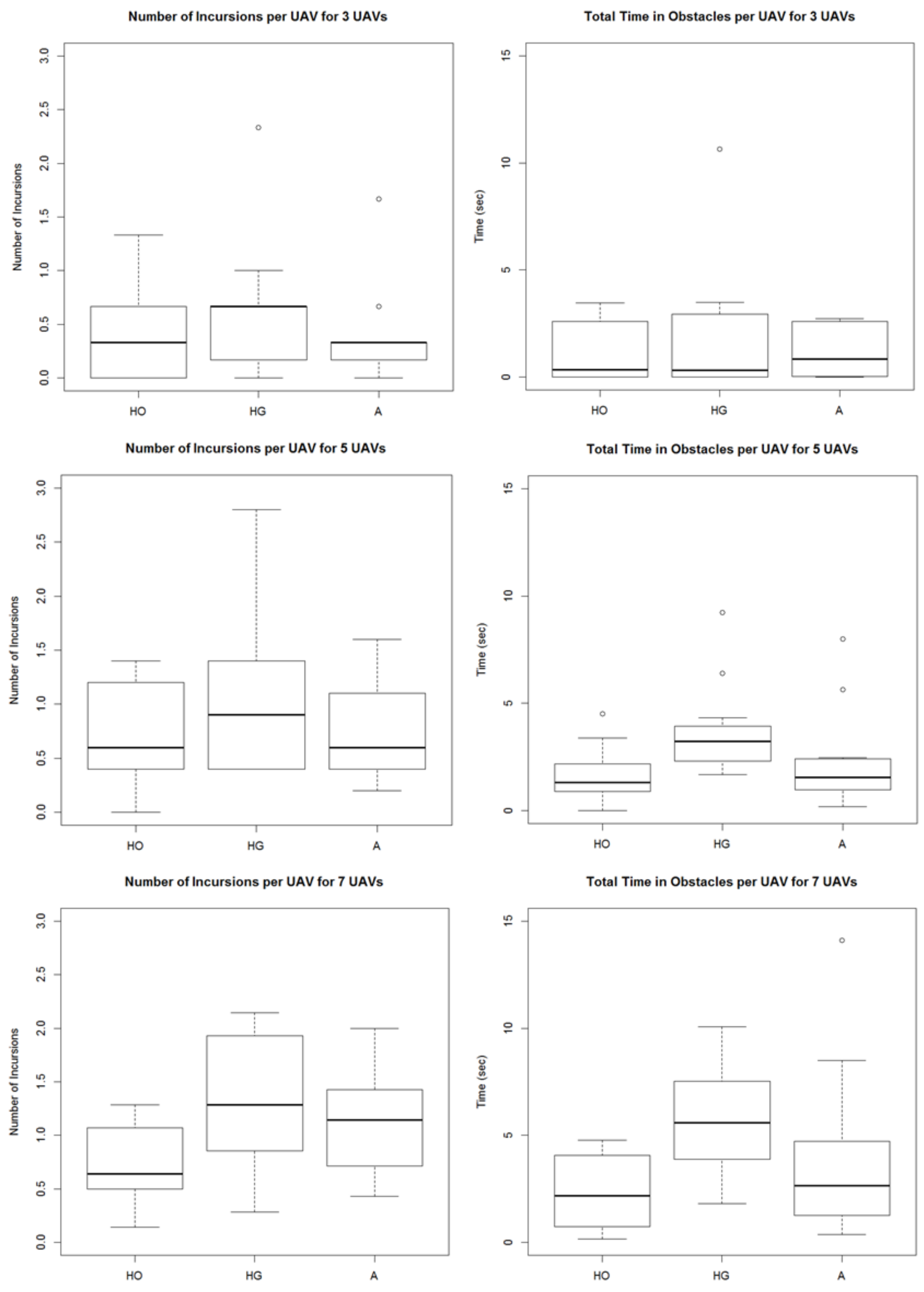


Figure 4.3: Number of Incursions per UAV and Total Time in Obstacles per UAV for 3, 5, and 7 UAVs

The second set of variables measuring performance involves obstacle avoidance: number of obstacle incursions per UAV and total time spent inside obstacles per UAV. Repeated measures ANOVA (Table F.7 in Appendix F) showed that the number of UAVs had a significant effect on the number of incursions per UAV ($F(2, 99) = 11.2$, $p < 0.0001$). Tukey HSD tests in Table F.8 in Appendix F show that missions with 3 UAVs have fewer incursions per UAV than missions with either 5 or 7 UAVs, while with 5 or 7 UAVs the number of incursions per UAV is approximately the same. Missions with 3 UAVs had an average of 1.42 total incursions; missions with 5 UAVs had an average of 4.25; missions with 7 UAVs had an average of 7.39. However, if the number of incursions is normalized for the number of UAVs, the average number of incursions per vehicle is 0.47 for 3 UAVs, 0.85 for 5 UAVs, and 1.06 for 7 UAVs. It makes sense intuitively that as the number of UAVs increases the number of incursions per UAV should also increase. There is more for operators to keep track of, and therefore it becomes harder for them to avoid every obstacle. Also, the probability of an obstacle appearing too close to a UAV to have time to avoid increases with the number of UAVs.

Repeated measures ANOVA also showed that the mode of operation had a significant effect on the number of incursions per UAV ($F(2, 99) = 5.23$, $p = 0.007$). Tukey HSD tests show that automation mode is indistinguishable from human only mode when comparing the number of obstacle incursions per UAV; human guided mode *is* distinguishable from human only mode, and performs significantly worse. Looking at the boxplots, with 3 UAVs and with 5 UAVs, human only and automation modes are very similar while human guided mode is significantly higher. With 7 UAVs, automation mode has more incursions than human only mode, but human guided is again the highest.

The reason for the relatively poor performance of human guided mode is that it does not allow operators to manipulate paths manually; they can only change paths by drawing wayareas and requesting an RRT solution. If the necessary modification is small, the interaction of drawing a wayarea and clicking the "Generate Path" button takes longer than clicking once to redirect the path with a waypoint. In addition, the algorithm is randomized and the time it requires to find a solution varies. Sometimes it finds a path very quickly and with little or no input from the operator, but other times it may go almost to its maximum time limit until it outputs a path, or it may not find a path at all, in which case the current path – with obstacle incursions – is kept. Because operators have no manual manipulation capabilities, while the algorithm is in the process of looking for a solution for a UAV, operators have no control over that vehicle. As a result, many incursions happened in human guided mode because the RRT algorithm was not quick enough and the time limit was too long.

Results for the total time spent inside obstacles per UAV are shown in Tables F.9 and F.10 in Appendix F. Repeated measures ANOVA showed that the number of UAVs had a significant effect on this variable (F(2, 99) = 10.63, p < 0.0001), and so did the mode of operation (F(2, 99) = 6.57, p = 0.0002). The average time spent inside obstacles per UAV is 1.38 seconds for 3 UAVs, 2.56 seconds for 5 UAVs, and 3.93 seconds for 7 UAVs. According to Tukey HSD test results, the difference between 3 UAVs and 5 UAVs is statistically insignificant but near the threshold of significance, while the difference between 3 UAVs and 7 UAVs is significant, as is the difference between 5 UAVs and 7 UAVs. The explanation of the results is similar to that given for the number of incursions. Human only mode is again indistinguishable from automation mode, but not from human guided mode. Automation mode is also indistinguishable from human guided mode, but just barely, with a p-value of 0.059.

The boxplots show all three modes having comparable results for total time spent inside obstacles per UAV with 3 UAVs. With 5 UAVs human only and automation modes are comparable, while human guided is significantly higher. With 7 UAVs, human only and automation modes are again similar, while human guided mode is even more significantly higher. With both 5 UAVs and 7 UAVs, the average time spent inside obstacles per UAV for human guided mode is a little more than twice the average for human only mode and a little less than twice the average for automation mode.

The third category of performance metrics involves the performance of the RRT algorithm in human guided and automation modes. The average algorithm runtime and the number of failures per UAV were measured for each mission in these modes. Figure 4.4 displays the results.

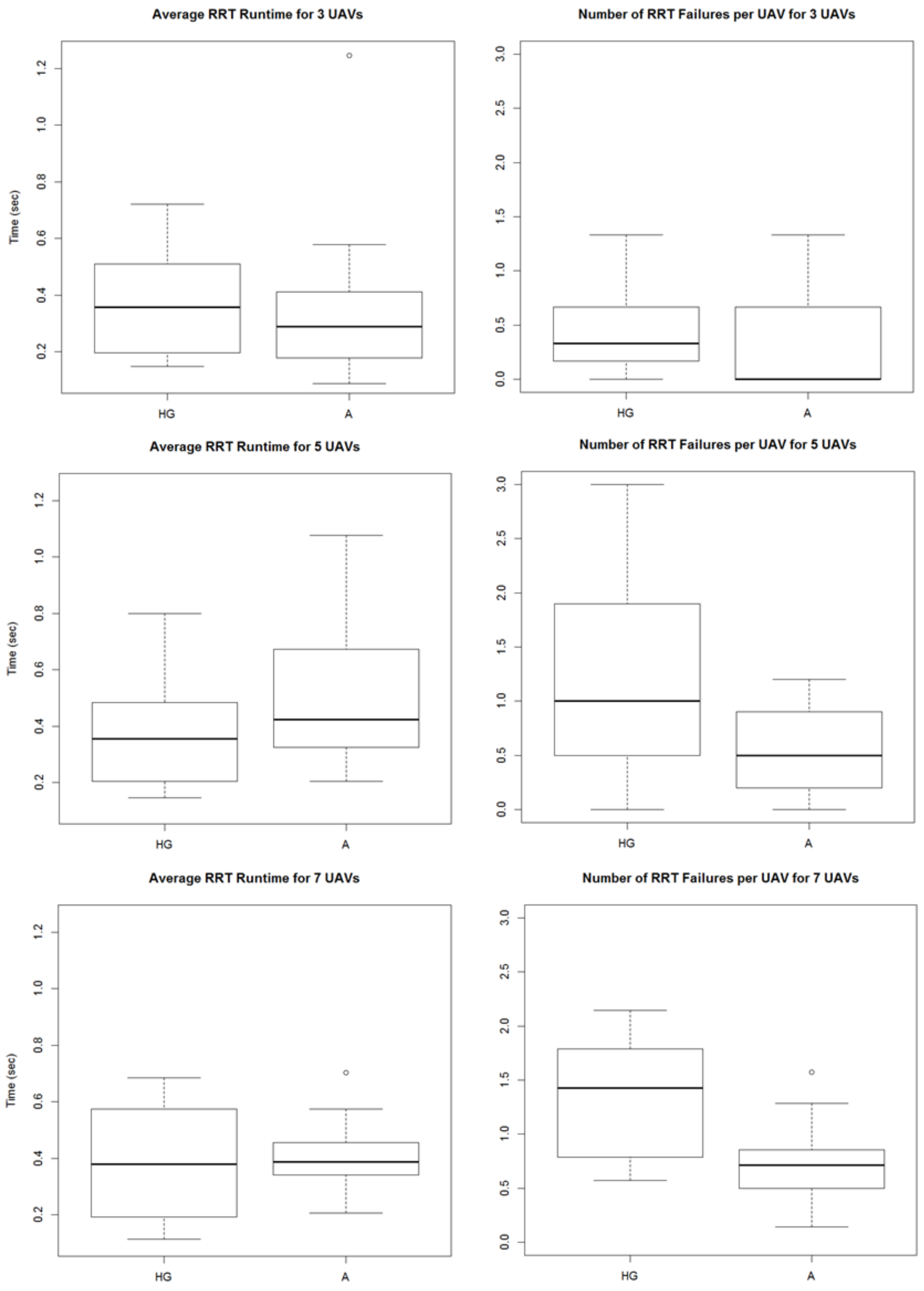


Figure 4.4: Average RRT Runtime and Number of RRT Failures per UAV for 3,5, and 7 UAVs

Repeated Measures ANOVA did not show a significant effect of either the number of UAVs or the mode of operation on the RRT runtime. The p-values were 0.49 and 0.31, respectively. These results signify that in both human guided mode an automation mode, when the algorithm succeeded at finding a path, the runtime was statistically indistinguishable. This means that allowing the operator to input wayareas did not improve algorithm runtime over fully automated path searching.

Tables F.11 and F.12 in Appendix F show the ANOVA and Tukey HSD results for the number of RRT failures per UAV. Repeated Measures ANOVA showed that both the number of UAVs ($F(2, 66) = 8.93$, $p < 0.001$) and the mode of operation ($F(1, 66) = 11.56$, $p = 0.0012$) had an impact on the number of RRT failures normalized for the number of UAVs. Tukey HSD tests demonstrated that the difference in failures is insignificant between missions with 5 UAVs and missions with 7 UAVs, but is significant for the other pair-wise comparisons. Human guided mode has significantly more failures per UAV than automation mode. This shows that human input into the algorithm actually did not help its performance, but rather made it worse. Possible reasons will be discussed in the next section.

4.2.3 Subjective Feedback

Research Question #3: How do human operators prefer to interact with RRTs to plan paths? How does using RRTs affect operators' perception of their performance?

For this question, the quantitative variables are the participants' responses to the survey questions asking them to rate their subjective performance and frustration levels after every mission on a 5-level scale. In the plots below, the higher the number, the better the rating. For frustration level, "5" represents "none", "4" represents "low", "3" represents "medium", "2" represents "high", and "1" represents "very high". For performance level, "5" represents "very good", "4" represents "good", "3" represents "ok", "2" represents "poor", and "1" represents "very poor". Table F.13 in Appendix F shows the results of the non-parametric Wilcoxon Signed Rank and Wilcoxon Rank Sum tests on the responses.

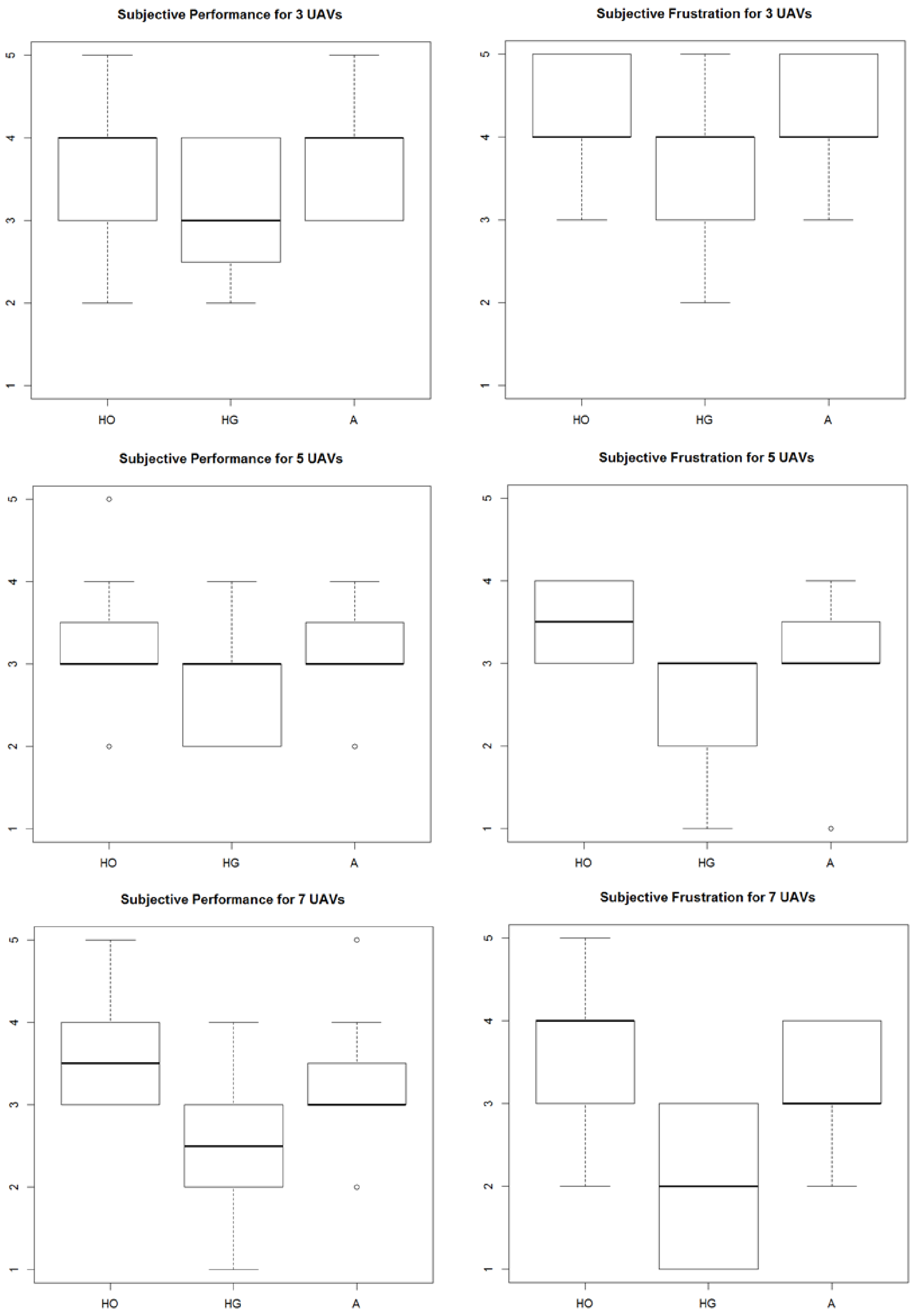


Figure 4.5: Subjective Performance and Frustration for 3, 5, and 7 UAVs

For subjective self-rated performance, the results of the Wilcoxon Signed Rank and Wilcoxon Rank Sum tests show that there is no significant difference between human only mode and automation mode (V = 102.5, p = 0.43), but participants perceived a worse performance in human guided mode. For HO vs. HG, V = 229, p = 0.00057; for HG vs. A, V = 20, p = 0.00019. These results are compatible with the objective performance metrics. The tests show a significant difference between 3 UAVs and the two other numbers of UAVs. Participants who controlled 3 UAVs gave themselves a better assessment than participants who controlled either 5 or 7; the participants who had 5 or 7 UAVs had statistically indistinguishable perceived performance rates.

Participants found human only mode to be the least frustrating, followed by automation mode, followed by human guided mode. Human only mode vs. automation mode is just over the threshold of statistical significance (V = 94, p = 0.043). When human only results are tested against automation results directly, for a given number of UAVs, the two groups are not found to have a statistical difference for any number of UAVs. These results are presented in Table F.13 in Appendix F.

Feedback from Participants

In addition to the two surveys, participants were asked several questions after the experiment to gain feedback and insight into their experience. The questions and common responses are summarized below. Full responses can be found in Appendix G.

Question 1: What was your favorite mode? How would you rate the modes in order of preference?

