Optimal and Receding-Horizon Path Planning Algorithms for Communications Relay Vehicles in Complex Environments

by

Karl Christian Kulling

S.B., Aerospace Engineering
Massachusetts Institute of Technology (2007)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Author .....
Department of Aeronautics and Astronautics
May 22, 2009

Certified by .....
Jonathan P. How
Professor
Thesis Supervisor

Accepted by .....
David L. Darmofal
Associate Department Head
Chair, Committee on Graduate Students

Optimal and Receding-Horizon Path Planning Algorithms for Communications Relay Vehicles in Complex Environments

by

Karl Christian Kulling

Submitted to the Department of Aeronautics and Astronautics
on May 22, 2009, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

This thesis presents new algorithms for path planning in a communications constrained environment for teams of unmanned vehicles. This problem involves a lead vehicle that must gather information from a set of locations and relay it back to its operator. In general, these locations and the lead vehicle's position are beyond line-of-sight from the operator and non-stationary, which introduces several difficulties to the problem. The proposed solution is to use several additional unmanned vehicles to create a network linkage between the operator and the lead vehicle that can be used to relay information between the two endpoints. Because the operating environment is cluttered with obstacles that block both line-of-sight and vehicle movement, the paths of the vehicles must be carefully planned to meet all constraints. The core problem of interest that is addressed in this thesis is the path planning for these link vehicles. Two solutions are presented in this thesis. The first is a centralized approach based on a numerical solution of optimal control theory. This thesis presents an optimal control problem formulation that balances the competing objectives of minimizing overall mission time and minimizing energy expenditure. Also presented is a new modification of the Rapidly-Exploring Random Tree algorithm that makes it more efficient at finding paths that are applicable to the communications chaining problem. The second solution takes a distributed, receding-horizon approach, where each vehicle solves for its own path using a local optimization that helps the system as a whole achieve the global objective. This solution is applicable to real-time use onboard a team of vehicles. To offset the loss of optimality from this approach, a new heuristic is developed for the linking vehicles. Finally, both solutions are demonstrated in simulation and in flight tests in MIT's RAVEN testbed. These simulations and flight tests demonstrate the performance of the two solution methods as well as their viability for use in real unmanned vehicle systems.

Thesis Supervisor: Jonathan P. How
Title: Professor

Acknowledgments

I would like to thank several people for their support during my two years as a graduate student. First, I would like to thank my advisor, Prof. Jonathan How, for his support and guidance that were essential to completing this work. I would also like to thanks Aurora Flight Sciences for establishing the Aurora Fellow program and having me as the first fellow in this new program. At Aurora I would like to specifically thank Dr. Jim Paduano for his guidance and for always giving me new problems to think about, and Olivier Toupet, with whom I've worked closely over these two years.

Additionally I would like to thank Kathryn Fischer for all her assistance and hard work, and Cameron Fraser, with whom I worked many long hours in the lab. My other colleagues in the Aerospace Controls Lab, Buddy Michini, Frant Sobolic, Dan Levine, Sergio Cafarelli, Josh Redding, Frank Fan, Brett Bethke, Brandon Luders, Sameera Ponda, Luc Brunet, and Andy Whitten were always available to help with work, to give advice, and to provide distractions. My friends Frank Johnson, Yann Heskestad, Chris Fennell, and Alyssa Anderson also deserve many thanks for their numerous visits to Cambridge and their friendship. Lastly, I would like to thank my parents for their inspiration and unwavering support.

Boeing also deserves acknowledgment for help in creating the Aerospace Controls Lab's RAVEN testbed.

THIS PAGE INTENTIONALLY LEFT (ALMOST) BLANK

Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT (ALMOST) BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT (ALMOST) BLANK

List of Algorithms

THIS PAGE INTENTIONALLY LEFT (ALMOST) BLANK

Chapter 1

Introduction

Unmanned Aerial Vehicles (UAV) have gained widespread use for military and civilian purposes. They are good at performing "dull, dirty, and dangerous" missions and thus are used for battlefield surveillance, border patrol, weather data gathering, and tactical surveillance, among other uses. These UAVs come in a wide range of sizes and capabilities from the 32,000 lb. Global Hawk [39], capable of carrying a radar for multi-day missions, to small "backpackable" models, such as the Aerovironment Wasp [1], used by small military units for tactical surveillance during combat.

The UAVs at the small end of this range, while generally limited to simple sensors and line-of-sight communication, can still be very useful. Small UAVs are increasingly being used by military units for "over the hill" or "around the corner" surveillance missions [34]. For example, imagine a tactical unit deployed in a city that wishes to see if an enemy ambush is setup several blocks away. A small UAV could be deployed and be directed to fly a route along the streets of the city while relaying back to the unit a real-time video feed from an onboard camera. This has the advantage that operators can remain out of harms way while gathering information. However, many small UAVs used for these missions can only communicate to the operator using a line-of-sight communications link. In this case, the range of the surveillance UAV would be severely limited, possibly to the point where it is no longer useful. However, if additional UAVs could be deployed between the lead UAV and the unit, then the effective range of the system can be greatly extended. Additionally, the multiple

UAVs can provide redundancy in case one UAV fails. The same limitation applies to expeditionary units performing operations in mountainous terrain. In that case, the unit may wish to conduct surveillance down a winding canyon or over a ridge without exposing itself to the enemy. Once again, if the UAV goes out of communications range, then the unit cannot receive real-time information or dynamically re-task the UAV.

In fact, one problem that plagues small UAVs is that many are lost due to loss of communication [9]. If a simple UAV flies outside of communications and command range, then it might not be able to find its way back to its base. While this may not be an expensive loss, it is a waste of resources. A unit would need to carry many small UAVs to account for the potential losses. However, an alternate solution is to put those additional UAVs to use in forming a communications link back to the user, thereby reducing the probability that UAVs are lost in the first place. Additionally, having constant communication with the vehicle allows for real-time re-tasking of the vehicle based on information received during the surveillance flight.

Of course, these problems are not limited to aerial vehicles. Ground robots have applications in exploring buildings [31] and the US Navy has been experimenting with unmanned surface vessels for patrols and reconnaissance [12]. Furthermore, teams of unmanned vehicles need not be limited to one class of vehicle, but can be composed of vehicles from various classes. For example, a ship-launched UAV can work together with an unmanned surface vessel.

These examples motivate the need for an unmanned vehicle deployment strategy that provides a communications link, or chain, between a lead vehicle and its operator. In fact, this role has already been identified for UAVs [33] and is certainly applicable to other classes of unmanned vehicles. However, to be useful, these vehicles need smart algorithms to guide them, especially in the case of small vehicles operating near the surface.

1.1 Objectives

This thesis will examine path planning algorithms for coordinating a team of unmanned vehicles that is required to maintain a continuous communications link to an operator. Two classes of problems will be investigated. The first, discussed in Chapter 2, considers the problem where all the vehicles start at a common base location and the lead vehicle needs to reach a known, fixed target location. In this *deployment* problem, a single solution is computed and then implemented by the team of vehicles. The second class of problems, discussed in Chapter 3, considers the case where the lead vehicle's target location moves incrementally, such as would be the case if it is following a moving vehicle, or if the operator directs its surveillance path in real-time. In general, the algorithm used for this class of *reconfiguration* problems will try to achieve an optimal vehicle configuration over a short time horizon for which the movement of the lead vehicle is known. The main focus of this second approach is flexibility. The vehicles position themselves in such a way that they can best react to the movement of the lead vehicle. These two approaches differ slightly in their goal, but nevertheless address the common issue of providing a real-time communications link to an unmanned vehicle.

1.1.1 Deployment Problem

In the deployment problem, the goal is to find the optimal path for each vehicle from its common starting location to a final location as determined by the algorithm, except for the lead vehicle, which must reach a specified target. The cost function for this problem attempts to minimize a combination of mission time (the time until the lead vehicle reaches its goal) and the total energy used by all the vehicles. The problem fits nicely into the framework of an optimal control problem. Formulating the problem as an optimal control problem allows the non-linear vehicle dynamics of the vehicles to be incorporated as dynamics constraints, but also, since the optimization is solved as a centralized problem, all inter-vehicle coupling constraints (such as line-of-sight constraints) can easily be incorporated as path constraints. Overall, the

optimal control framework is a flexible way of posing a problem that avoids a fixed discretization that many path planning techniques use.

A pseudospectral method was chosen to solve the optimal control path planning problem. Pseudospectral methods are direct numerical solution methods for solving optimal control problems; they approximate the states and controls by a basis of global polynomials, which are determined from an optimally-chosen set of discretization points [5]. The problem is transcribed into a nonlinear program, which can be solved by an available nonlinear optimization package. Specifically, a Gauss pseudospectral method was chosen because it has been shown to efficiently solve optimal control problems with path constraints, including multi-vehicle path planning problems [2, 22].

To solve problems quickly, a pseudospectral solver needs to be initialized with a good initial guess for the state and control trajectories. To obtain this initial guess, a Rapidly-Exploring Random Tree (RRT) [3, 13, 14, 28] is used to search for a path for the lead vehicle from the operator to the goal, and then heuristics are used to fill in the rest of the trajectory. A RRT provides a fast, probabilistic way of searching for a path through a high-dimensional space. It has good exploration characteristics because it biases the search towards the unexplored area of the search space. Furthermore, RRTs have been well studied in the literature and have been applied to many different problems. Thus, many useful extensions of the original algorithm have been developed, some of which will be used in this thesis.