Out of the 36 participants, a total of 11 preferred automation mode, 21 preferred human only mode, 2 tied human only and automation modes in first place, and only one preferred human guided mode. Table 4.1 shows the breakdown of favorite modes for different numbers of UAVs. With 3 UAVs and 7 UAVs, approximately half the participants preferred human only mode and approximately half preferred automation, however with 5 UAVs, the vast majority of participants preferred human only mode. Almost all participants did not like human guided mode. Out of the 36 test users, there were only 5 who did not put human guided mode in last place in their order of preference. Most people who preferred human-only mentioned that they liked automation almost as much. The full set of feedback is listed in Appendix G but the key points are summarized below.

Table 4.1: Favorite Modes for Different Numbers of UAVs

	HO	A	Tie HO-A	HG
3 UAVs	6	5	1	0
5 UAVs	9	2	0	1
7 UAVs	6	4	2	0

Human Only Mode

A lot of people who liked human only mode mentioned that in this mode they felt they had the most control. They liked that they could predict the paths that would be made; they did not like the unpredictable and inefficient nature of the computer's output paths. Some people commented that they are used to micromanaging in strategy games, such as Starcraft, and so preferred human only because it gave them more ability to micromanage. Another important reason was that people felt this mode was faster than the others, because in the modes involving RRT, operators would occasionally have to wait while the algorithm searched for a path, while in human only a path would immediately appear because the path simply connected the waypoints. Several people mentioned that this mode was the most intuitive to them, and the easiest to learn.

Some participants said that because they were the ones who set the paths instead of allowing the computer to do it, they were "more aware of all the UAVs". Participants mentioned that in the modes involving RRT, the algorithm would generate longer paths than was necessary. People believed they were better at finding quick paths than the algorithm, and therefore preferred to plan the paths themselves. A common strategy was to click "Generate Path" without any waypoints in order to output a straight path, start the UAV moving, and then modify the path a minimal amount to avoid obstacles while trying to keep the path close to the straight-line trajectory. Compared to automation mode, people liked that human only involved less waypoints, and it was therefore easier to make large modifications to paths because fewer waypoints needed to be changed. Finally, an interesting comment by one participant was that in automation mode he could get the UAVs to start moving right away because no input was necessary, while in human only mode the start was slow; however, "once things got spaced out in human only I didn't have to deal with as many vehicles at once".

For the people who did not prefer human only mode, some of the complaints were that they had "a lot to keep track of", and that they "had to give so much detail". These people preferred one of the automation modes because the automation would do some of the work for them. For example, one participant mentioned that with 7 UAVs, in human only mode it was sometimes difficult to locate the target for a given UAV in order to input waypoints, whereas in the other modes she could simply click "Generate Path", and the RRT algorithm would know where the target was located (this participant did not use the strategy of outputting a straight line first and modifying it later).

Human Guided Mode

Most participants strongly disliked human guided mode; they thought it was unintuitive and frustrating. First of all, they did not like the loss of control over the other modes, which had waypoints for manual manipulation. They felt a lack of control when the output path went places they did not tell it to go. They were disappointed that when they replanned en route, the entire path would change instead of only the section that they were trying to modify. Replanning always required the use of the algorithm, which meant that sometimes it would take too long to find an alternate path and so it was hard to prevent obstacle incursions. People did not think human guided mode was responsive to their input. One participant said, "it's sad because it doesn't follow my suggestions". In general, participants stated that they preferred working with waypoints rather than wayareas.

Participants did not understand how wayareas work. Even though it was emphasized in the training slides that the algorithm will select a point inside each circle uniformly at random, participants expected that the paths would go through the centers of the wayareas and were surprised when sometimes the paths would pass through the edges. A lot of participants chose to draw very small wayareas to emulate waypoints and get more control over the solutions. People did not understand how best to use wayareas. There were times when it seemed no matter what input they tried, the algorithm would not find a solution even though they could clearly see a path.

Although it was suggested in the training slides to use wayareas to guide the UAV through narrow passages among obstacles, it was common for participants to put one wayarea before the narrow passage and one wayarea on the other side and expect the algorithm to discover the most difficult part of the path by itself. Because participants did not understand vehicle constraints well, they would occasionally place wayareas in such a way that it was practically impossible to find feasible paths through them, given the heading of the vehicle. This lack of understanding led to algorithm failures and operator frustration. If the algorithm failed to find a feasible solution, it would either output a straight line if the UAV did not already have a path, or keep the current path, in either case usually resulting in a path with collisions. Since failing to find a path would result in no changes (except if the UAV did not have a path), operators had the impression that their input was not being taken into consideration and felt frustrated. Another problem with the wayareas was that occasionally the participants would make tiny unintentional wayareas without realizing it and then would not understand why the output path kept going so far out of the way. The interface was programmed to ignore wayareas with a radius of 0 pixels, but making this threshold at least 5 pixels would have fixed the issue. However, this problem was not discovered until after the experiment began.

People mentioned that human guided mode did not work fast enough. First, it took participants more time to draw wayareas than to click waypoints. Second, unlike in automation mode where they could modify a path manually if the algorithm took too long to think, in this mode they were forced to wait for the algorithm to finish and sometimes, due to its randomized nature, it would take too long and slow them down. Participants said that they did not feel this delay was worth it; they thought since they are often forced to wait a few seconds anyway, and they do not even know whether the output path will be efficient, they might as well use that time to click waypoints and make a path that they know will be good. In general, people felt that the paths output in this mode were "strange". Several mentioned that they thought the paths output in human guided mode were worse than the paths output in automation mode. People thought that they could find better paths than the algorithm could, and it took them extra effort to correct the algorithm's poor suggestions, so they did not think its involvement was helpful.

The feedback given on human guided mode emphasizes the difference between the strengths of the operator and those of automation. People claimed that they could find better paths than the algorithm, however unlike the algorithm they did not take vehicle constraints into consideration, and when they tried to influence the algorithm sometimes their input would cause the algorithm to fail to find a path. It is interesting that several people had the impression that this mode produced worse paths than automation mode. The two modes used the exact same algorithm, the only difference being that human guided mode allowed human input into the solutions. Another reason why human guided mode was criticized for taking a long time is that the time limit for the algorithm to find a path depended on the number of input wayareas. Sometimes participants made unhelpful wayareas, and the more wayreas they input the longer it took for the algorithm to return with no solution.

The few people who did have positive feedback on human guided mode mentioned that it took care of a lot of the work and "made [the participants] not worry too much about what was going on". It involved less micromanaging than the automation mode, which for some participants was a negative feature, but for others it meant that they were less tempted to modify paths unnecessarily, and so they felt they had a smaller workload. One participant remarked that it was "the best of both worlds, with human guided you could give specifications to the algorithm, hey go here, it saved the time of connecting dots but would still come up with good paths avoiding obstacles". Some participants liked that compared to human only mode it provided collision-free default paths and they did not find editing them to be too difficult. A participant mentioned that it made path planning faster because the algorithm could find complex paths by itself, whereas in human only mode the participant would have had to click waypoints along the entire path.

Automation Mode

Many participants liked automation mode. In general, they thought it had a reasonable approach: the algorithm suggests paths, and the operator can modify them. People thought it was intuitive and easy to learn. They thought the paths were generally fairly good, and they were easy to modify. Participants were happy that they could use the algorithm to find feasible collision-free paths, but also had manual control with waypoints. They thought it made starting the UAVs fast and easy because they could simply click "Generate Path" to start each UAV on a collision-free path, and then tweak the paths as necessary. They also liked that compared to human only mode, they could click the "Generate Path" button multiple times and could choose between different paths. One participant said, " I like to see what the algorithm comes up with". Some participants mentioned that in automation mode, they spent less time preoccupied with the UAVs. One user noted, " it seemed to do a lot of the thinking for me, I just had to make little adjustments. It was just tweaking and looking for yellow to pop up."

The main complaint about automation mode was that it displayed too many waypoints. The waypoints meant that when operators wanted to make major changes to a path, they had to move or delete a lot of waypoints along the path. A lot of the users found this annoying. They said the computer would output fairly optimal paths, but they would have to unclick a lot of the waypoints in order to make the paths shorter. So even though the mode saved them work and time in generating the paths, it also created more work for them because of the extra waypoints. Some people noted that even though they realized the output paths were good and deleting waypoints would not improve them significantly, they still felt the need to delete the extra waypoints, sometimes to facilitate easier modifications in the future, sometimes because they had nothing else to do, sometimes because they felt it would help even if a little, and in one case, a participant deleted them simply because she did not like the way they looked: "extra waypoints annoyed my sense of aesthetics. It's a beautiful line with stuff in the middle". Many participants wished that they could have deleted many waypoints at a time. One participant summed up, "in automation mode it would generate really abnormal paths that were close enough that I didn't want to generate a whole new path, but not optimized enough, so I'd go in and delete several waypoints, and that took a lot of time." Apart from excess waypoints, a few participants complained about occasionally having to wait for the algorithm to output a path. A few others complained that the paths were suboptimal.

Question 2: What would you change in the interface?

Participants gave a lot of feedback on the interface. Several of the features that participants mentioned that they especially liked were the timeline, obstacles turning yellow to warn of collisions, and UAVs flashing orange to call the attention of the operator when they are waiting for a path to their next target. Many participants said the interface worked well, but there were a few common suggestions.

Many participants wanted some information to be conveyed more clearly. For example, several mentioned that it would have been helpful to point the UAV symbol in the direction it the vehicle was traveling in; this would have given them information about heading and helped them plan paths so that the UAVs would not need to turn around. Some people wanted the waypoints to be larger for easier modification. Several wanted more contrast in the colors of the interface, for example between the green of the UAVs in their normal state and the orange that they flashed when they were waiting for a path. Participants could not always tell whether a UAV was inside an obstacle when it appeared or not, and would have liked a visual cue distinguishing the two cases, because in the first they could ignore the obstacle, whereas in the second they would want to guide the UAV out of the obstacle as quickly as possible.

Some participants complained that it was occasionally hard to find the UAVs' targets. Some people mentioned that it was difficult to distinguish the paths; they were all black. When an obstacle appeared, if it overlapped with some paths they would have to disentangle the paths to figure out which path corresponded to which UAV. To help with this issue, several people said it would have been helpful to be able to select paths, and not only UAVs. Some participants commented that in addition to flashing orange, it would have been nice to have a queue of UAVs in need of a path. They did not like that UAVs disappeared from the timeline when they reached their targets. One person mentioned that he would have liked to be able to select UAVs directly in the timeline.

Several people mentioned that they kept missing secondary task questions, so they would have liked the secondary task panel to flash brighter when a question appeared, or to issue an auditory "ding" in order to attract their attention. Some participants mentioned similar feedback about obstacle warnings; sometimes the users did not notice when obstacles turned yellow, and would have liked the obstacles to either blink or issue auditory warnings.

Some of the feedback centered on improving the efficiency of the interface. Many users wanted keyboard shortcuts for selecting UAVs, generating paths and confirming, and switching to the secondary task panel text box. A couple of participants did not like having the details panel on the side, and would have preferred to move that information to the center of the interface, near the timeline. One user said that he would want the UAVs to start moving immediately at the start of each mission and upon reaching a target. He would want them to move in a straight line in human only mode, and to automatically generate paths in the other two modes.

The final category of comments regarded the functionality of the interface. One suggestion made was for the RRT modes to present several possible paths simultaneously and allow the operator to pick among them. Another suggestion was to allow the operator to choose whether to generate a path using the RRT algorithm, or to manually input a path. To address the occasional long wait times, a suggestion was made to allow the operator to stop the algorithm if it was taking too long, and instead plan a path manually. To address the issue of having too many waypoints in automation mode, one idea was to allow the operator to set the number of waypoints or the interval at which they are displayed. Another suggestion was to allow the operator to delete many waypoints at once, for example by clicking and dragging a selection box. One participant said that it was annoying when secondary task questions changed while he was typing an answer, so he would like the interface to prevent the current question from being replaced with a new question while there is text in the text box.

Discussion

It is interesting to note that two of the suggestions made – for the interface to have keyboard shortcuts and for the UAVs to start moving immediately – were features of the previous version of the interface that were found to be unsuccessful [6] and therefore eliminated from the current version. Participants in the previous experiment had difficulty remembering keyboard shortcuts because shortcuts are efficient but have poor visibility and learnability. Participants found the immediate simultaneous start of the UAVs to be overwhelming.

In principle, having keyboard shortcuts in addition to a point-and-click interface would improve efficiency, but for the purposes of this experiment they would add an additional unwanted independent variable. There would be a higher variation in mission completion times because some people would prefer to use the shortcuts, while others would not use them.

Question 3: Did you have enough training?

Almost all participants said that they had enough training. The ones who did not explained that they would have liked to have more training time in human guided mode. They did not feel they had a good grasp of the capabilities of that mode after training. They felt they had not learned how to use the wayareas well.

Question 4: Do you have video game experience?

Out of the 36 participants, 9 admitted to having very little or no experience with video games. The rest mentioned at least some experience with games such as strategy games, first person or third person shooters, fighting games, role playing games, and several other genres of games. Many people mentioned that they have experience with the popular real-time strategy game Starcraft, which involves building and manipulating armies to destroy the opponents' armies. An interesting observation was that nearly all Starcraft players preferred human only mode in the experiment. 71% of participants who mentioned that they have experience with Starcraft preferred human only mode, and the rest tied human only and automation in first place. (One participant mentioned that he would rate automation first for usability, but human only for fun.) Starcraft involves a lot of micromanaging and very little automation, so people who are used to playing this game are accustomed to having a lot of control, and therefore prefer the mode that gives them as much control as possible.

Question 5: What field are you in?

Of the 36 participants, nearly all were in technical fields. There were 10 people in electrical engineering and computer science, 10 in aeronautics and astronautics, 6 in human factors, 5 in mechanical engineering, and one person in each of materials science and engineering, comparative media studies, linguistics, math education, and physics.

4.2.4 Increasing the Number of UAVs

Research Question #4: How does RRT path planning impact performance and subjective assessment as the number of UAVs increases?

The first research question was whether or not using c-RRT reduces the cognitive load on the operator. The first variable, secondary task response time, was found to stay fairly constant for all numbers of UAVs. The second variable, percentage of secondary task questions answered, decreased with the number of UAVs. This means that as the number of UAVs increases, the cognitive load on the operator increases also, which is as expected. For all numbers of UAVs, human only mode performed best in this category, while human guided and automation modes performed equivalently. Looking at the box plots, it seems that automation mode performed marginally better than human guided for 3 UAVs and 7 UAVs, while with 5 UAVs the two modes have very similar performances.

The second research question was whether or not using c-RRT improves performance in terms of mission completion time and obstacle avoidance. For both total and average mission completion times, the number of UAVs affected mission time: missions with 3 UAVs took the least amount of time, while missions with 5 UAVs and missions with 7 UAVs took approximately the same amount of time. The mean for total mission time with 3 UAVs was 324.4 seconds, for 5 UAVs it was 398.5 seconds, and for 7 UAVs it was 425 seconds. The reason for this may be correlated with algorithm failures. Mission time is largely determined by the speed of the UAVs. Since all UAVs move in parallel, mission time is the time that it takes for the UAV with the longest total distance to travel to reach its final target, with the addition of some extra time. The extra time is due to interactions, RRT runtime, and bad paths. While the runtime of the algorithm when it is successful stays constant with increasing numbers of UAVs, the number of RRT failures was found to increase between 3 UAVs and 5 UAVs, but did not differ significantly between 5 UAVs and 7 UAVs. If the algorithm fails to find a path, this means that it ran until its limit and was unsuccessful. An algorithm failure slows down operators because it forces them to wait a significant amount of time but produces no results. RRT failures are especially detrimental in human guided mode, because in this mode users cannot simply plan paths manually if the algorithm fails.

The box plots reveal that for both total and average mission completion time, human only is the best mode, automation is slightly worse than human only, and human guided is by far the worst. As the number of UAVs increases, the difference between human only and automation becomes more apparent. With 3 UAVs it is barely distinguishable, while with 7 it is fairly obvious. Appendix F shows Tukey HSD values for the different modes and numbers of UAVs, and the p-values for human only vs. automation decrease with the number of UAVs. This pattern conveys that as the number of UAVs increases, the gap in performance between human only and automation also increases.

The other performance variables were the number of obstacle incursions per UAV and the total time spent inside obstacles per UAV. As the number of UAVs increases, so does the number of collisions and the time spent inside obstacles. For the number of incursions, the missions with 3 UAVs had the fewest incursions per UAV, while missions with 5 UAVs and 7 UAVs were found to be statistically indistinguishable. For the time spent inside obstacles per UAV, missions with 7 UAVs had the highest values while missions with 3 UAVs and 5 UAVs were indistinguishable. However, for both variables, even the groups that were not found to be statistically different were close to the threshold for statistical significance, with p-values of 0.23 and 0.09, respectively. For both variables, human only mode was indistinguishable from automation but distinguishable from human guided. For the number of incursions, for 3 UAVs and 5 UAVs, the automation results look very similar to human only results, while human guided has more incursions. For 7 UAVs, automation is clearly in between human only and human guided. For total time inside obstacles, all modes spend an approximately equal amount of time inside obstacles with 3 UAVs, but with 5 and 7, human only and automation are very close, while human guided is clearly significantly worse, with about twice as much time spent in obstacles per UAV as the other two modes.

Human guided mode was found to be the least successful at avoiding obstacles because it relied entirely on the RRT algorithm for finding paths. Since the RRT algorithm is randomized and sometimes takes some time to find paths or return unsuccessfully, this delay compared to manual manipulation meant that UAVs sometimes either could not avoid obstacles in time or spent extra time inside obstacles while the algorithm looked for a way out. The average runtime of the RRT algorithm was 0.4 seconds when it succeeded, and 10 seconds when it did not. The problem was generally caused by the 10 second wait time when the algorithm could not find a path. As the number of UAVs increased, so did the relative number of incursions and time inside obstacles for human guided mode, because the more UAVs there are, the higher the chance that an obstacle will appear on a path very close to some vehicle and will cause the need for immediate replanning. Human only and automation modes performed similarly because with both modes, the operator could simply click waypoints to immediately redirect a UAV.

The third research question was whether using RRT affects operators' subjective assessment of their performance, and how operators prefer to interact with RRT. People who controlled 3 UAVs perceived the best performance, while people with 5 or 7 perceived that they did about the same. Human only and automation modes do not have a statistically significant difference, while human guided has significantly worse perceived performance. The box plots show similar data visualizations for human only and automation for all three numbers of UAVs, however the biggest difference is visible with 7 UAVs; automation seems to have a slightly worse self-assessment with more UAVs. For frustration, 5 and 7 UAVs are again indistinguishable, while 3 is significantly better than the others. All the modes are distinguishable, with human only incurring the least frustration, followed by automation, and with human guided causing significantly more frustration. As the number of UAVs increases, the difference between human only and automation becomes more apparent, as shown by the boxplots in Figure 4.5 and the p-values in Table F.13: 1 for 3 UAVs, 0.24 for 5 UAVs, and 0.1 for 7 UAVs..