1.1.2 Reconfiguration Problem

Unlike in the deployment problem, where the focus is on achieving optimal deployment of a team of vehicles to a known target location, the reconfiguration problem focuses on a team of vehicles that is used in an uncertain situation and which must react quickly to new information or new commands. Such situations would typically be encountered in the scenarios discussed previously, for example the surveillance of a street in a military situation. Ideally, this real-time algorithm also scales well in the number of vehicles so as not to be limited to just a few vehicles.

The requirement to be scalable naturally drives the problem formulation to a distributed formulation where each vehicle solves a local optimization with some exchange of information between adjacent vehicles. This approach is scalable both from a computational perspective and from a communications perspective. If the algorithm would require global communication with all vehicles in the team, then the communications requirement (per vehicle) would increase with the team size, which is undesirable. Additionally, since the team has a linear communications structure where the degree of each communications node is at most two, requiring vehicles to pass messages to vehicles far away on the chain would cause increasing messaging delays as the chain length grows. As with many systems, communications delays could cause instabilities in the chain of vehicles [29]. Since, in the reconfiguration algorithm presented in this thesis, each vehicle only exchanges information with two vehicles (the one ahead of it in the chain, and the one behind it), the algorithm scales linearly in the number of vehicles[1].

The general approach taken by this algorithm is to have each vehicle optimize its position so as to facilitate the task of the vehicle ahead of it, with the exception of the lead vehicle, which is trying to minimize the system's overall cost function. If each vehicle does this, then the positive effect of each vehicle's actions propagates forward and benefits the lead vehicle. This optimization is performed in a sequential manner, where each vehicle waits to perform its optimization until the vehicle ahead is done performing its optimization. This allows each vehicle to use the short-term planned path of the vehicle ahead in its own optimization. Of course, once a vehicle is done calculating its own short term path, it can immediately start implementing that plan without waiting for all other vehicles behind to finish. If this sequence is performed regularly, then the vehicles will continually adjust their position and the team as a whole will move towards the desired goal.

The optimization that each vehicle performs optimizes its own position relative to the vehicle ahead of it in order to provide the best communications link with the greatest flexibility for future movement of the vehicle ahead. This approach reflects

[1]Presumably, so does the processing power of the system

that the algorithm as a whole is acting on short-term information but is essentially "hedging its bets" and preparing for the vehicle ahead to move in any direction. This heuristic provides the best general performance given the unpredictability in the movements of the lead vehicle, and subsequently the link vehicles.

1.2 Literature Review

There are many possible architectures for creating communications links, including mesh networks [7], hierarchical backbone networks [8, 38], and linear networks. This thesis will focus on this last type of network where a single lead vehicle requires a persistent, real-time communications link to a single operator.

In fact, this type of mission has been studied by previous authors. In [32], a team of tactical ground robots exploring a building was studied. The linking robots follow the lead robot until a communications link is about to be broken. At that point, the rearmost robot in the convoy stops at its current position and acts as a communications relay. This approach, while simple, is not very flexible. There are no provisions for a link robot to readjust its position in response to the movements of its adjacent teammates. This inflexibility can lead to inefficient configurations.

A solution using Mixed Integer Linear Programming (MILP) was studied in [36, 37]. This approach discretizes time and the state space and then searches for the optimal solution subject to constraints such as avoiding obstacles and maintaining a clear line-of-sight to adjacent vehicles. To reduce computational complexity, this optimization is performed in a receding horizon framework, where the optimization is only performed for a few time steps, up to a planning horizon. It is then repeated periodically, thereby building up a solution in small steps. The authors studied both a centralized problem formulation as well as a decentralized problem formulation, where each UAV performs its own local optimization. The main drawback of this approach is that solving a MILP is computationally intensive, especially in the centralized problem formulation.

Dixon and Frew have also studied communications chaining [10, 11]. However,

their main focus was on planning and control in realistic RF environments and sensing this environment online with the communications equipment onboard their UAV. They did not address the issue of obstacles in the operations area, which affects both communications and path planning.

Holmberg and Olsson studied link UAV path planning by using constructive solid geometry to determine the common field-of-view shared by a lead UAV and a link UAV [17]. Their algorithm addresses the deployment problem and calculates the path for the link UAV based on a given lead UAV path. The environment is described by the surfaces of the obstacles, so as the number of obstacles grows, so does the number of plane intersection computations that must be performed. Also, their algorithm does not address the optimality of the solution and relies on a human operator to accept and modify the planned route.

Ibrahim, Seddik, and Liu addressed maintaining connectivity in wireless mesh sensor networks by using relay vehicles [23, 24]. They determined the strength of the network using the Fiedler value and analyzed the increase in this metric by adding a relay vehicle. They showed that adding even one relay vehicle can increase the Fiedler value by 35%.

Using Gauss pseudospectral optimization methods with an initial solution created from a Rapidly-Exploring Random Tree (RRT) search was studied by [2]. That work also addressed multi-vehicle path planning with coupling constraints. However, that author's specific approach performed the RRT in the full configuration space, whereas this thesis performs a simplified RRT search and fills in the missing states and controls with a heuristic. Pseudospectral optimization and specifically Gauss pseudospectral optimization has been studied by several authors [5, 19].

Sequential optimization, used in the solution to the reconfiguration problem was studied in [26]. That work looked at multi-vehicle path planning where each vehicle has its own dynamics but also has constraints that couple it to other vehicles. This thesis uses a similar distributed and coordinated optimization framework.

1.3 Thesis Overview

Chapter 2 discusses the algorithm developed for the *deployment* problem and presents several simulation results. The main contributions of this chapter are the formulation of the communications chain path planning problem as an optimal control problem and the modification of the Rapidly-Exploring Random Tree algorithm to provide an initial solution to this problem. This initial solution is an important part of the Gauss pseudospectral optimization. The simulation results show typical results produced by the optimization and also show various interesting properties of the optimal solution.

Chapter 3 presents the algorithm developed for the *reconfiguration* problem. This algorithm is applicable to real-time implementation on a team of unmanned vehicles performing the mission described above. While the algorithm does not necessarily produce an optimal solution, simulation and flight test results show that the algorithm works well in practice. One main contribution of this chapter is the development of a new heuristic for the communications chaining problem when the path planner uses a short planning horizon. This algorithm can be used to augment the performance of unmanned vehicle systems currently in use by military and civilian users alike.

Chapter 4 discusses implementation details of both algorithms and presents simulation and flight test results. The simulation results show the reconfiguration algorithm's applicability to, and performance in, various scenarios that might be encountered by unmanned vehicle system. The flight tests show the behavior of this algorithm under actual disturbances and variations not modeled in the simulations. They also validate the applicability of the algorithm to real-time implementation. Lastly, this chapter also compares the reconfiguration algorithm to a modified version of the deployment algorithm.

Chapter 2

Deployment of Vehicles from a Common Base

2.1 Background

The deployment problem addresses scenarios where an operator wishes to conduct surveillance of a location that requires an unmanned vehicle to go beyond the operator's communications volume. The proposed solution is to use additional unmanned vehicles to provide a communications link between the lead vehicle and the operator. The vehicles that are being used are considered to be small vehicles that have only line-of-sight communications equipment and that can not communicate via satellite or very high altitude aircraft. The environment that the vehicles operate in can be mountainous or urban, which in either case contains obstacles. This precludes trivial solutions such as a straight line of link vehicles.

Because the target surveillance location is known, the path for all the vehicles can be computed once and then implemented, possibly with some local corrections, but without needing to create a new plan again. This assumption allows for a centralized mission planner that plans the path of all the vehicles at once. The advantage of this approach is that, because there are many inter-vehicle constraints and couplings, it allows the planner to better optimize the paths. Of course, because the vehicles maintain a strongly connected network, if a new plan needs to be uploaded to the

team, it can be disseminated through the established network.

2.1.1 Connectivity Maintenance

Maintaining a strongly connected network–a network where there is a directed path from any vehicle to any other vehicle–is important for several reasons. First, a key part of the problem is to provide a real-time data link from the lead vehicle to the operator. As evidenced by the multitude of real-time surveillance UAVs currently in use by militaries, this type of surveillance is of great tactical importance. Second, if the team remains strongly connected it remains possible to give instructions to the vehicles at any time rather than having to wait until contact is reestablished at some later time.

2.1.2 Path Planning as an Optimal Control Problem

Path planning problems can be written in the framework of a continuous time, finite horizon optimal control problem with path (non-dynamics) constraints as well as the usual dynamics constraints. For single vehicle path planning problems, the state vector is simply the state of the single vehicle and the control inputs are the inputs to this vehicle. However, the optimal control formulation is also applicable to multi-vehicle path planning problems where the state vector is the state of all the vehicles concatenated together, and likewise for the control inputs.

The optimal control problem has the cost functional

$$J = \phi(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)dt, \tag{2.1}$$

and is subject to the dynamics constraint

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \quad t \in [t_0, t_f], \tag{2.2}$$

the boundary constraint

$$\phi(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) = \mathbf{0}, \tag{2.3}$$

and the path constraint

$$\mathbf{C}(\mathbf{x}(t), \mathbf{u}(t), t) \leq \mathbf{0}, \quad t \in [t_0, t_f]. \tag{2.4}$$

This framework is flexible to account for all of the constraints in the communications link deployment problem, including the line-of-sight constraints. Of course, the problem formulation is a centralized problem formulation where the entire problem is solved as a single problem rather than divided into subproblems.