Subjective performance and frustration correspond fairly well to the objective performance metrics. As the number of UAVs increases, however, so does frustration in automation mode relative to human only mode. The main reason for this is the number of waypoints displayed along paths in automation mode; this was the most common negative feedback comment on automation mode. Many people found the waypoints annoying and spent a significant fraction of their time deleting the displayed waypoints. As the number of UAVs increases, so does the number of paths and hence the number of waypoints to delete. With 3 UAVs, the number of available waypoints is only slightly more than the waypoints created by the operators in human only mode, but as the number of UAVs increases, so does the effort to delete extra waypoints.

Out of the 36 participants about a third preferred automation mode and 2/3 preferred human only; however, most people said both human only and automation mode worked well. Most people strongly disliked human guided mode. For 3 UAVs and 7 UAVs, the human only/automation split is about half/half. For 5 UAVs, the majority preferred human only mode. A reason for this may be that when there are 3 UAVs, automation mode does not display significantly more waypoints (there are only three paths) and therefore does not cause much extra work or frustration for the operator. The operator can enjoy the benefits of ready-made feasible collision-free paths and take only a little time to modify them when necessary. With 5 UAVs, there are many extra waypoints and operators are not busy enough that they will ignore them: they try to delete all the waypoints and this takes a lot of time and effort, and so most people with 5 UAVs preferred human only mode. With 7 UAVs, there are even more extra waypoints but the operators have more work to do, so they do not have time to modify all paths and often if a path is feasible, choose to leave it as is and take advantage of the algorithm.

Chapter 5: Observations and Additional Questions

While running the experiment and collecting feedback, there were several interesting observations made that led to additional research questions, and potential research questions for future experiments. This chapter presents these observations and results.

One such observation is that sometimes people planned paths for UAVs starting with UAV 1, then UAV 2, and so on, in increasing numerical order, instead of starting with the UAV farthest from its goal, which would be more rational. This implies that our brains could be accustomed to paying attention to numerical order and instinctively sort tasks in this way. This behavior leads to the following question: would operators perform better if UAVs were numbered depending on their distance from their target or had IDs that were not obviously ordinal, such as shapes?

Another observation is that people who play video games that involve a lot of micromanagement, such as Starcraft, prefer human only mode because it gives them the most control. Human guided mode frustrates them because they do not have manual control over the paths, and automation mode frustrates them because they spend a lot of their time deleting extra waypoints. For example, one participant's response to why he prefers human only mode was, "It felt more natural, I'm used to micromanaging things". His answer to the question about whether he has video game experience was, "yes, lots of real-time strategy games like Starcraft, they involved lots of micromanaging". This type of participant feedback leads to the following question: how does video game experience influence operator performance and preferences when interacting with RRT?

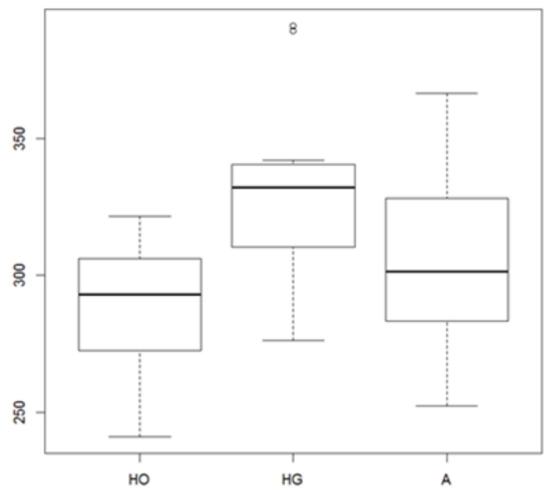
There are some concepts that humans do not understand very well. For example, it was hard for participants to understand vehicle constraints. They were explained in the training slides, but participants were surprised during planning or replanning when the output paths sometimes made wide turns or went in the wrong direction for a while so the vehicle could turn around. In one of the missions, a UAV's target was located in a narrow alley between two obstacles. The fastest path to its next target seemed to be to go north, however because the vehicle came from the north it needed to turn around and it had no space to do that in the alley without colliding with an obstacle. Many participants struggled with this problem for a long time. Human guided mode performed especially badly in this case, because participants repeatedly placed wayareas in the northern part of the map and the algorithm of course could not find a collision-free path through them. Another concept not immediately obvious to human operators is that in human only mode, generating a path will connect the input waypoints with a (Dubins constrained) straight line. Some participants would make waypoints at regular intervals along their desired trajectory, instead of only placing them at turning points.

There is more to remember and understand when using modes with RRT than in human only mode, such as the fact that the RRT algorithm is very good at finding straight paths when they exist, or that when there is a path through a narrow passage, wayareas should be placed inside the passage to help the algorithm navigate it. A few people forgot that they could press the "Generate Path" button multiple times for new paths. Many people only started learning to use these tricks toward the end of the missions, and they are quite important for better efficiency and lower frustration. From the researcher's experience, it took different amounts of time to adjust to each mode, and learn to use each mode efficiently. This phenomenon leads to the question: how does the performance of modes with RRT relative to human only mode change as training time or the intensity of training increases?

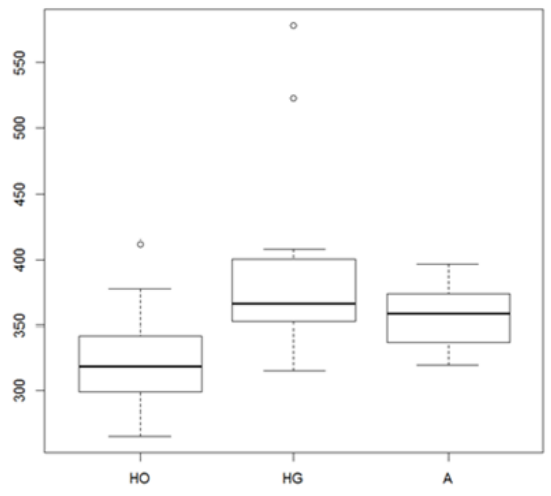
To see if there was a learning effect during the experiment, the results for the performance metrics were reanalyzed for only the third mission and compared to the overall results. The supportive statistics tables can be found in Appendix F. For total mission completion time, the only difference was that in the third mission, human guided mode and automation mode were not found to be statistically distinguishable, whereas for all missions they were distinguishable. This same change was true for average mission completion time, with the additional change that for only the third mission, ANOVA did not show an impact of the number of UAVs on average completion time. The results for the third mission did not show a significant effect of either the number of UAVs or the mode on the number of obstacle incursions per UAV, whereas for all the missions both variables had a significant effect. Repeated measures ANOVA did not show a significant impact of the mode on the time inside obstacles per UAV for only the third mission, while for all missions there was a statistical difference between human only and human guided modes. In addition, ANOVA did not show a distinction between missions with 5 UAVs and missions with 7 UAVs for the third mission, while for all missions these were distinguishable. There was no significant impact of either the number of UAVs or the mode of operation on RRT runtime in the third mission; these results were the same for all missions. Results for the number of failures per UAV were similar in both analyses as well, except that the difference between 7 UAVs and 3 UAVs and that between 7 UAVs and 5 UAVs was much smaller for the third mission than it was for all missions. Examining results for only the third mission did not provide evidence of a learning effect, but this is not surprising because the missions were short enough that they did not allow operators to gain the necessary experience. In addition, the modes are fairly different, so using the interface in one mode does not teach an operator how to use the other modes well.

A possible reason for finding several effects to be insignificant for only the third mission that were significant for all missions is the smaller sample size. More experiments would have to be run to study the changes in these effects. The most important difference in the results was that automation and human guided modes had statistically indistinguishable total and average mission completion times. The mean of average mission completions time for human guided mode over all missions and numbers of UAVs was 369.5 seconds, while the mean of average mission completion times for automation mode over all missions and number of UAVs was 335.2 seconds. For only the third mission, the average for human guided was 351.1 seconds, and the average for automation was 347.3 seconds; automation performed significantly worse than usual, while human guided performed significantly better than usual. A possible reason for this is that having completed a mission in automation mode before starting a mission in human guided mode may have shown operators what RRT is capable of and helped them understand better how to help it in human guided mode. On the other hand, having had a frustrating experience with human guided mode before seeing automation mode may have led participants to try to gain as much control as possible and rely on manual replanning instead of using RRT, which may have increased their mission completion times. To back up this claim, the average of the ratio of waypoint deletions to "Generate Path" clicks in automation mode over all numbers of UAVs and all missions is 1.52. This average for only the third mission is 1.7. For the first mission, this average is the lowest: 1.25. At this point operators have neither become accustomed to manual control in human only mode nor had a frustrating experience with human guided mode. Repeated measures ANOVA did not show a significant difference between any of the three missions, but this may be because of a small sample size, and should be studied further. The significance of this behavior is discussed later in this section.

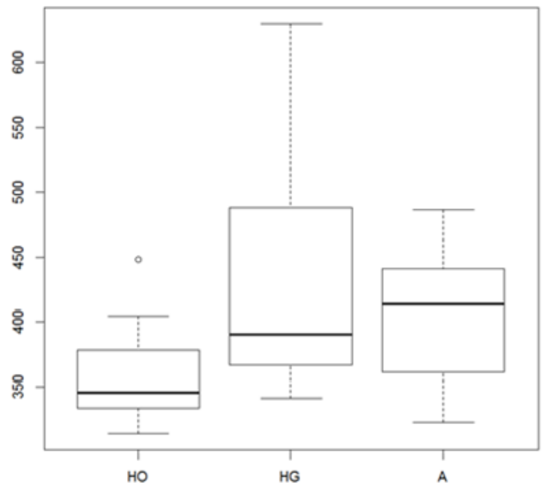
Another interesting behavior that was observed while proctoring the experiment was that fairly often during a mission, there was one UAV that finished significantly later than all the others. This would happen if the operator was busy with the other UAVs and kept neglecting a certain UAV unintentionally, not realizing that it was behind schedule relative to the others. This would happen more often with higher numbers of UAVs, because the more UAVs there are the harder it is to supervise each one equally. This seemed to occur more often in modes with RRT than in human only mode, which could be related to a participant's comment that it was easier to be aware of all the UAVs in human only mode, where the operator set the paths manually and had more control. This behavior happened most frequently in human guided mode, and the reason for this, apart from having less control, is that there were several scenarios in the missions where the paths from a UAV to its next target were nontrivial, such as in the case described earlier. Often in these situations, because of the participants' poor understanding of RRT and wayareas, their input only made it harder for the algorithm to find a feasible collision-free path. Participants would continue to press the "Generate Path" button, and the algorithm would continue to output straight-line paths due to failure. Participants did not understand how to help it find a good path, so they would either keep pressing the button or wait for the troublesome obstacles to disappear. As a result, that UAV would fall behind the others. Eventually, either the algorithm would succeed at finding a path, or the participant would give up and let the UAV travel through the obstacles. Figure 5.1 shows total mission completion times that ignore the last UAV for the different modes and numbers of UAVs.



3 UAVs



5 UAVs



7 UAVs

Figure 5.1: Total Mission Completion Times When the Last UAV is Ignored

The data presented in Figure 5.1 shows that the relative performance of human guided mode becomes better when one UAV is ignored, due to the behavior mentioned above. In fact, this difference becomes more apparent as the number of UAVs increases. With 3 UAVs, human guided mode and automation are still quite distinguishable. With 5 UAVs, they are much closer, and with 7 UAVs they are even closer. This shows that human guided mode suffered from some edge cases that negatively affected its performance and increased frustration. When those cases are ignored, human guided mode performs significantly better. Future iterations of this interface should take these edge cases into account and find a way to avoid them.

As mentioned previously, a common behavior in automation mode was to delete extra waypoints along paths. Because of this behavior, of the three modes of operation, automation mode generally had the highest interaction count, as conveyed by Figure 5.2. An interaction is any user action that is associated with manipulating the UAVs, and includes adding, deleting, or moving waypoints, adding or deleting wayareas, clicking "Generate Path" and clicking "Confirm".

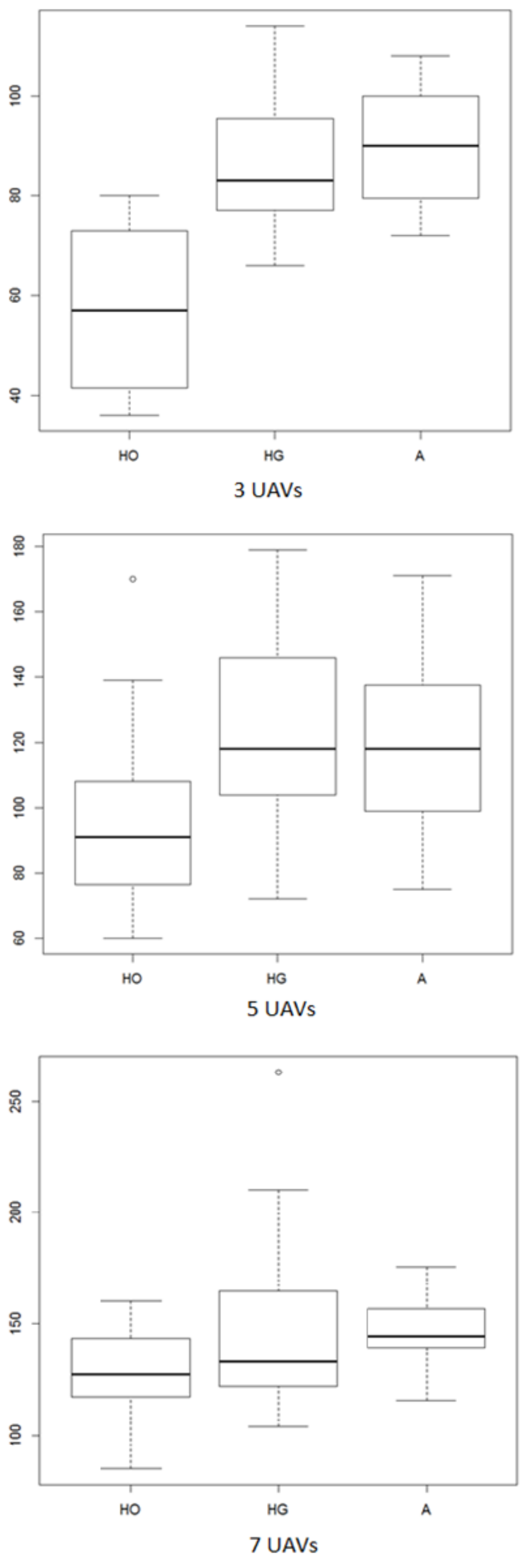


Figure 5.2: Number of Interactions for Different Modes and Numbers of UAVs

The purpose of automation mode was to reduce workload because path generation was done for the operator. The process of manual replanning was made more difficult than necessary in order to encourage the operator to use automation. In human only mode, waypoints were operator-defined, and operators who understood how the interface works only placed waypoints at turning points along their paths. Unlike human only mode, automation mode displayed waypoints at a regular interval along paths, and therefore even straight stretches had multiple waypoints. The reasoning behind placing so many waypoints along the paths was that it would discourage the operators from using automation mode like human only and making significant modifications to the paths. Instead it was expected that participants would use the "Generate Path" button to replan, and make small changes only when necessary, for example in order to avoid obstacles. Instead, participants tried to emulate human only mode and deleted the extra waypoints to get as much control over the paths as possible. By doing this, they did not take advantage of the full benefits of the RRT algorithm in automation mode. This observation led to two questions.

The first question is, how does this behavior change as the number of UAVs increases? Figure 5.3 shows the ratio of the number of waypoint deletions to the number of "Generate Path" button clicks. The lower the ratio, the more the operator is relying on the automation for path planning, and higher ratios mean the operator prefers to manipulate the paths manually.

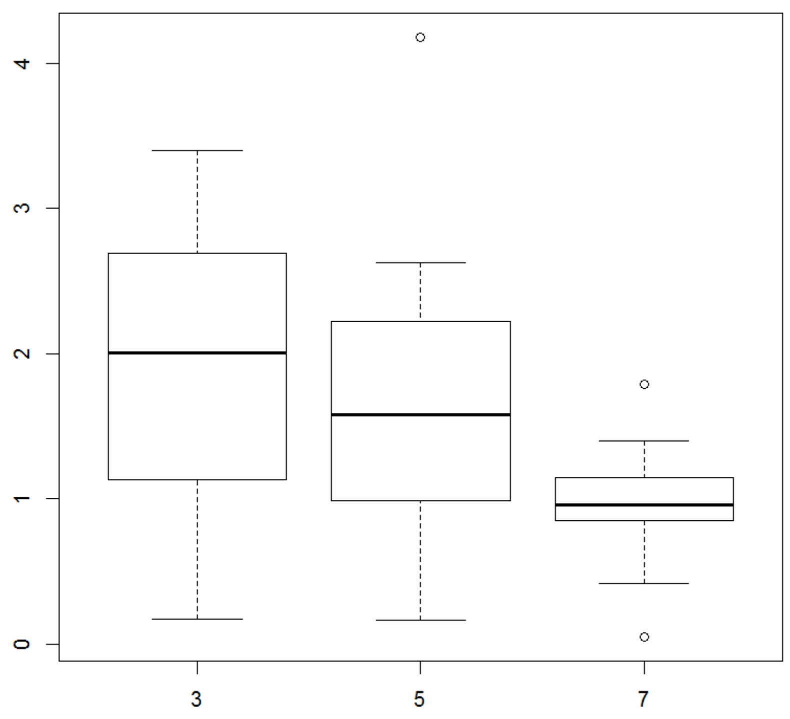


Figure 5.3: Ratio of Number of Waypoint Deletions to Generate Path Clicks in Automation Mode for Different Numbers of UAVs

Figure 5.3 and Table F.26 in Appendix F show that the number of UAVs does have an impact on this ratio (F(2, 33) = 3.66, p = 0.037). There is a statistically significant difference between 3 UAVs and 7 UAVs. It seems that the bigger the number of UAVs, the more people rely on automation to plan paths. This makes sense intuitively: when there are more UAVs the operators are kept busier, so they do not have the time to optimize all the paths by deleting extra waypoints. They spend more time generating paths for UAVs and starting them moving rather than modifying existing collision-free paths.

The second question is, how do the performances of the people who relied more on automation differ from the performances of the people who chose to excessively delete the extra waypoints? Tables 5.1 and 5.2 show least squares linear regression statistics for various performance metrics against the ratio of waypoint deletions to "Generate Path" clicks for 3 UAVs and 7 UAVs. Figures F.1 and F.2 in Appendix F show plots of this data as well as the least squares regression lines.