Gauss Pseudospectral Method

To use the continuous time, finite horizon optimal control problem formulation, a solution method capable of solving this problem must be chosen. The Gauss pseudospectral method (GPM) has been shown to work well for these types of problems [2]. Specifically, the Gauss Pseudospectral Optimization Software ($\mathcal{GPOPS}$) [5, 6, 19–22, 35] is the implementation used in this thesis. The GPM approximates the states and controls by a basis of Lagrange interpolating polynomials. These polynomials pass through optimally-spaced Legendre-Gauss discretization points [5]. The problem is then transcribed into a nonlinear program (NLP) and solved using SNOPT, a nonlinear optimization package [15].

2.2 Problem Formulation

This section will describe how the communications link deployment problem was formulated as an optimal control problem. The lead vehicle will use index $i = 1$, the link vehicle directly behind the lead vehicle will use index $i = 2$, and so on, until the last link vehicle, which uses index $i = N$. The communications base station is node $i = N + 1$. The set of obstacles in the environment that block vehicle movement as well as line-of-sight communications is $\mathcal{O}$ and the various obstacles will be index with index j.

The vehicles are modeled as holonomic vehicles with state $\mathbf{x_i} = [x\ y\ u\ v]^T$, whose

components are the position and velocity in two perpendicular directions. The control inputs are $\mathbf{u_i} = [F_x \; F_y]^T$, which are the force on the vehicle in the x and y direction. As mentioned previously, the states and controls of all the vehicles will be concatenated into one state vector $\mathbf{x} = [x_1 \; y_1 \; u_1 \; v_1 \ldots x_N \; y_N \; u_N \; v_N]^T$ and one control vector $\mathbf{u} = [F_{x,1} \; F_{y,1} \ldots F_{x,N} \; F_{y,N}]^T$. All the vehicles share the same dynamics given by

$$\dot{x}_i = u_i, \qquad \dot{u}_i = -c_d({u_i}^2 + {v_i}^2)\hat{u}_i + F_{x,i} \tag{2.5}$$

$$\dot{y}_i = v_i, \qquad \dot{v}_i = -c_d({u_i}^2 + {v_i}^2)\hat{v}_i + F_{y,i} \tag{2.6}$$

$$\sqrt{{F_{x,i}}^2 + {F_{y,i}}^2} \leq F_{\max}, \tag{2.7}$$

where c_d is a drag coefficient and $F_{\max}$ is the maximum allowable control effort.

The base station ($i = N + 1$) is located at (x_{N+1}, y_{N+1}) and does not move. The unmanned vehicles start near the base with some initial state $\mathbf{x}(0) = \mathbf{x_0}$. The vehicles move in the plane and are constrained to stay clear of obstacles, as well as meet other constraints, which will be introduced later. The lead vehicle ($i = 1$) is attempting to reach a target location (x_g, y_g) at a variable final time t_f. Part of the cost function will try to minimize this final time, which is the overall time to complete the mission. Also, the final locations of all the link vehicles are free. These vehicles will move to provide the required communications service while optimizing the overall cost function. Lastly, it is assumed that the set of obstacles $\mathcal{O}$ is known and given.

Objective Function The objective function used for this problem is

$$\min J = (t_f - t_0) + \alpha \int_{t_0}^{t_f} \sum_{i=1}^{N} ({F_{x,i}}^2 + {F_{y,i}}^2)\, dt, \tag{2.8}$$

where α is a tuning term that trades off between the two terms in the objective function. Equation 2.8 tries to minimize both the duration of the mission and the sum of the energy usage of the vehicles. These are competing goals, because due to the nature of the vehicle dynamics, specifically the drag on the vehicles, the faster a vehicle flies, the more energy it has to spend to travel at that speed. Thus, to conserve

energy, the vehicles would like to fly slowly, while to minimize mission duration, the vehicles would like to travel at their maximum allowable speed or control authority. Simulation results will show that the optimal solution carefully balances these two competing objectives.

Obstacle Constraints To simplify the encoding of the obstacle constraints into this problem formulation, the obstacles are approximated as circles. This allows these constraints to be written as

$$(x_i - x_j)^2 + (y_i - y_j)^2 \geq {R_j}^2, \quad j \in \mathcal{O}, \quad \forall\ i = 1, \ldots, N, \tag{2.9}$$

where (x_j, y_j) is the center of obstacle j, and R_j is the radius of that obstacle. This simplification does not preclude the use of more complicated obstacle shapes.

Line-of-Sight Constraints In general, the line-of-sight constraint would be written as

$$(1 - \lambda)[x_{i+1}\ y_{i+1}]^T + \lambda[x_i\ y_i]^T \notin \{\mathcal{O}\}, \quad \forall\ 0 \leq \lambda \leq 1, \quad i = 1, \ldots, N. \tag{2.10}$$

However, the line-of-sight constraint encoding is also simplified by the choice of circular obstacles. There is one unique point on the line-of-sight segment that is closest to the obstacle and if the distance from this point to the center of the obstacle is greater than the radius of the obstacle, then the line-of-sight is clear of that obstacle. Either, one of the two segment endpoints is the closest point to an obstacle, or the line from the center of the obstacle to the checkpoint is perpendicular to the line segment, as shown in Figure 2-1. Checking the endpoints uses an equation similar to Eq. 2.9, while checking a point in the interior of the line segment uses the formula for the distance from a line to a point, given in Algorithm 1, and then checking that that distance is greater than the radius of the obstacle.