Table 5.1: Ratio vs. Various Metrics for 3 UAVs

	B	SE(B)	t	R^2	Adj. R^2	F(1, 10)	p
Total Mission Time	10.8	9.94	1.09	0.11	0.016	1.18	0.3
Average Mission Time	12.3	9.04	1.36	0.16	0.072	1.86	0.2
Time in Obstacles per UAV	0.21	0.37	0.56	0.03	-0.067	0.31	0.59
2nd Task Percent Answered	-4.2	4.1	-1.03	0.095	0.0048	1.05	0.33

Table 5.2: Ratio vs. Various Metrics for 7 UAVs

	B	SE(B)	t	R^2	Adj. R^2	F(1, 10)	p
Total Mission Time	33.2	39.7	0.838	0.066	-0.028	0.703	0.42
Average Mission Time	27.8	28.0	0.992	0.09	-0.0014	0.98	0.34
Time in Obstacles per UAV	-2.26	2.75	-0.82	0.063	-0.031	0.672	0.43
2nd Task Percent Answered	-11.1	10.7	-1.04	0.098	0.0075	1.08	0.32

The estimated slopes, listed under B in Tables 5.1 and 5.2, demonstrate better performance in terms of mission completion times, time inside obstacles per UAV, and secondary tasks answered with a smaller ratio of waypoint deletions to "Generate Path" clicks. This means that in general, operators who used RRT more in automation mode performed better and had a lower cognitive load than operators who used it less. This is true for 3 UAVs and 7 UAVs (with the exception of time spent inside obstacles for 7 UAVs), but interestingly missions with 5 UAVs do not follow this pattern. Results for 5 UAVs are shown in Table 5.3. Plots of this data can be found in Figure F.3 in Appendix F.

Table 5.3: Ratio vs. Various Metrics for 5 UAVs

	B	SE(B)	t	R^2	Adj. R^2	F(1, 10)	p
Total Mission Time	-4.0	12.608	-0.32	0.01	-0.089	0.101	0.76
Average Mission Time	-6.6	7.814	-0.85	0.067	-0.026	0.716	0.42
Time in Obstacles per UAV	-0.5	0.6635	-0.76	0.055	-0.04	0.577	0.46
2nd Task Percent Answered	3.3	5.184	0.64	0.039	-0.057	0.403	0.54

With 5 UAVs, the regressions demonstrate that operators had lower mission completion times, lower total time in obstacles, and more secondary task answers with higher ratios of waypoint deletions to "Generate Path" clicks. A possible explanation for this difference is that with 7 UAVs, operators did not have time to make many waypoint deletions; the people who spent a lot of time deleting waypoints neglected UAVs that were waiting for a path or missed obstacle incursions; they also had less time to respond to secondary task queries. Since operators did not have the time to optimize every path, usually it was more advantageous to use the algorithm in order to lower workload. With 5 UAVs, it seems the operators had enough idle time to be able to use that time to optimize most of the paths by deleting extra waypoints, so they chose to do this and this strategy indeed reduced mission completion time, because in general even some of the best paths output by the algorithm could be modified to become even shorter. A possible reason why the pattern holds with 3 UAVs and not with 5 is that with only 3 UAVs, operators had plenty of time to experiment with the algorithm and use it to find a good path, instead of modifying a barely acceptable solution. With 5 UAVs operators had a heavier workload and sometimes they would get nervous about spending too much time on one UAV, so they would decide to modify the current path manually instead of searching for a new path with the algorithm, afraid of the delay that the algorithm would sometimes introduce.

None of the p-values of the linear regressions show a statistically significant effect, so the questions raised by this observation cannot be answered conclusively with the data from this experiment. However, the patterns found with linear regression support the observation and indicate that this area should be explored further. Given that in this experiment, automation mode performed almost as well as human only mode, and given that in automation mode people who relied on RRT more when replanning performed better, if the mode were redesigned with these considerations it could potentially outperform human only in all metrics. Future versions of this interface should contain a mode that like automation combines both RRT and manual modification, and in addition encourages or enforces the use of RRT or somehow limits the use of waypoint manipulation.

Chapter 6: Conclusions and Future Work

This thesis presented the second round in an iterative design project studying human operator interaction with a collaborative Rapidly exploring Random Tree (c-RRT) path planning algorithm. The implementation of the algorithm was kept mostly unchanged from the original project [6], but the interface was redesigned with principles of learnability and usability, and with considerations of the feedback from the previous version. An experiment was run in which users completed three UAV missions in a simulated dynamically changing environment, with 3, 5, or 7 UAVs. Each mission presented a different mode of operation, where each mode allows a different type of interaction with the RRT algorithm. The human only mode does not involve the algorithm; the human guided mode allows operators to influence the algorithm's solution but not to modify it; and the automation mode allows operators to modify the algorithm's solution but not to influence the search. Objective data was measured during the experiment and additional subjective feedback was collected from participants after the experiment. The data was used to analyze the effects of RRT on the cognitive load of the operator, performance in the missions, self-rated operator performance and frustration, and the consequences of increasing the number of UAVs.

The results of the experiment did not conclusively answer the main research questions of the thesis, but they provided many interesting insights that point to future directions for research. Interacting with RRT through the use of wayareas but with no manual manipulation was shown to result in significantly poorer performance and higher frustration. Completely manual manipulation and the use of RRT with the ability to modify paths resulted in comparable performance and subjective feedback. Increasing the number of UAVs produced some interesting patterns. For example, it was found that with increasing numbers of UAVs, operators relied more on automation (as opposed to manual input) when replanning. A surprising result was that significantly fewer participants preferred automation mode with 5 UAVs than with either 3 or 7 UAVs.

5.1 Future Work

This research effort raises additional questions, including:

Would operators perform better if UAVs were numbered depending on their distance from their target or had IDs that were not obviously ordinal, such as shapes? An experiment could be designed to compare performance between groups of participants with different identification schemes, including the default constant numbering scheme, and these suggested IDs.

How does people's video game experience influence their performance and preferences when interacting with RRT? Participants in a new experiment could be divided by video game experience, and results could be compared to the games they play.

Does the performance across modes change as training time or the intensity of training increases? A similar experiment to this one could be repeated but with increasingly thorough training for different groups of participants.

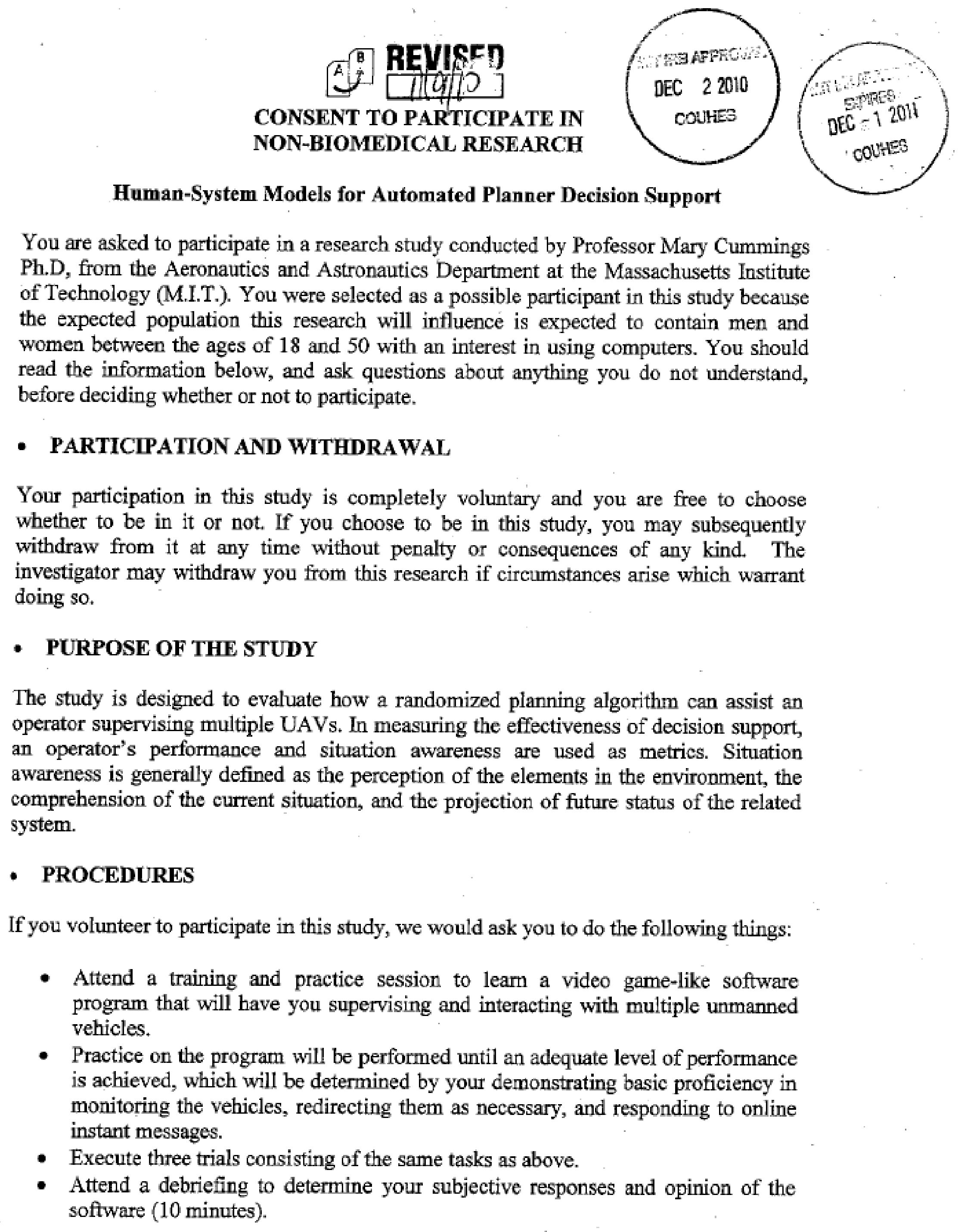
There are many important issues that were discovered in this iteration of the interface, which should be dealt with in future versions. One element that caused frustration is the overly long time limit for the algorithm, so the redesigned interface should significantly decrease this value; during the experiment, the average runtime of the algorithm during a successful search was only 0.4 seconds, while the time limit was 10 seconds, so if the limit is lowered to around 1 second the RRT modes should be much less frustrating. The automation mode in this experiment can be made more successful if the number of unnecessarily displayed waypoints is decreased. In the modified version of automation mode, waypoints should only appear at turning points along the paths, or the number of waypoints displayed should be possible for the operator to control. In addition, the use of RRT can be encouraged or the manipulation of waypoints can be restricted to lead operators to rely more on automation. This study showed that many people prefer to have manual control over paths; waypoint manipulation was significantly preferred to wayareas because of the waypoints' precision.

Some possible additional features to explore are allowing the algorithm to automatically replan when an obstacle collision is imminent, making the algorithm display several paths simultaneously to allow the operator to choose among them, and providing the capability to modify only a section of a path using RRT instead of replanning the entire path. Future studies can look into combining the human guided and automation modes; the main complaint about human guided mode was the lack of control due to not having waypoints. Combining the two types of interaction may result in better performance than either mode separately. An additional mode can be added to the interface, in which the mission is completed entirely or almost entirely by automation. The performance of modes with a significant amount of human input can be compared to the performance of a mode where the majority of actions and decisions are carried out by automation to gain insight into the value of having a human operator.

There are many interesting questions that can be explored by looking at the data collected from this experiment that were not answered for this thesis. For example, the data provides information on the paths that were accepted by human operators and the paths that were rejected. A machine learning model could be constructed to predict whether the operator will approve of a path, and this model could help improve the automation.

Lastly, in order to study RRT in more realistic situations, a third dimension can be added to the space so that path planning will become a 3D problem, as it usually is in real applications.

Appendix A: Consent Form



REVISED 11/9/10

APPROVED DEC 2 2010 COUHES

EXPIRES DEC 1 2011 COUHES

CONSENT TO PARTICIPATE IN NON-BIOMEDICAL RESEARCH

Human-System Models for Automated Planner Decision Support

You are asked to participate in a research study conducted by Professor Mary 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 expected population this research will influence is expected to contain men and women between the ages of 18 and 50 with an interest in using computers. 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 study is designed to evaluate how a randomized planning algorithm can assist an operator supervising multiple UAVs. In measuring the effectiveness of decision support, an operator's performance and situation awareness are used as metrics. Situation awareness is generally defined as the perception of the elements in the environment, the comprehension of the current situation, and the projection of future status of the related system.

- **PROCEDURES**

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

- Attend a training and practice session to learn a video game-like software program that will have you supervising and interacting with multiple unmanned vehicles.
- Practice on the program will be performed until an adequate level of performance is achieved, which will be determined by your demonstrating basic proficiency in monitoring the vehicles, redirecting them as necessary, and responding to online instant messages.
- Execute three trials consisting of the same tasks as above.
- Attend a debriefing to determine your subjective responses and opinion of the software (10 minutes).

- All testing will take place in MIT building 37, room 301, or in the mobile experimental testbed vehicle (MACCS), a computer-equipped cargo van to be parked and stationary while experiments take place.
- Total time: 1.5 hours, depending on skill level.

- **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 applicability of randomized planning algorithms in UAV applications.

- **PAYMENT FOR PARTICIPATION**

You will receive $10 per hour for your efforts. The participant who reaches the best overall performance upon completion of the full experiment, will receive a $100 gift card. If several participants reach the same best performance, they will be entered in a drawing, and a winner will be chosen at random to receive the $100 gift card.

- **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 33-305, Cambridge, MA 02139. The graduate student investigator is Alina Griner phone (617) 258-5046, email, a_griner@mit.edu and the Postdoctoral Associate investigator is Luca F. Bertuccelli phone (617) 258-5046, email lucab@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

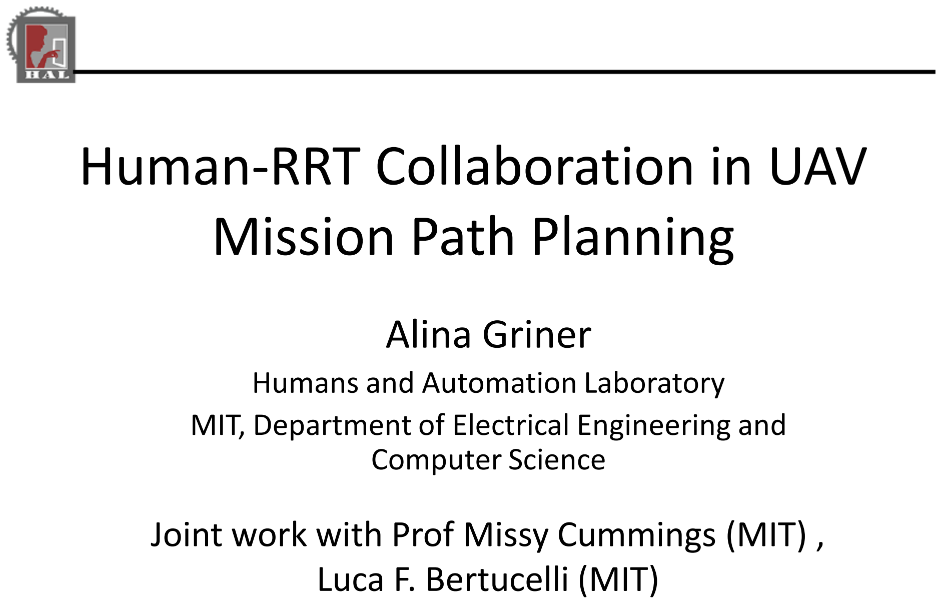
________________________________ ____________
Signature of Subject Date

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: Training Slides



Human-RRT Collaboration in UAV Mission Path Planning

Alina Griner

Humans and Automation Laboratory

MIT, Department of Electrical Engineering and Computer Science

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

Context

- UAVs (Unmanned Aerial Vehicles) are unmanned aircraft that are used for a variety of different purposes, such as surveillance, combat, and search and rescue
- Path planning is an important factor in all types of UAV missions
- This is a path planning experiment where your goal is to find collision-free paths for each UAV to its target

Context

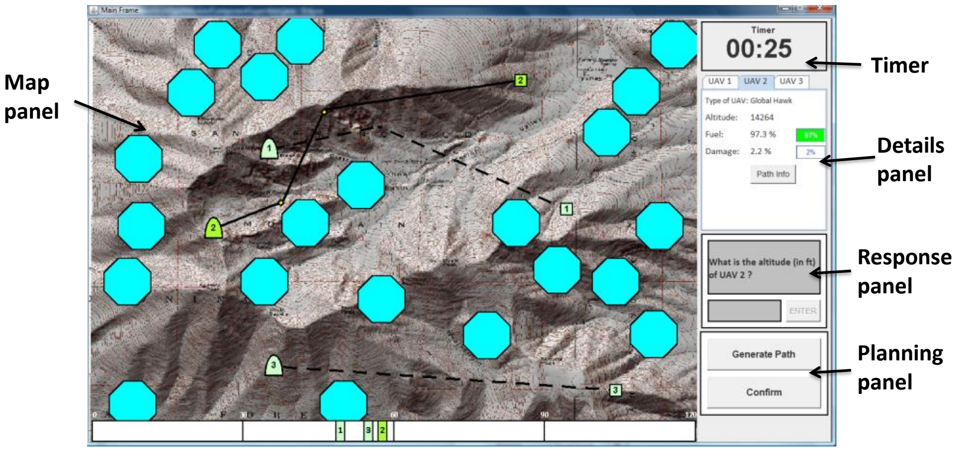
- You will be supervising UAVs in simulated missions
- Objectives:
  - Guide UAVs to a sequence of targets
  - Maneuver the UAVs to their targets as quickly as possible
  - Avoid obstacles whenever possible
- The mission is completed when each UAV has reached its last target

Experiment Outline

- Training (10 min)
- Mission (5-10 min) + post-mission feedback survey
- Mission (5-10 min) + post-mission feedback survey
- Mission (5-10 min) + post-mission feedback survey
- Additional Feedback

The Environment

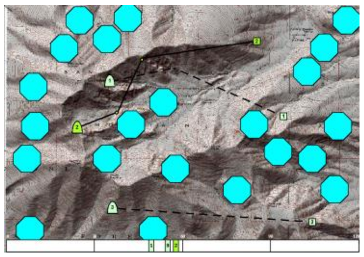
- 🟩 – obstacle, 1 – UAV, 1 – target (number inside is ID)
- When UAV is selected, the vehicle and its target change to **green**
- Path shown in black, yellow points on path are movable waypoints



The Environment

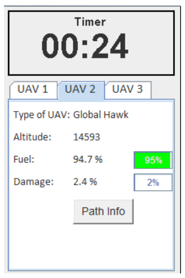
- 🟩 – obstacle, 1 – UAV, 1 – target (number inside is ID)
- When UAV is selected, the vehicle and its target change to **green**
- Path shown in black, yellow points on path are movable waypoints

Map Panel



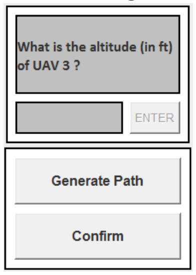
- Shows UAV paths and obstacles
- Can manipulate paths by interacting with the map panel

Timer and Details Panel



- Timer shows time that has elapsed in current mission
- Details panel shows UAV status

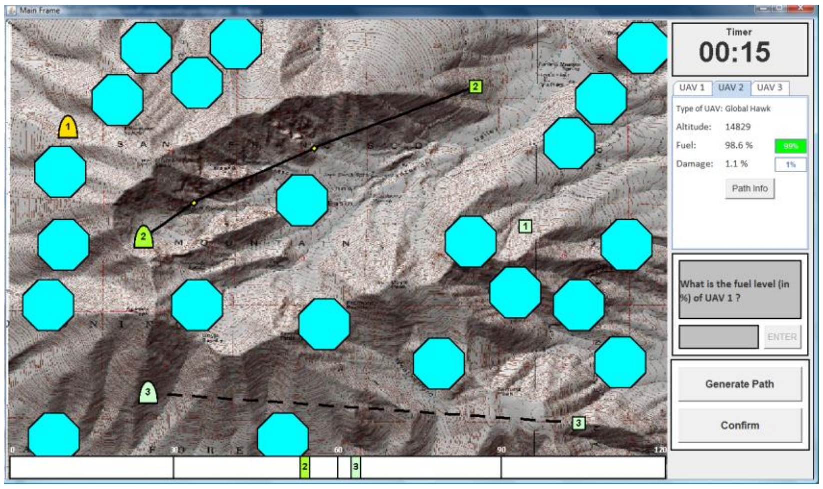
Response Panel, Path Planning Panel



- Response panel poses questions about UAV status
- Generate paths using path planning panel

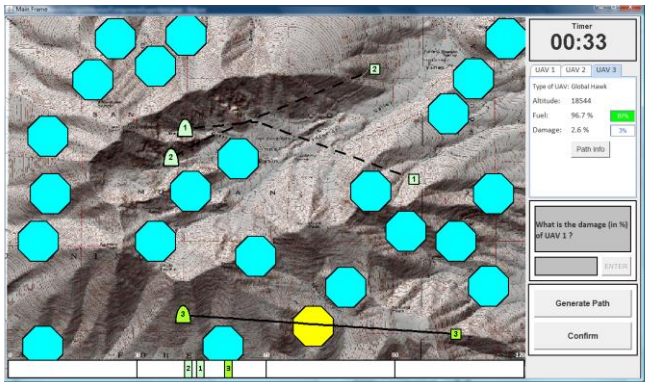
The Environment

- UAVs that have a target but are not moving flash **orange**
- The selected path is **solid**, unselected paths are **dashed**



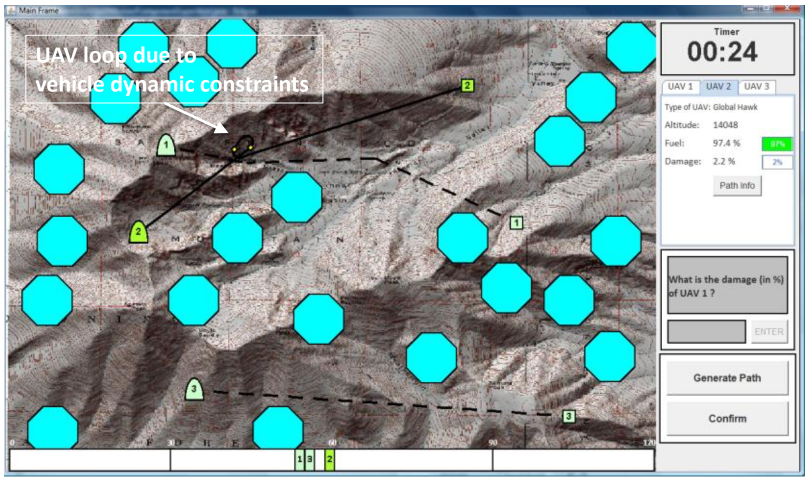
The Environment

- Obstacles are no-fly zones – the UAVs can fly through them, but you should avoid them as there is a large penalty for doing so.
- Obstacles appear and disappear – avoid them as best you can
- If a UAV's path goes through an obstacle, that obstacle will turn yellow in warning, as shown below
- If an obstacle appears and a UAV is inside it, it does not count as a collision



The Environment

- Adding a waypoint too close to the UAV or to another waypoint will result in a loop in the path (see UAV 2 path below)
- Timeline at bottom shows arrival time of each UAV, in seconds

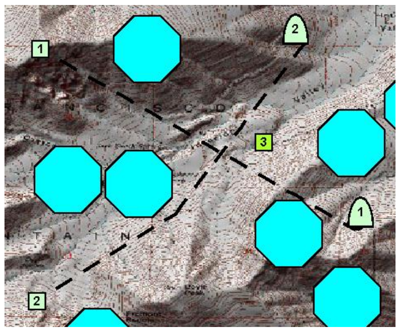


Type of UAV: Global Hawk
Altitude: 14048
Fuel: 97.4 % 97%
Damage: 2.2 % 2%

What is the damage (in %) of UAV 1 ?

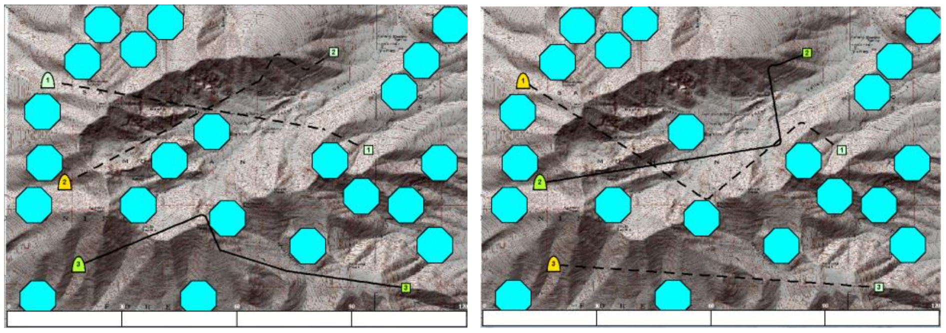
The Targets

- In all missions, the UAVs have four targets each
- One target is visible at a time for each UAV; as soon as a UAV reaches a target, its next target appears
- When a UAV arrives at a target it keeps its heading, so it may need to turn around to go to the next target, as both UAV 1 and UAV 2 do below



RRT – The Planning Algorithm

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



Examples of random solutions generated by the RRT algorithm

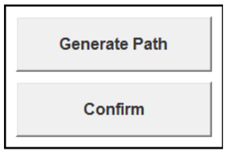
RRT – The Planning Algorithm

- Sometimes the algorithm does not find a solution within its time limit. In this case, if the UAV does not currently have a path, it will output a straight line from the UAV to its goal; if the UAV does have a path, it will keep the current path. If there are obstacle incursions, you should try to replan.
- While the algorithm is searching for a solution for one UAV, you can plan paths for other UAVs.

Human Only Mode

Human Only Mode

- Manual path planning: RRT algorithm not involved
- Select UAV by clicking on it
- To generate a path:
  - click on map to make waypoints
  - (do **not** need to click goal)
  - click “Generate Path”
  - (can generate a path without making waypoints: will result in straight line from UAV to target)
  - modify path if not satisfied with it (explained in next slide)
  - click “Confirm”, and the UAV will begin to move



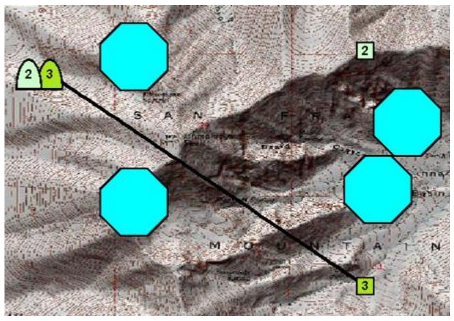
Human Only Mode

- Modifying paths: can add, delete or move waypoints for selected UAV
  - Add a waypoint by left clicking – will connect the two closest waypoints to it on the path with straight lines
  - Delete a waypoint by right clicking the waypoint – will connect the preceding and following waypoints with a straight line
  - Move waypoint by clicking and dragging the waypoint
- Waypoints are hidden for unselected UAVs

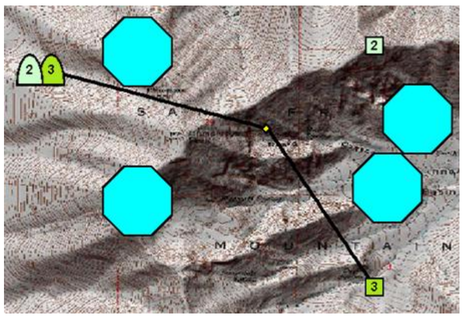
Human Only Mode

Adding a waypoint

Path before adding waypoint



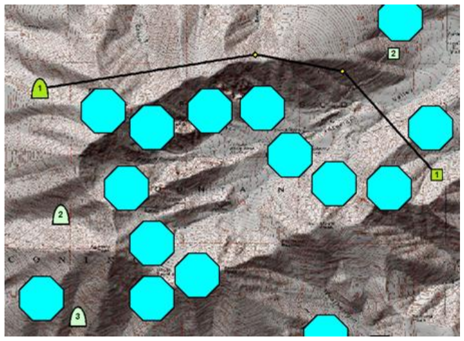
Path after adding waypoint



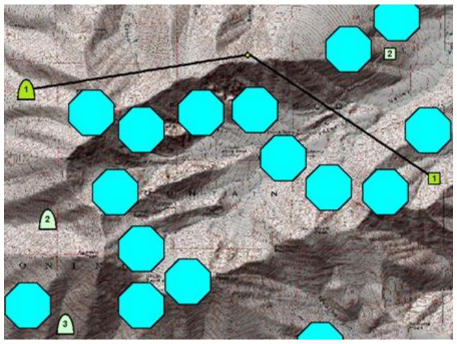
Human Only Mode

Deleting a waypoint

Path before deleting waypoint



Path after deleting second waypoint



Human Guided Mode

Human Guided Mode

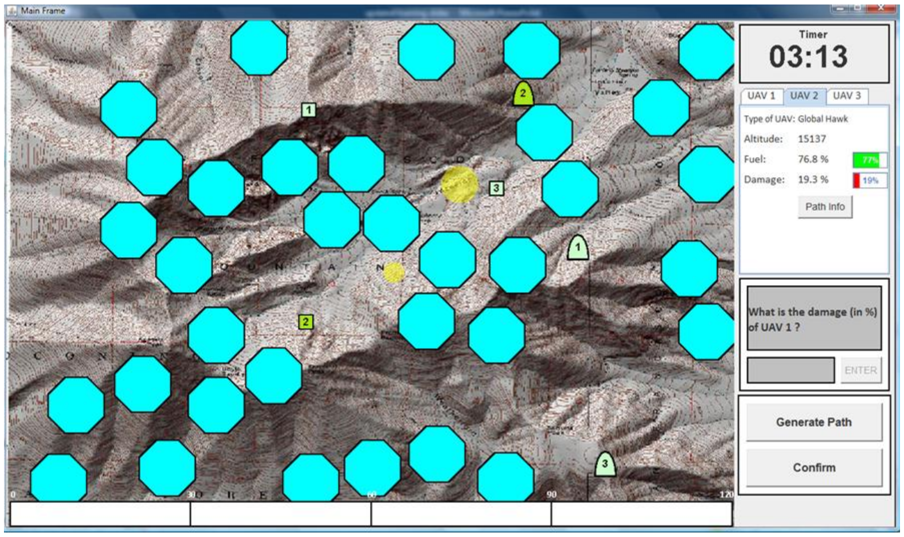
- You can **influence** the solution found by the RRT algorithm
- To generate a path:
  - click and hold mouse on map for center of wayarea, drag mouse for size
  - (wayareas are shown as semi-transparent yellow circles)
  - can delete wayarea by right clicking it
  - click “Generate Path”
  - (can generate a path without making wayareas)
  - Click “Confirm”, and the UAV will begin to move

Human Guided Mode

- The path found by the RRT will go through each wayarea, unless a large part of a wayarea is contained inside an obstacle, in which case that wayarea may be ignored
- You have no control over which part of a wayarea the path will contain, so if you want the UAV to go through a specific point, draw a small wayarea there, whereas if there is a large space you want the UAV to go, draw a large wayarea there.
- Use wayareas to help the algorithm find paths through narrow passages
- No waypoints will be shown on paths; you cannot modify paths after they are generated
- If you need to replan, generate another path as above
- Sometimes you can make small wayareas without realizing; if this happens, you can start the UAV moving and then replan

Human Guided Mode

- Wayareas are input for UAV 2

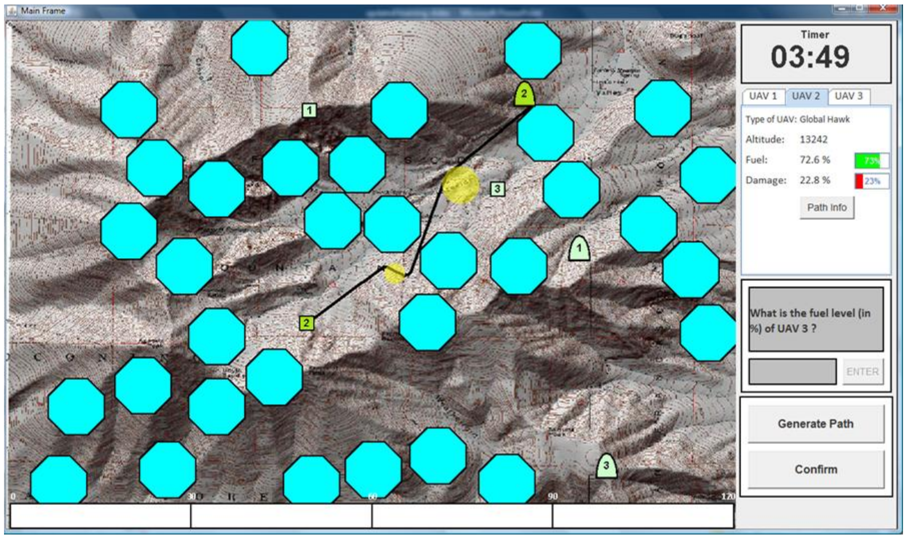


Type of UAV: Global Hawk
Altitude: 15137
Fuel: 76.8 % 77%
Damage: 19.3 % 19%

What is the damage (in %) of UAV 1 ?

Human Guided Mode

- The path generated goes through the wayareas



Type of UAV: Global Hawk
Altitude: 13242
Fuel: 72.6 % 73%
Damage: 22.8 % 23%

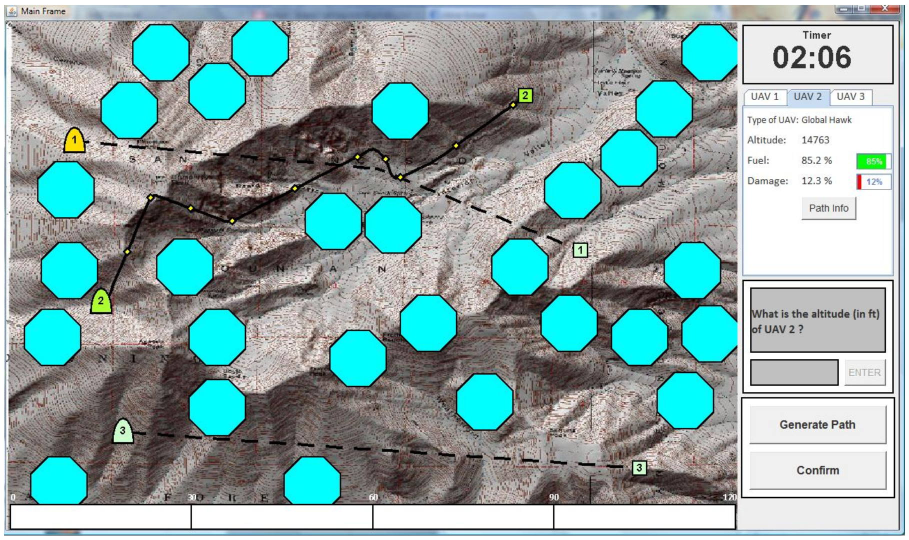
What is the fuel level (in %) of UAV 3 ?

Automation Mode

Automation Mode

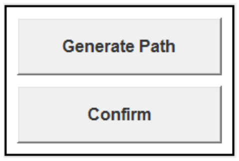
- You can **modify** the solution found by the RRT algorithm
- To generate a path:
  - click “Generate Path”
  - can modify it as in human only mode
  - when satisfied with the path, click “Confirm”
- Waypoints will be shown on the selected UAV’s path
- You can modify paths after they are generated by adding waypoints, deleting waypoints, and moving waypoints as in human only mode
- If you need to replan, you can either modify the current path by manipulating waypoints or you can generate a new path by clicking “Generate Path”

Automation Mode



Path Planning in All Modes

- While a UAV is stopped, you can continue to generate paths for it until you find one you like. When you are satisfied with a path, you must click “Confirm”, and then the UAV will begin to move.
- If you replan while a UAV is moving, you will not have to click “Confirm”. The UAV will automatically switch to the new path.
- When the algorithm is in the process of finding a path for the selected UAV, the Generate Path button is disabled.

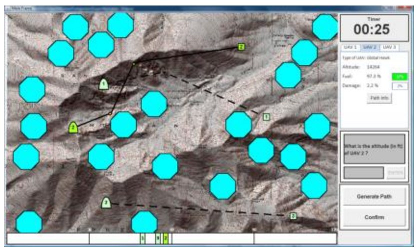


Strategy Hints

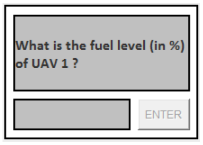
- If there is a straight line path available from the UAV to its target, there is a high probability that the RRT algorithm will find it. If it does not find it, it may be easier to click “Generate Path” several more times until it outputs it rather than modify the path.
- The timeline is a good tool for finding which UAV will be the next to arrive at its goal.
- In the automation and human only modes, if a generated path is fairly good but needs some modification, it may save you time to first confirm it so that the UAV starts moving and then modify it, rather than the other way around.