Algorithm 1 Point to Line Distance

$\mathbf{dx} \leftarrow [x_i \ y_i]^T - [x_{i+1} \ y_{i+1}]^T$

$\mathbf{r} \leftarrow [x_{i+1} \ y_{i+1}]^T - [x_j \ y_j]^T$

$\mathbf{p} \leftarrow \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{r}$

$\hat{\mathbf{p}} \leftarrow \frac{\mathbf{p}}{|\mathbf{p}|}$

return Distance $= |\hat{\mathbf{p}} \cdot \mathbf{r}|$

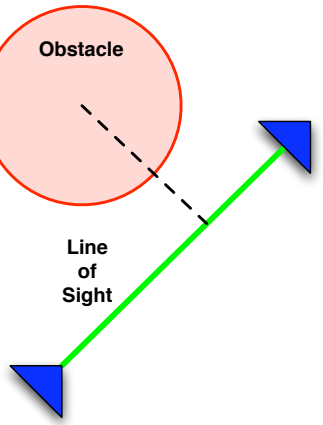


Figure 2-1: Line-of-Sight Constraint Check

Safe Distance and Maximum Range Constraints The last two constraints affect the separation between vehicles. For collision avoidance, the vehicles are required to stay at least a distance d_{safe} apart from each other, and for communications purposes, they are required to stay within the communications range $d_{\max}$ of each other. These constraints can be written as

$$d_{safe} \leq \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \leq d_{\max}, \quad \forall \ i = 1, \ldots, N \tag{2.11}$$

2.3 Initial Solution

The Gauss Pseudospectral Optimization Software requires an initial guess for all the states and controls, and, generally, the better the initial guess, the quicker the solver can find a solution. In some cases, the solver has difficulty finding a solution that is far from the initial solution. For example, if the initial solution passes on one side of

a large obstacle, the solver may not find a better solution that passes on the other side of that obstacle. To this end, an important aspect of the communications link deployment algorithm presented here is the creation of the initial guess. This initial guess should be feasible, and close to optimal. This section will present an approach that uses a Rapidly-Exploring Random Tree to create a subset of the initial guess and then fills in the rest of the initial guess using a heuristic. This approach was inspired by [2], but has several differences due to differing constraints and vehicle dynamics.

2.3.1 Rapidly-Exploring Random Trees

Previous authors [2, 3, 13, 14] have used a Rapidly-Exploring Random Tree (RRT) [28] to search for a feasible path in the full configuration space. The main advantage of using a RRT is that it is a viable solution for searching high-dimensional spaces, such as the ones that exist with multi-vehicle path planning problems. However, when applied to this problem, a normal RRT has many samples that are infeasible, and it does not exploit the problem structure.

Several authors have created modified RRT algorithms that improve performance. For example, Kuffner and LaValle created RRT-Connect [25], which grows two Rapidly-Exploring Random Trees, one from the initial configuration and one from the goal configuration. It uses a heuristic to try to connect these two trees, thereby creating a connected path from the start to the goal. While this algorithm has some useful ideas, it is not applicable to the communications link problem. In this problem, the goal state is not completely specified, because the position of the link vehicles is free.

To exploit the structure of the problem, a new modification to the standard Rapidly-Exploring Random Tree algorithm was developed and combined with other simple RRT modifications, such as biased sampling [27] to create an algorithm named RRT-Backtrack. In fact, two main properties of the communications link problem were exploited. First, since the communications equipment assumed in this problem follows line-of-sight, long straight paths are preferred over curvy paths. Second, since the vehicles form a serial chain, if each vehicle follows the vehicle ahead of it, a feasible path is formed. While this is not necessarily the optimal answer, it is a

Algorithm 2 Rapidly-Exploring Random Tree

$\mathcal{T} \leftarrow x_0$
for $k = 1$ to K **do**
  $x_{rand} \leftarrow$ RANDOM_CONFIGURATION();
  $x_{near} \leftarrow$ NEAREST_NEIGHBOR(x_{rand}, $\mathcal{T}$);
  $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
  $x_{new} \leftarrow$ NEW_STATE(x_{near}, u, Δt);
  $\mathcal{T}$.add_vertex(x_{new});
  $\mathcal{T}$.add_edge(x_{near}, x_{new}, u);
end for
return $\mathcal{T}$

feasible answer and is good enough as an initial guess for $\mathcal{GPOPS}$. Also, since the vehicle dynamics don't have a minimum speed constraint, the algorithm will be able to search for a path without regard to the full dynamics of the vehicles.