Secondary Task

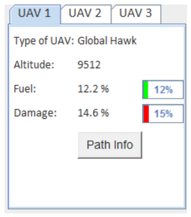
- On the right of the screen, there is a response panel that will occasionally display questions for you to answer (it flashes when a new question appears)
- You can find the answers to these questions in the details panel above the response panel (you can change tabs by clicking)
- You should try to answer the questions, but the **UAVs are your first priority**: if you have a choice between answering a question and helping a UAV avoid collision with an obstacle, help the UAV avoid collision



Response Panel


What is the fuel level (in %) of UAV 1 ?

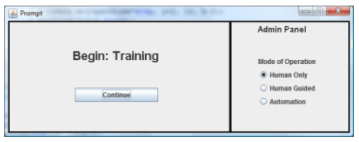
Details Panel



Type of UAV: Global Hawk
Altitude: 9512
Fuel: 12.2 % 12%
Damage: 14.6 % 15%

Training

- Allows practice in all modes with 3 UAVs
  - Select mode in Admin Panel
  - Click “Continue”
  - Can restart by clicking “Restart Training” in upper left corner (can try another mode by repeating process)
- When done or 10 minutes are up, click “End Training”

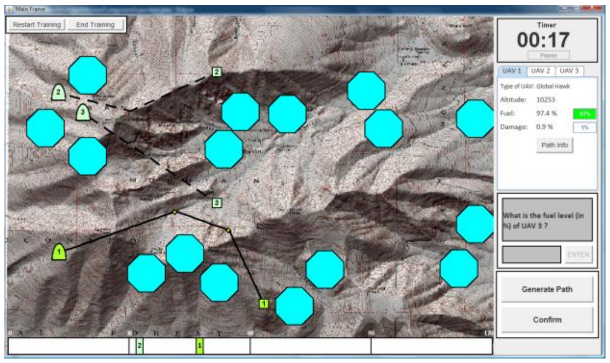

Prompt

Begin: Training

Continue

Admin Panel

Mode of Operation
[✓ Human Only]
Human Guided
Automation



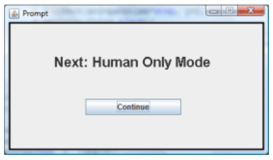
Timer
00:17

Type of UAV: Global Hawk
Altitude: 10253
Fuel: 97.4 % 97%
Damage: 0.9 % 1%

What is the fuel level (in %) of UAV 3 ?

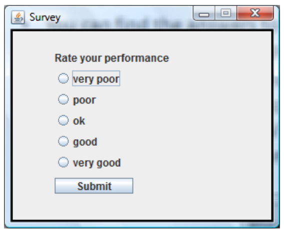
Transitions

- Before every mission, you will get a screen like the one shown below to the left indicating the mode of the next mission
- After every mission (not including training) you will get two survey windows (below to the right). Select response, then click “Submit”.


Prompt

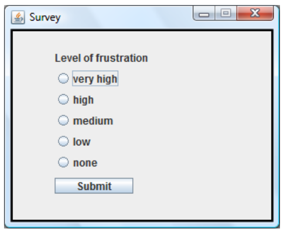
Next: Human Only Mode

Continue


Survey

Rate your performance
- very poor
- poor
- ok
- good
- very good

Submit


Survey

Level of frustration
- very high
- high
- medium
- low
- none

Submit

Summary

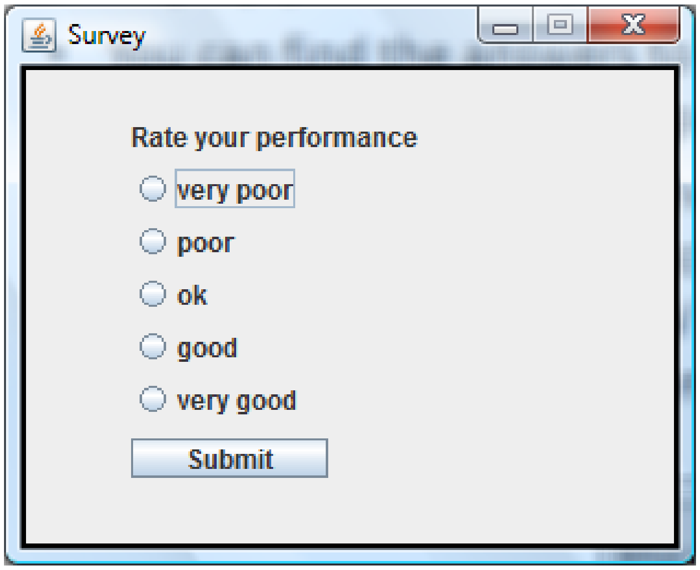
- Get the UAVs to their targets as quickly as possible.
- Avoid obstacles. The time a UAV spends inside an obstacle matters, so try to get UAVs out of obstacles as soon as possible.
- Answer the secondary task questions when you can, but the UAVs are your first priority.

The participant who achieves the highest score will win a $100 gift card!

Questions?

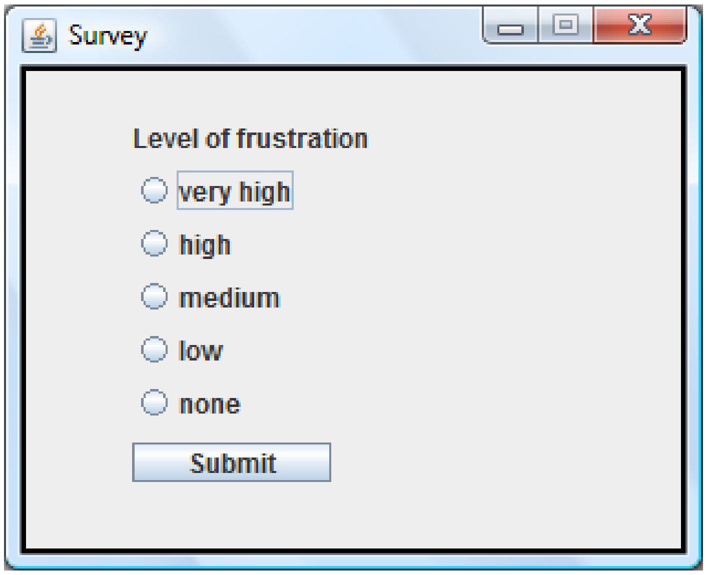
If not, then proceed to training!

Appendix C: Post-Mission Surveys



Rate your performance

- very poor
- poor
- ok
- good
- very good



Level of frustration

- very high
- high
- medium
- low
- none

Appendix D: Data Logged During the Experiment

Event Log File:

```
% GoalReached,timestamp, UAV ID, target location, list of points on
path taken to target

% WPAdd,timestamp,UAV ID, UAV location, way point location,list of
angles of curves, path length, straight line distance to next target

% WPDelete,timestamp,UAV ID, UAV location, way point location,list of
angles of curves, path length, straight line distance to next target

% WPMove,timestamp,UAV id, UAV location, way point start location,way
point end location, move duration, list of angles of curves, path
length, straingh line distance to next target

% WPsRemoved,timestamp, UAV id, UAV location, list of angles of
curves, path length,straight line distance to next target

% SubgoalAdd,timestamp,UAV ID, UAV location, subgoal location

% SubgoalDelete,timestamp,UAV ID, UAV location, subgoal location

% SubgoalMove,timestamp,UAV id, UAV location, way point start
location, way point end location, move duration

% SubgoalsRemoved,timestamp, UAV id, UAV location

% UAVOutOfObstacle,timestamp, UAV ID, UAV location, time in obstacle,
location of obstacle

% RRTPathGenerated,timestamp, UAV ID, UAV location, list of goals,
list of angles of curves, path length, straight line distance to next
target, runtime (ns), path points

% NoRRTSolFound,timestamp, UAV ID, UAV location,runtime (ns)

% ClickedInputButton,timestamp, UAV ID, UAV location

% ClickedGeneratePath,timestamp, UAV ID, UAV location, list of
wayarea/waypoint center (and radius), number of wayareas/waypoints

% ClickedTooSoon,timestamp, UAV ID, UAV location, difference between
click times

% HOPathGenerated,timestamp, UAV ID, UAV location, list of path
points, path length, straight line distance
```

```
% AddWA,timestamp, UAV ID, UAV location, wayarea center, wayarea
radius

% DeleteWA,timestamp, UAV ID, UAV location, wayarea center, wayarea
radius

% ClickedPathInfo,timestamp, UAV ID, UAV location

% ObsTurnsYellow,timestamp, obstacle location, list of UAV info for
UAVs in danger and whether UAV was inside when appeared

% SecondaryTaskAppears,timestamp, UAV ID, category of question,
correct answer at time appears

% AnswersSecondaryTask,timestamp, user's answer, correct answer

% ConfirmPath,timestamp, UAV ID, UAV location
```

Snap Log File:

```
% ObstacleProximity,timestamp, pairs of UAV ids and distance to
closest obstacle

% ObstacleLocation,timestamp, list of obstacle locations

% Path,timestamp, UAV ID, UAV locationpoints on remainining path to
next target
```

Appendix E: Post-Experiment Feedback Survey

1. What was your favorite mode? Why?

2. What did you think of the other modes?

3. How would you order the modes in order of preference?

4. What would you change in the interface?

5. Did you have enough training before the experiment?

6. Do you have any video game experience? If yes, what kind of games?

7. What major are you, or what field are you in?

Appendix F: Supportive Statistics

Table F.1: ANOVA for Percentage of Secondary Tasks Answered

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	37289	18644	75.63	< 2.2e-16
Mode of Operation	2	3082	1541	6.25	0.0028
Interaction	4	175	44	0.18	0.95
Residuals	99	24407	247		

Table F.2: Tukey HSD for Percentage of Secondary Tasks Answered

Number of UAVs:

	diff	lwr	upr	p adj
5-3	-23.75	-32.55	-14.94	0e+00
7-3	-45.5	-54.31	-36.69	0e+00
7-5	-21.75	-30.56	-12.95	2e-07

Mode of Operation:

	diff	lwr	upr	p adj
HG-A	-3.1	-11.91	5.7	0.68
HO-A	9.46	0.65	18.26	0.032
HO-HG	12.56	3.75	21.37	0.0028

Table F.3: ANOVA for Total Mission Completion Time

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	196073	98037	29.48	9.03e-11
Mode of Operation	2	138420	69210	20.81	2.85e-08
Interaction	4	19642	4911	1.48	0.22
Residuals	99	329217	3325		

Table F.4: Tukey HSD for Total Mission Completion Time

Number of UAVs:

	diff	lwr	upr	p adj
5-3	74.18	41.84	106.53	1.1e-06
7-3	100.67	68.33	133.01	0.0
7-5	26.49	-5.86	58.83	0.13

Mode of Operation:

	diff	lwr	upr	p adj
HG-A	52.81	20.47	85.15	0.00054
HO-A	-34.23	-66.57	-1.89	0.035
HO-HG	-87.03	-119.38	-54.69	0.0

Table F.5: ANOVA for Average Mission Completion Time

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	52607	26303	13.57	6.2e-06
Mode of Operation	2	63843	31921	16.47	6.7e-07
Interaction	4	3945	986	0.51	0.73
Residuals	99	191913	1939		

Table F.6: Tukey HSD for Average Mission Completion Time

Number of UAVs:

	diff	lwr	upr	p adj
5-3	31.83	7.14	56.53	0.0078
7-3	53.76	29.06	78.45	3.5e-06
7-5	21.92	-2.77	46.62	0.092

Mode of Operation:

	diff	lwr	upr	p adj
HG-A	34.25	9.56	58.94	0.0038
HO-A	-25.07	-49.76	-0.38	0.046
HO-HG	-59.32	-84.01	-34.63	3.0e-07

Table F.7: ANOVA for Total Number of Incursions per UAV

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	6.3	3.15	11.2	4.14e-05
Mode of Operation	2	2.95	1.47	5.23	0.0069
Interaction	4	0.83	0.21	0.74	0.57
Residuals	99	27.87	0.28		

Table F.8: Tukey HSD for Total Number of Incursions Per UAV

Number of UAVs:

	diff	lwr	upr	p adj
5-3	0.38	0.08	0.68	0.0089
7-3	0.58	0.29	0.88	0.000029
7-5	0.21	-0.092	0.5	0.23

Mode of Operation:

	diff	lwr	upr	p adj
HG-A	0.26	-0.033	0.56	0.092
HO-A	-0.13	-0.43	0.16	0.54
HO-HG	-0.4	-0.69	-0.1	0.0056

Table F.9: ANOVA for Total Time in Obstacles Per UAV

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	117.5	58.74	10.63	6.56e-05
Mode of Operation	2	72.6	36.32	6.57	0.0021
Interaction	4	22.6	5.64	1.021	0.4
Residuals	99	547.0	5.52		

Table F.10: Tukey HSD for Total Time in Obstacles per UAV

Number of UAVs:

	diff	lwr	upr	p adj
5-3	1.17	-0.14	2.49	0.091
7-3	2.55	1.23	3.87	0.000036
7-5	1.38	0.06	2.7	0.038

Mode of Operation:

	diff	lwr	upr	p adj
HG-A	1.28	-0.04	2.6	0.059
HO-A	-0.7	-2.02	0.61	0.42
HO-HG	-1.98	-3.3	-0.66	0.0016

Table F.11: ANOVA for Number of RRT failures per UAV

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	5.82	2.91	8.93	0.00037
Mode of Operation	1	3.77	3.77	11.56	0.0012
Interaction	2	0.93	0.46	1.42	0.25
Residuals	66	21.52	0.33		

Table F.12: Tukey HSD for Number of RRT failures per UAV

Number of UAVs:

	diff	lwr	upr	p adj
5-3	0.53	0.14	0.93	0.0053
7-3	0.65	0.26	1.05	0.00052
7-5	0.12	-0.27	0.52	0.74

Mode of Operation:

	diff	lwr	upr	p adj
HG-A	0.46	0.19	0.73	0.0012

Table F.13: Non-parametric Test Results for Subjective Performance and Frustration

Subjective Performance:

Wilcoxon Signed Rank Test

	V	p-value
HO vs. HG	229	0.00057
HO vs. A	102.5	0.43
HG vs. A	20	0.00019

Wilcoxon Rank Sum Test

	W	p-value
3 vs. 5	862	0.0095
3 vs. 7	834	0.026
5 vs. 7	637	0.9

Subjective Frustration:

Wilcoxon Signed Rank Test

	V	p-value
HO vs. HG	378	2.23e-06
HO vs. A	94	0.043
HG vs. A	10.5	1.93e-05

Wilcoxon Rank Sum Test

	W	p-value
3 vs. 5	1078	2.72e-07
3 vs. 7	1028	7.41e-06
5 vs. 7	660	0.89

HO vs. A:

Wilcoxon Signed Rank Test

	V	p-value
3 UAVs	1.5	1
5 UAVs	21	0.2402
7 UAVs	18.5	0.1058

Table F.14: ANOVA for Total Mission Completion Time in Third Mission Only

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	56818	28409	10.85	0.00035
Mode of Operation	2	54789	27395	10.46	0.00043
Interaction	4	23610	5903	2.25	0.09
Residuals	27	70711	2619		

Table F.15: Tukey HSD for Total Mission Completion Time in Third Mission Only

Number of UAVs:

	diff	lwr	upr	p adj
5-3	93.27	41.48	145.07	0.00037
7-3	70.66	18.87	122.46	0.0061
7-5	-22.61	-74.41	29.19	0.53

Mode of Operation:

	diff	lwr	upr	p adj
HG-A	36.46	-15.34	88.26	0.21
HO-A	-58.26	-110.06	-6.46	0.025
HO-HG	-94.73	-146.53	-42.93	0.0003

Table F.16: ANOVA for Average Mission Completion Time in Third Mission Only

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	8280	4140	2.7	0.085
Mode of Operation	2	16400	8200	5.36	0.011
Interaction	4	1906	477	0.31	0.87
Residuals	27	41339	1531		

Table F.17: Tukey HSD for Average Mission Completion Time in Third Mission Only

Number of UAVs:

	diff	lwr	upr	p adj
5-3	32.05	-7.56	71.66	0.13
7-3	32.29	-7.32	71.9	0.13
7-5	0.24	-39.37	39.85	1.0

Mode of Operation:

	diff	lwr	upr	p adj
HG-A	3.79	-35.81	43.4	0.97
HO-A	-43.26	-82.87	-3.65	0.03
HO-HG	-47.05	-86.66	-7.45	0.017

Table F.18: ANOVA for Number of Incursions per UAV in Third Mission Only

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	0.97	0.49	1.22	0.31
Mode of Operation	2	0.62	0.31	0.77	0.48
Interaction	4	1.94	0.48	1.2	0.34
Residuals	27	10.95	0.41		

Table F.19 ANOVA for Time in Obstacles per UAV in Third Mission Only

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	74.67	37.34	3.84	0.034
Mode of Operation	2	32.08	16.04	1.65	0.21
Interaction	4	53.98	13.50	1.39	0.26
Residuals	27	262.53	9.72		

Table F.20: Tukey HSD for Time in Obstacles per UAV in Third Mission Only

Number of UAVs:

	diff	lwr	upr	p adj
5-3	0.69	-2.46	3.85	0.85
7-3	3.34	0.19	6.5	0.036
7-5	2.65	-0.51	5.81	0.11

Table F.21: ANOVA for Average RRT Runtimes in Third Mission Only

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	5.3e+16	2.65e+16	0.73	0.5
Mode of Operation	1	4.21e+15	4.21e+15	0.12	0.74
Interaction	2	2.93e+16	1.47e+16	0.4	0.68
Residuals	18	6.55e+17	3.64e+16		

Table F.22: ANOVA for Number of Failures per UAV in Third Mission Only

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	6.17	3.08	12.19	0.00045
Mode of Operation	1	3.58	3.58	14.14	0.0014
Interaction	2	2.36	1.18	4.66	0.023
Residuals	18	4.56	0.25		

Table F.23: Tukey HSD for Number of Failures per UAV in Third Mission Only

Number of UAVs:

	diff	lwr	upr	p adj
5-3	1.24	0.6	1.88	0.0003
7-3	0.61	-0.029	1.26	0.063
7-5	-0.63	-1.27	0.013	0.056

Mode of Operation:

	diff	lwr	upr	p adj
HG-A	0.77	0.34	1.2	0.0014

Table F.24: Two-Way ANOVA for Ratio of Waypoint Deletions to Generate Path Clicks in Automation Mode

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	5.74	2.87	3.48	0.045
Mission Number	2	1.37	0.68	0.83	0.45
Interaction	4	2.27	0.57	0.69	0.61
Residuals	27	22.29	0.825		

Table F.25: Tukey HSD for Ratio of Waypoint Deletions to Generate Path Clicks in Automation Mode

Number of UAVs:

	diff	lwr	upr	p adj
5-3	-0.18	-1.1	0.74	0.88
7-3	-0.92	-1.84	-0.0016	0.05
7-5	-0.75	-1.67	0.17	0.13

Mission Number:

	diff	lwr	upr	p adj
Mission 2-Mission 1	0.36	-0.56	1.28	0.6
Mission 3-Mission 1	0.45	-0.47	1.37	0.46
Mission 3-Mission 2	0.086	-0.83	1.005	0.97

Table F.26: One-Way ANOVA for Ratio of Waypoint Deletions to Generate Path Clicks in Automation Mode

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Number of UAVs	2	5.74	2.87	3.66	0.037
Residuals	33	25.92	0.79		

Table F.27: Tukey HSD for Ratio of Waypoint Deletions to Generate Path Clicks in Automation Mode

Number of UAVs:

	diff	lwr	upr	p adj
5-3	-0.18	-1.064	0.71	0.88
7-3	-0.92	-1.81	-0.033	0.041
7-5	-0.75	-1.63	0.14	0.11

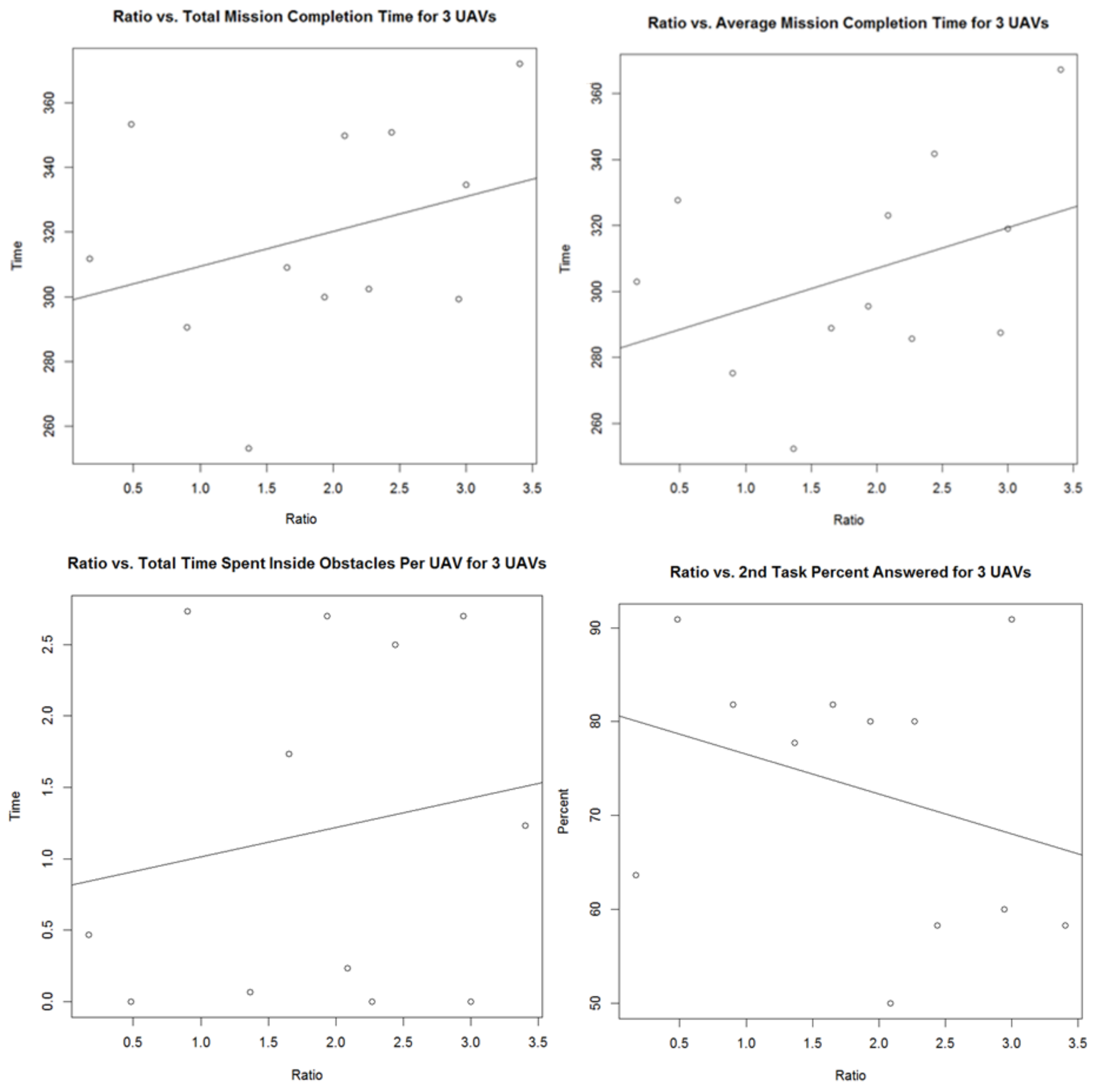


Figure F.1: Ratio of Waypoint Deletions to "Generate Path" Clicks vs. Various Metrics for 3 UAVs

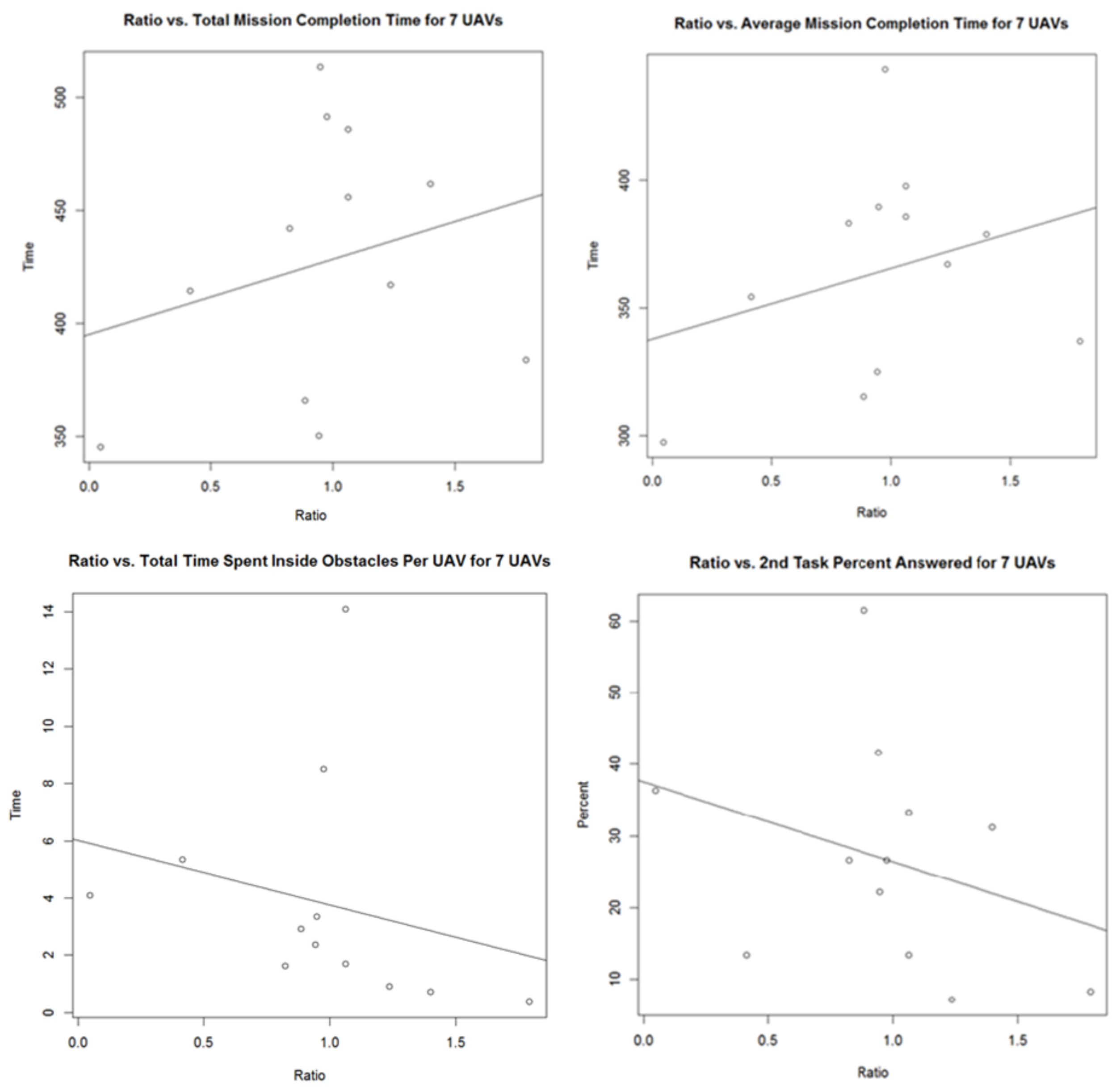


Figure F.2: Ratio of Waypoint Deletions to "Generate Path" Clicks vs. Various Metrics for 7 UAVs

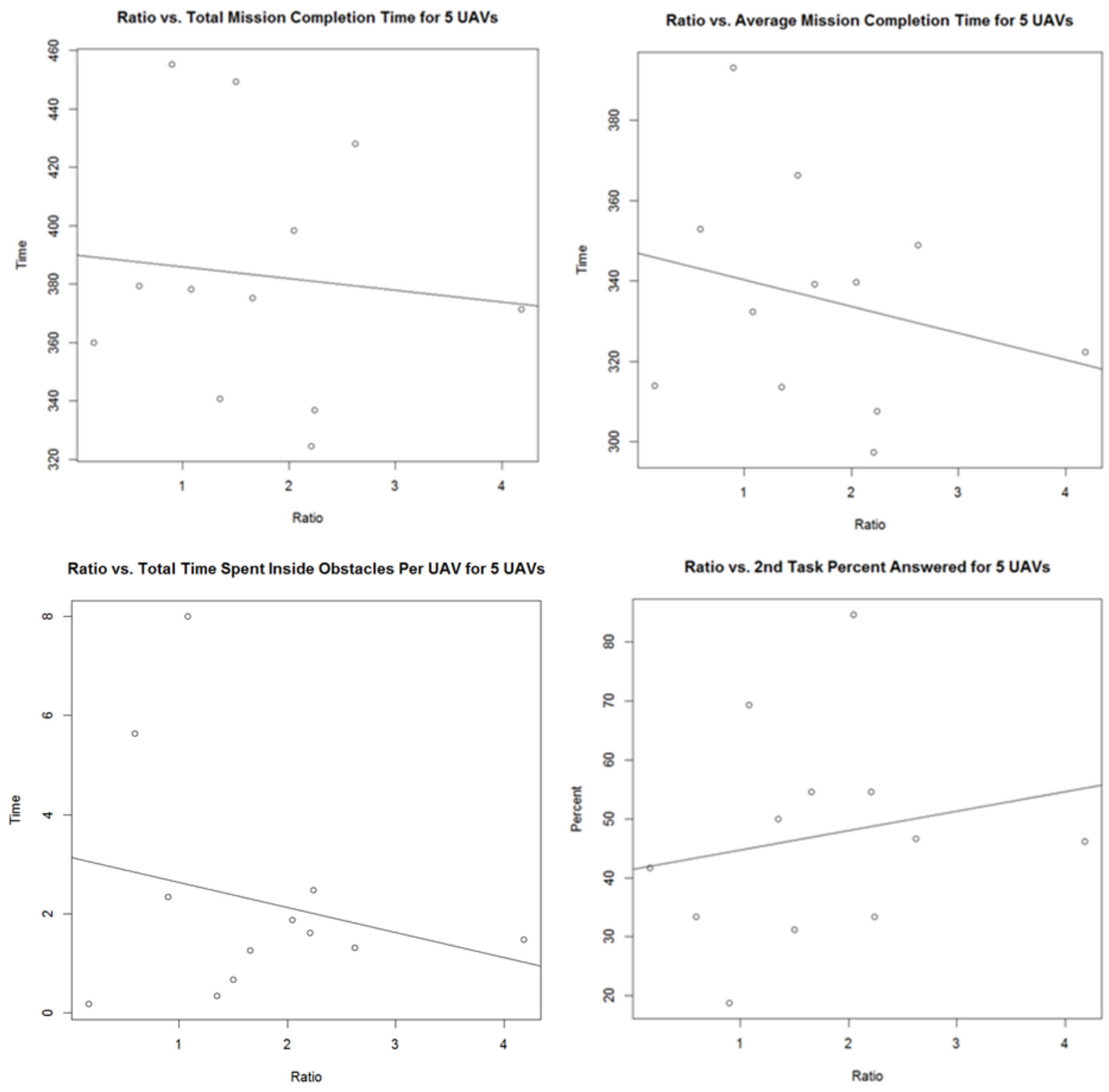


Figure F.3: Ratio of Waypoint Deletions to "Generate Path" Clicks vs. Various Metrics for 5 UAVs

Appendix G: Post-Experiment Feedback Survey Responses

Favorite mode:

3 UAVs:	HO: 6,	A: 5,	HG: 0,	one tie HO-A
5 UAVs:	HO: 9,	A: 2 ,	HG: 1	
7 UAVs:	HO: 6,	A: 4,	HG: 0	two ties HO-A

Order of preference:
Parentheses around two modes indicate they are tied for that place. Next to each order of preference is the number of UAVs that participant had in the missions.

A HO HG	3
HO (A HG)	3
A HO HG	5
HG (A HO)	5
(HO A) HG	3
HO A HG	5
A HO HG	3
HO A HG	5
A HO HG	7
HO A HG	3
HO A HG	5
HO A HG	5
A HO HG	7
A HO HG	3
HO A HG	3
HO A HG	5
HO A HG	5
HO A HG	7
A HO HG	3
HO HG A	5
HO HG A	3
HO A HG	3
HO A HG	5
HO A HG	3
(A HO) HO	7
HO A HG	7
A HO HG for usability, HO A HG for fun	3
HO A HG	7
HO A HG	7
(HO A) HG	7
A HO HG	7
HO A HG	7
HO HG A	7
A HO HG	5
HO A HG	5
A (HG HO)	7

Why:

HO:

- interesting. It's like a game more than the other modes, because you're completely in control.
- easier deciding without computer making weird paths
- HO was fine, but a lot to keep track of
- didn't like human only because had to give so much detail
- because I could predict what would happen. I like the human one more because it had less waypoints
- because I didn't have to try to predict what the computer was going to do, I think I was more aware of all the UAVs because I set all their paths instead of letting them do what the algorithm did. In manual always knew where the UAVs were.
- it felt more natural, I'm used to micromanaging things
- HO mode, because there are so many vehicles [7], I have to locate the positions of both the vehicle and the target which takes more time, so takes more time for vehicle to start moving
- because I knew where they were going and the computer wasn't trying to move them for me
- more control, didn't have to wait if took a long time to generate a path
- I had control over what was going on, it didn't do stupid things
- it was nice. the best thing to do seemed to be to get it going, and then modify the path from there
- I like human-only mode because I like having direct control of everything
- because if you have computer generate a straight path then you can just modify it
- in the other two modes, it generated longer paths than was necessary
- the algorithm took a long time, and often was not most efficient way to do it. human mode is really quick.
- I liked that one too. I liked the ones with waypoints.
- because the path finding algorithm seems very erratic.
- HO gave me the most control.
- I thought that was the middle one, in terms of hardness. It seemed that you could more precisely control things that way, but it was still more work.
- in human only, even though bigger turns, easier to manipulate because didn't have to delete points
- it seemed a lot easier. it was less frustrating, easier to construct the paths.
- because the other ones either took too long or did weird stuff
- what i liked about human mode was you could hit generate path to make a straight path to get it moving. I only started taking advantage of this in the second half. at first i would make the waypoints first. that was nice because you could really easily modify on the fly. if an obstacle popped up you could drag the nearest waypoint and guide them around it.
- it was easier for me to adapt, so when you click generate path it generates a straight line so you can immediately start the UAV and modify the waypoints so they don't go into obstacles. and it was easier to do that on the fly. when you see that it's going to run into something I can divert it to the side and that gives me time to figure out the best path. so it was easy to generate path, start moving, and that was easy to minimize the amount of time I spend on it.
- I felt like I had more control of it.
- I thought manual was way faster.
- because I think I can figure out a very straight forward path quickly, rather than waiting for computer to generate, and it's easier to modify the path.

- with HO it just gave you a straight line and then you had to change it if it hit obstacles. To start A was faster. I remember thinking HO was really slow. But once things got spaced out in HO I didn't have to deal with as many vehicles at once. If I had fewer vehicles to control I think I would prefer HO mode.
- it was fine. I found it to be less work than the wayarea one.
- The other modes were slow because I'd wait for the computer to generate paths that were not necessarily that good and in order to fix the paths took a lot of effort.
- I could make efficient paths pretty quickly just by doing a straight line and one or two waypoints. I didn't have to count on it taking a long time to generate a path.
- Requires more attention and more multi-tasking but I found [HO and A] equally challenging I think.
- it was easy to get a direct path to start on and then modify it

HG:

- I didn't like it so much. I don't like circles. I don't like that it's not precise. It felt wrong that would go places where there were no circles, and could not change that.
- RRT modes made strange paths
- the way that the algorithm works, maybe I didn't fully understand how it worked, the solutions didn't seem like they would work very well.
- best of both worlds, with HG could give specifications to algorithm, hey go here, saved time of connecting dots but would still come up with good path of avoiding obstacles
- it's sad because it doesn't follow my suggestions
- had hardest time with the HG one, and should have rated the frustration level higher than medium. I had more trouble guiding it.
- because had to click multiple times until generated something sensible, made bad paths.
- HG was pretty difficult to work with mostly because were a lot of cases when would take a long time to find path or wouldn't find path at all, and when you're restricted to only confirming a path when you have a path, that can get difficult
- because it's not possible to modify path after generated, and regenerating is slower, it takes a long time
- human-guided was frustrating because I kept making little circles that were hard to erase
- wasn't sure how to effectively use wayareas
- I didn't like it. When it did something silly, there was no way to fix it. It wasn't good about following the wayareas I gave it.
- it seemed the least intuitive, but it was probably one of the more powerful ones, once the user develops a firm grasp for it
- in HG, it's hard to adjust quickly enough
- Human-guided mode I didn't like because it made paths I didn't like. Human-guided might be better if it aimed closer towards the center of the circles.
- it was interesting, but I didn't really see a difference between the paths when I made a small circle and a large circle, and it was easier to make small circles so I did that. Also I like being in control.
- I thought that was really hard to use, because it generated a lot of paths that didn't seem to be the shortest. It was much less intuitive with the wayareas.
- human guided was by far the worst. The wayareas didn't really work for me. It made me fly through four obstacles just because it couldn't find a better way, and there was clearly a better way.

- It was harder to control, it seemed random. I didn't know that you could generate a path again and again until half way through.
- HG was interesting. it seemed to really take care of things that made me not worry too much about what was going on, except when it did not find a path and decided to go in a straight line.
- The last one (HG) was workable. I liked it better than the second one, but my frustration is higher because I had to hit the button more often. I would eventually get it to go straight in a reasonable amount of time.
- HG involved less micromanaging than A.
- it was the hardest. I preferred dragging points around to making the little circles in various places.
- took too much time to figure out how to drop circles, and wasn't sure how solution is done. I thought the path would go through middle of circle, but then the path would squiggle through them instead of taking optimal path
- I didn't like human guided. I didn't know how to change the paths. Sometimes I would choose sizes that were larger than I intended. it was a lot easier to work with points.
- you always want it to be as precise as possible, so why not just click it. I didn't like dragging because dragging took too long.
- I thought it was going to be a lot easier than it was to use. What I disliked the most about it was how it takes the user a level back in the degree of control they have because you don't have waypoints. you're saying go in this general area and you're trusting the algorithm to come up with a good path that goes through where you want it to go. you have to spend the time to already say i want you to go here and here and then spend more time for the computer to figure out how to do it. I'd rather just take the same amount of time or marginally more to drop waypoints and not have to wait for the computer. particularly if it comes up with a crazy path then you have to wait for it to do it again. if it was instantaneous maybe it would be ok, if i didn't have to wait, if the processing power was quick enough. but if it takes a few seconds I want to spend that time dropping waypoints and know it's going to be a really good path. And I want the ability to guide them around obstacles by dropping a waypoint. in HG mode you would have to say I want you to go in the circle area and it would recompute and it's possible it would be crazy. and the time it took to compute the solution was not short enough for me to be willing to sit and wait for a solution. in general the lack of control -- I didn't have the ability to set a waypoint and that loss of control was what I didn't like about it.
- what was frustrating was that the wayareas -- was frustrating when algorithm went through edge of circle and did couple of loops on the side, so had to be very precise when defined wayareas. and also when tried to modify en route it changed the entire path rather than just the segment I wanted to tweak.
- HG was very hard to use because you would make wayareas that you didn't know about, or it would go through wayareas but not the way you wanted it to.
- It was the least intuitive. I don't know how it's done in real life, but guiding UAVs by giving them wayareas where they can choose anywhere in the wayareas is something not intuitive for me. And editing the paths using wayareas is even less intuitive for me.
- In human guided mode it is very difficult to edit the path once the UAV starts flying because there are no waypoints, only circles, and it takes time for the computer to calculate the path. A few times I flew into obstacles and I had to way to change the path.
- maybe I didn't understand it, but at the beginning I was trying to get 4 to choose a path and it didn't, and making circles didn't help either, so for time's sake I gave up and made it go through the obstacles. It just wouldn't choose a different path at all. I found it harder to avoid the

obstacles so I let the vehicles get more damages. And there were times when obstacles popped up and there was no way I could avoid them. I didn't find it as intuitive as clicking waypoints. Most of the time it would find a path that I was ok with and I didn't end put choosing more efficient paths. I just accepted it and moved on because I felt less in control so as long as it wasn't hitting things I was happy. In the other two modes I was more focused on being more efficient, getting speed.

- The wayareas you don't have to make wayareas but I feel like the path planning isn't as good in that one.
- About the same as A. Sometimes the computer found good paths through obstacles when it would have taken me a while to click that in HO mode. But most of the time it took a long way around.
- next favorite one. It would give me a default path and it was pretty easy to edit.
- One UAV got stuck and couldn't find a path, but the rest worked out, so it was fair.
- it was a little frustrating when it would take long to generate a path or when it seemed clear to me what path it should take but it would do extraneous loops. It was more useful to narrow paths. Sometimes it might take too many waypoints to get through an area, so in the human only mode I would make a UAV go around the long way around obstacles to get it moving.
- it was ok. If you disabled wayareas below a certain small size that would make it easier because there were times when I accidentally put in really small wayareas that were pretty much invisible.
- I liked in HG how the UAVs could be going somewhere without crashing into stuff, but waypoints were easier to deal with. I made really small wayareas so they acted like waypoints.

A:

- I like to fix the paths, I like the interaction more. I like to see what the algorithm comes up with.
- RRT modes made strange paths
- it would make the plan and then I could modify it, and the modifications were a lot easier than the HG one.
- problem with automation, had to go back and do manual editing to get rid of excess waypoints
- Extra waypoints annoyed my sense of aesthetics. It's a beautiful line with stuff in the middle.
- for automation one I could assume it would do something decent
- because I didn't have to enter waypoints, I just had to remove them, and I liked the fact that could pick the path, start moving so not waste much time, instead of first inputting and then also tweaking.
- automation would have been equivalent if it didn't put so many waypoints so it didn't need to make many modifications to tweak it to path that I want.
- because it can provide a solution so that I can get the vehicles to move quickly, and then I can modify the solution if anything happens
- automation mode was ok, except it put too many waypoints so I would delete them all
- liked automation mode, could get an estimate of path
- it gave me some waypoints, and I could move them around
- it made a few points but I could modify it
- compared to HG, the path finding was more responsive. compared to HO, it's faster to optimize it to see if there's anything shorter, and you can quickly find other possible paths.
- automation is good because I can start it and then fix it

- it was nice because you didn't have to spend so much time worrying about the UAVs, but the paths that the computer made were suboptimal and it was hard to change them when I wanted to.
- automation was also ok, but I felt I had to unclick a lot of what the computer did to make the paths shorter.
- It was ok because I could modify the waypoints when I needed to.
- because it was easy, it generated paths by itself, then it could go and you could change it.
- The automated one seemed to find the craziest paths. From my experience with it, it made me care too much about fixing everything.
- automation was really good and efficient, except there were too many waypoints. it made a nice path but it was hard to customize. I wanted to delete multiple waypoints and I couldn't. In A it would generate really abnormal paths that were close enough that I didn't want to generate a whole new path, but not optimized enough so I'd go in and delete several waypoints, and that took a lot of time.
- it seemed to do a lot of the thinking for me, I just had to make little adjustments. It was just tweaking and looking for yellow to pop up.
- too many waypoints in Automation, so harder to adjust path
- I liked the automation
- it was ok, it wasn't as bad as the last one with the areas
- The automation mode was kind of nice that it would give you a pretty good first path, but it was annoying the time you had to wait for it. and also it was kind of annoying sometimes that it would have a string of waypoints along a straight line because you had to remove all the waypoints to change it sometimes. but it was nice because at least you could hit generate path and confirm and start it moving and it probably wouldn't hit anything.
- it worked well except the automation generated a lot of waypoints so if I had to modify it, there was a lot I had to delete.
- it was pretty easy to use too. it wasn't as fun because you had to hit a button to make the paths. it was easier to use than HG.
- wasn't bad but sometimes it took too long to find a proper route. And often the first route it generated wasn't the shortest route so I had to regenerate it.
- It was good. If you can let humans choose whether first to let computer generate path or draw path by themselves, I think it would be better. Or if computer starts calculating and you find out it's too slow, can click a button "stop" and draw your own path.
- One thing I liked about A was that the first time you generated a path it gave you a semi intelligent path right away so you could confirm and then change it. To start A was faster. But once things got spaced out in HO I didn't have to deal with as many vehicles at once.
- because I didn't have to do as much work.
- The algorithm wasn't very efficient.
- Least favorite. I didn't like having to keep refreshing the paths. It was kind of frustrating to click it several times for it to make the optimal path.
- it generated almost perfect routes, and the level of frustration was the lowest in that. I just had to make small modifications.
- I liked that too.
- it was ok. It would have been easier if there were a way to delete a lot of waypoints at once.
- with human only when generated paths it just generated a straight path but this one would generate paths that wouldn't crash into anything, so I could immediately start UAVs moving and

then adjust the paths. In A, I generally deleted extra waypoints, then moved the rest around instead of making new ones.

Suggested changes in interface:

- I would make the UAVs look more like planes and would direct them in the direction of motion. It would make it more interesting if varied the sizes of the obstacles. Would make movement faster or make it change throughout the game.
- make the questions flash brighter, sometimes clicking on UAVs didn't work
- it was pretty straight forward, one click is nice to change waypoints, nice that can click to add anywhere
- wouldn't mind having an auditory ding when new question, kept missing secondary task. nothing major otherwise. maybe blink obstacles because some didn't register when turned yellow. understood role of circles in HG, didn't play around with it enough to know exactly where sensitivity was, would want to play around with circles. everything else fine.
- I would make blank background because I need to ignore it. I would have the UAVs bigger or smaller than the targets so they would be easily distinguishable. Also when click on UAV I sometimes need to click a few times before it registers. When I type in secondary answer I want to know if correct. Didn't know what's more important, deleting waypoints or working with other path. Do I care more about fuel or time.
- since all paths are black, sometimes it was hard to detangle all the paths, wasn't as easy to determine whose path I needed to alter if there was an obstacle.
- have auditory feedback when question pops up. not sure why would need to choose size of circle, seems like if input waypoint would want to go through waypoint, sometimes it would take time thinking and you couldn't make changes in HG mode. I liked the timeline. A: I liked it. Maybe I'd have a setting to determine number of waypoints on path, sometimes too many waypoints.
- sometimes obstacles appeared very suddenly, it would have been nice to have warnings about when things would be changing
- whenever obstacle appears as yellow, need to figure out which path causes problem and go back and locate vehicle, so if I could click on path instead of vehicle, it would be helpful
- I would use number shortcuts
- not much
- If there was a way to not have to switch tabs to view info about UAVs that would be helpful. It would be nice if could glance at all info and see if one of them had a problem. I really wanted to select a path because there were many paths overlapping. When a path went through an obstacle had to trace it back to the UAV.
- would add keyboard shortcuts to select UAVs
- hot key bindings to select UAVs
- keyboard shortcuts for UAVs and for the text box, touch screen. Might be better to say that UAVs are really just the point in the center, sometimes it would go close to an obstacle and technically it wasn't a collision because it wouldn't turn yellow but it made me nervous
- sometimes it was hard to figure out which UAV was following which path if the paths overlapped, and I didn't know what to do if a UAV entered an obstacle. I liked that the obstacles turned yellow and the UAVs started blinking when it was time to move them. Sometimes it was hard to figure out where their targets were.
- The only complaint is the stats were on the side. If they were on the bottom where I could see the distance to the target, it would be good. I would combine them.

- I liked it, it worked pretty well.
- I liked it. Maybe more colorful, the green was kind of annoying. If it was brighter, you would see it better. The waypoints were really small.
- it seems perfectly reasonable. It was hard for me to tell whether an obstacle appeared just before a UAV entered it, or it if appeared on top of it. So I was not sure if I wasted time trying to get out of obstacles when I did not have to. But the interface itself seemed fine.
- add hotkeys for 1,2,3 to switch between UAVs easier, and add fourth hotkey for entering answer to question instead of clicking on box. Also being able to delete lots of waypoints at once, like dragging a box. Also it would be nice if the algorithm presented several possibilities and the operator could click on the one he likes best.
- The interface is fine.
- would put details info somewhere more central instead of off to the corner
- I didn't like the wayareas.
- hot keys, maybe sound, to warn of collisions
- The timeline was nice. I definitely used that to see if I have a lot of time to answer questions and also what should I be paying attention to next. I occasionally missed for a few seconds when a UAV had reached its target. it would be nice if there was a visual queue of UAVs that are waiting at their destination. it would be really nice to select a UAV by clicking it in the timeline. The flashing was useful. Maybe more contrast in colors. It would be nice to know which direction the UAV's are pointing. Its only really an issue when they're waiting at a destination and it is closely packed by obstacles, but it would be nice to know where it will go immediately upon moving. Also, maybe there's an obstacle directly between me and the destination, I'd rather keep the UAV moving in its current direction rather than spending time turning around. Currently you'd have to remember what path the UAV came in on in order to know that information.
- an easier way to see when something is waiting for a new path. if the software could auto generate straight lines between UAV and goal so you know which direction needs to go and from there can modify it. the tabs were nice to see information on each UAV.
- having UAVs immediately going to next target (straight line in HO, for others generate path using algorithm), and also instead of confirm being able to hit enter, more keyboard inputs.
- targets could be bigger, because in human guided you need to know where the target is, and it was hard to see. more contrast in colors. I think red is better color than yellow for collision. yellow and green doesn't quite jump out at you. in timeline, once UAVs finish path they disappear from timeline. If they could stay at 0 that would be nice. I was attempting to see in timeline if UAV had finished its path.
- maybe when I put my mouse pointer on a waypoint you could give it some effect like enlarge, to make it easier to move around. And generate path or confirm should have keyboard shortcuts. Maybe if a UAV comes to an obstacle it could stop and wait for you to give it a new path. If you can let humans choose whether first to let computer generate path or draw path by themselves, I think it would be better. Or if computer starts calculating and you find out it's too slow, can click a button "stop" and draw your own path.
- One thing I would like is when a task is completed, flash it a different color, I thought the green and orange were too close. Maybe green and red. They didn't pop out to me enough. If I was trying to answer the questions on the side, I thought it was annoying to switch through the tabs, but I don't know a better way to do that. At the start, it would be nice if all the vehicles generated a path and if it's clear of obstacles just to start going on it, and if not then wait for my input. It took me a long time to go through all the vehicles, select a path, make sure it's not going to hit anything, then go.

- have number key pad like on desktop keyboard for answering the questions.
- It was easy to use. Sometimes I didn't notice when UAVs finished. They were flashing but I didn't notice them. Maybe make it more apparent. Sometimes I confused generate path and confirm. It would be nice to be able to stop a UAV.
- A couple times it wasn't clear when I selected something.
- change colors -- give more contrast between blue-green, yellow, the waypoints are yellow too, maybe make them red, and grey background, so have to pay attention and kind of hard to see on the map.
- not really. maybe in guided mode to have an option to input human only path if it doesn't find one quickly enough.
- questions: make it so don't have to select text field, when type it should select it automatically. maybe prevent it from changing while there is text in the field.
- Maybe in A and possibly HG, have a button to make a straight path.

Enough training:
Yes. Yes. Yes. Yes. Yes. Yes. Yes. Yes. Yes. Yes. Yes. Yes. I felt like I needed more, especially for wayareas. It was the strangest mode, I feel like something could be done to emphasize it a little more. Yes. Yes. Yes. Yes. Yes. Yes. Yes. Yes. Yes. Yes. Yes. I guess, I didn't remember which mode was which. Yeah I think so; I could have spent more time on HG mode so I was more comfortable with the limitations of that setup; I didn't get a chance to see what would be the best way in HG mode to guide a UAV through a wall of obstacles with a narrow passage through it. Yes. I think so, maybe a little more would help. Yes. Yes. Both HO and A are really intuitive, they did pretty much what I expected them to; the HG I felt like I didn't know how to use the circles. Yes. Yes. Yes. Yes. Yes. Yes.

Video game experience:
Yes. No. A little. Yes. No. No. Yes. Yes, lots of real-time strategy games like Starcraft, they involved lots of micromanaging. No. Some. Some. No. Yes. Some. Yes, a lot – Starcraft, Warcraft, RTS's. No. Some. A little bit, strategy games. No. Yes, every type except for shooters. Lots, as well as being game designer – shooters, strategy games, action RPG's, occasional RPG. A little bit – Portal, Mario Brothers, Starcraft. Yes – hockey, first person shooters. A little, FPS. Yes, quite a bit – mostly strategy games, Starcraft, where you also have to click units. Yes – first person shooters, strategy games like Starcraft, Age of Empires, RPG's like Skyram, lately on console, but also PC. Yes, primarily first person shooters, couple of role playing and also Civilization, Starcraft. No. Yes, strategy, first person shooters, third person, RPG's. Yes, first person shooters, Starcraft, Red Alert. Yes, first person shooters, strategy – Age of Empires. A little bit, RPGs. Yes, RPGs, flash games, fighting games. Yes, command and conquer games, Halo, sports games, first person shooters. Very little. Yes, all kinds, a lot of flight simulators, puzzle games, rhythm games (like DDR). Yes, real time strategy.

Field:
Physics
linguistics and English
HAL
HAL
Math education
HCI
HAL
HAL

HAL
Mechanical Engineering or EECS
EECS
HAL, aero/astro
CS
EECS
CS
Prosthetics and orthotics
Math and EECS
EECS or aero/astro
CS and environmental engineering
EECS
CMS
EECS
Mechanical engineering
Aero/astro
Aero/astro
Aero/astro
Mechanical engineering
Aero/astro
Aero/astro
Aero/astro
Ocean engineering
CS
Aero/astro
Aero/astro
Aero/astro
Materials science and engineering

References

[1] "Flight of the Drones". *The Economist*. October 8th, 2011.
Available: http://www.economist.com/node/21531433

[2] 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.
Available: http://web.mit.edu/aeroastro/labs/halab/papers/ICCRTS_I-171_Nehme.pdf.

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

[4] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning", *International Journal of Robotics* Research, vol. 30, iss. 7, pp. 846 – 894, 2011.
Available: http://sertac.scripts.mit.edu/web/wp-content/papercite-data/pdf/karaman.frazzoli-ijrr11.pdf

[5] 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.

[6] A. Caves, *Human-Automation Collaborative RRT for UAV Mission Path Planning*.
M. Eng. thesis, Massachusetts Institute of Technology Department of Electrical Engineering and Computer Science, June 2010.
Available: http://web.mit.edu/aeroastro/labs/halab/papers/americoThesis.pdf.

[7] D.Wagner and T. Willhalm, "Speed-Up Techniques for Shortest-Path Computations", in *Proceedings of the 24th International Symposium on Theoretical Aspects of Computer Science*, pp. 23 – 36, 2007.

[8] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*. 2nd Edition. Cambridge, MA: MIT Press, 2001.

[9] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime Search in Dynamic Graphs", *Artificial Intelligence*, vol. 172, pp. 1613 – 1643, 2008.

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

[11] 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.
Available: http://web.mit.edu/aeroastro/labs/halab/papers/CummingsMitchell07.pdf.

[12] 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.
Available: http://web.mit.edu/aeroastro/labs/halab/papers/marquez_thesis_final.pdf.

[13] G. Carrigan, *The Design of an Intelligent Decision Support Tool for Submarine Commander*, S. M. thesis, Massachusetts Institute of Technology Engineering Systems Division, June 2009.

[14] M. Cummings, M. Buchin, G. Carrigan, and B. Donmez, "Supporting Intelligent and Trustworthy Maritime Path Planning Decisions", *International Journal of Human Computer Studies*, vol. 68, iss. 10, pp. 616 – 626, October 2010.

[15] 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.

[16] D. Norman, *The Design of Everyday Things*. 2002.

[17] 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.

[18] 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.

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

[20] M. L. Cummings and S. Bruni, "Collaborative Human-Automation Decision Making", in *Handbook of Automation* (S. Y. Nof, ed.), pp. 437 – 447, Springer, 2009.