The original RRT algorithm, developed in [28] is presented in Algorithm 2. The tree $\mathcal{T}$ is initialized with the starting configuration, and then at each step the tree is grown towards (but not necessarily all the way to) a randomly chosen configuration from the tree node closest to the randomly chosen configuration. A new node is created a certain Δt away from the nearest node and then connected to the tree with a feasible edge. With the vehicle dynamics used in this thesis, the sampled input u is simply a movement towards the sampled configuration, which leads to the tree edges being straight line segments.

The main modification in RRT-Backtrack is in the $\mathcal{T}$.add_edge() step. Whereas the standard RRT connects the newly created node to the nearest node, RRT-Backtrack backtracks up the tree and connects to the eldest node that has a free line-of-sight to, and is in range of the new node. This modification creates longer, straighter connections. As a side-effect, the tree will generally have a lower maximum depth and a larger branching factor. The new ADD_ELDEST_EDGE() function is described in Algorithm 3.

Also, instead of stopping after K iterations, the loop is continued until a path to the goal is found. To promote paths towards the goal, the RANDOM_CONFIGURATION()

Algorithm 3 RRT-Backtrack Edge Addition Step

ADD_ELDEST_EDGE($\mathcal{T}$, x_{near}, x_{new}, u):
$x_{best} \leftarrow x_{near}$
$d_{best} \leftarrow \infty$ {best distance}
$g_{best} \leftarrow \infty$ {best depth}
for all x_{curr} **in** $\mathcal{T}$ **do**
  if x_{curr}.depth() == g_{best} **then**
    if LINE_OF_SIGHT(x_{new}, x_{curr}) && IN_RANGE(x_{new}, x_{curr}) && $\|x_{new} - x_{curr}\|_2 < d_{best}$ **then**
      $x_{best} \leftarrow x_{curr}$
      $d_{best} \leftarrow \|x_{new} - x_{curr}\|_2$
    end if
  else if x_{curr}.depth() $< g_{best}$ **then**
    if LINE_OF_SIGHT(x_{new}, x_{curr}) && IN_RANGE(x_{new}, x_{curr}) && $\|x_{new} - x_{curr}\|_2 < d_{best}$ **then**
      $x_{best} \leftarrow x_{curr}$
      $d_{best} \leftarrow \|x_{new} - x_{curr}\|_2$
      $g_{best} \leftarrow x_{curr}$.depth()
    end if
  end if
end for
$\mathcal{T}$.add_edge(x_{best}, x_{new});
return $\mathcal{T}$

step is biased by making a certain percentage of the samples deterministically the goal location (x_g, y_g).

2.3.2 Full Initial Guess

The RRT-Backtrack algorithm above creates a path for the lead vehicle, but it does not directly create the path for the link vehicles, and it doesn't create the velocity or control profiles either. However, as mentioned previously, for the initial solution the link vehicles will follow the same path as the lead vehicle. Because the edges in the chosen path are lines-of-sight, a feasible final configuration is to place one vehicle at each node in the path The lead vehicle will travel the full path all the way to the goal, vehicle 2 will travel the same path but stop at the parent node of the goal node,

and so on.

To fill in the velocity and control trajectories, another optimal control problem is set up. A controller is used in conjunction with the vehicle dynamics to determine the required controls as well as the resulting velocities. The basic cost function for this optimal control problem is

$$\int_{t_0}^{t_f} 1dt. \tag{2.12}$$

Since the vehicles are constrained to move along a line, the dynamics can be reduced from having four states to having two states, namely position and velocity, and the control vector can correspondingly be reduced to one state. The modified dynamics are

$$\dot{x} = V \tag{2.13}$$

$$\dot{V} = -c_d V^2 + F \tag{2.14}$$

$$F \leq F_{\max}. \tag{2.15}$$

Additionally, to ensure feasibility, the vehicles are constrained to come to a stop at each node in the path. This allows each segment in the path to be simulated separately. Also, remember that this approach is only being used to find an initial guess, and not the final optimal solution.

This optimal control problem is a constrained optimal control problem where the optimal solution is a bang-bang controller. At the beginning of the trajectory, full control is applied towards the end of the segment currently being traversed, and then at the last moment full reverse control is applied to bring the vehicle to a stop at the end of the segment. The acceleration and velocity profiles can be solved for using the dynamics and these known control inputs. The last unknown that must be determined is the switching time, which is when the control must switch from full forward control to full reverse control. To do this, the augmented Hamiltonian is formed:

$$H_a = 1 + p_1 V + p_2(-c_d V^2 + F). \tag{2.16}$$

Since the objective is to minimize H, the following control law can be established:

$$F(t) = \begin{cases} F_{\max} & \text{if } p_2(t) < 0 \\ -F_{\max} & \text{if } p_2(t) > 0 \end{cases}, \tag{2.17}$$

and it can be seen that the switch will occur when $p_2(t)$ is zero[1]. Next, using the condition that $\dot{\mathbf{p}} = -{H_x}^T$, the equations for the co-states are found:

$$\dot{p_1} = 0 \tag{2.18}$$

$$\dot{p_2} = p_1 - 2c_d p_2 V. \tag{2.19}$$

Then, using the transversality condition $H(t_f) + h_t(t_f) = 0$, and the boundary conditions $x(t_f) = d$ and $V(t_f) = 0$, where d is the length of the segment, the following condition can be stated:

$$1 + p_2(t_f)F(t_f) = 0. \tag{2.20}$$

Using the above equations along with the boundary conditions $x(t_0) = 0$ and $V(t_0) = 0$, the switching time and the final velocity profiles can be solved for. This full initial guess is then discretized and passed into $\mathcal{GPOPS}$ as the initial guess to the optimal (and feasible) solution.

2.4 Results

This section will discuss results from the Rapidly-Exploring Random Tree (RRT) algorithm and from the $\mathcal{GPOPS}$ package applied to two different scenarios. In both scenarios shown, the base station is located near the southwest corner of the map at coordinate $(2, 2)$, and the goal is located near the northeast corner at coordinate $(45, 45)$. The grey circles represent obstacles. A two-dimensional problem is shown for ease of illustration, but the same approach is applicable in three dimensions. The first scenario has an extra obstacle on the west side of the map that prevents any feasible solutions using just one link UAV. The second scenario removes this obstacle

[1]No singular arcs exist in this problem

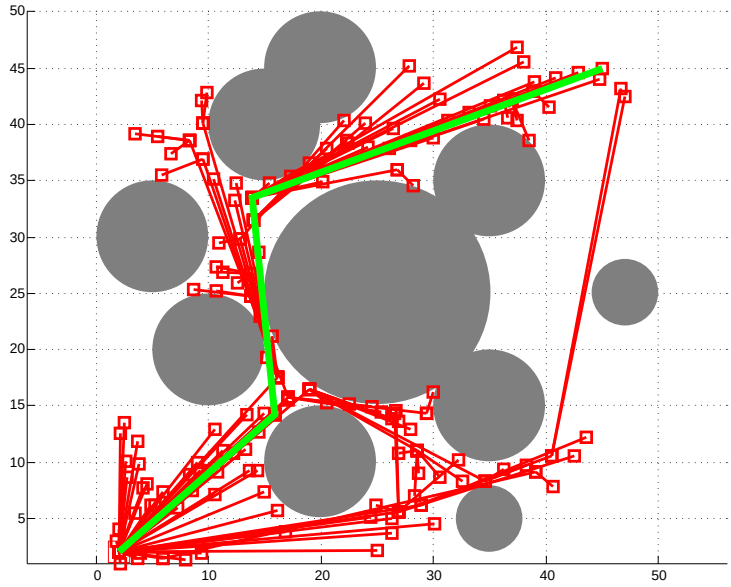


Figure 2-2: RRT-Backtrack Solution, Scenario 1

and demonstrates another interesting property of the solution, namely the elimination of unneeded vehicles.

2.4.1 First Scenario

This scenario presents the vehicles with several different options for traversing the obstacle field. All options require three vehicles, but the resulting path length, mission time, and energy expenditure varies by chosen path.

Rapidly-Exploring Random Tree Search

Figure 2-2 shows a typical initial solution that is obtained from the RRT algorithm. The red boxes and lines represent the RRT's nodes and edges, respectively, while the green line shows the chosen path from the base to the goal. It can be seen that the RRT explores several different gaps between or around obstacles. While there is a feasible path that passes through the gaps on the southern part of the map, this path is more circuitous than the path along the northern part. The path that the search finds has the minimum number of edges (3) and is a short path to the goal. Due to the design of RRT-Backtrack, the paths found by this algorithm tend to be

the shortest paths with the fewest number of edges, which is the desired property for setting up a communications chain. The figure also shows how many edges tend to originate from a single node and bloom out from it. This is due to the connection heuristic that connects node to the eldest possible node.

Optimization

Once the RRT finds a path to the goal, the full initial solution is interpolated. The final locations of the vehicles in the initial solution are at each of the corners in the RRT path, and the velocity profiles are created as discussed previously. Once the full initial solution is created, it is fed into $\mathcal{GPOPS}$ and the software solves the optimal control problem.

The paths of the vehicles is shown in Figure 2-3 and the corresponding velocity profiles of the vehicles are shown in Figure 2-4. Although the vehicles fly through the same gaps as in the initial solution, the final solution looks significantly different than the RRT solution. First, the paths are smoother in the final solution. The lead vehicle follows the contour of the obstacles because this is the shortest path to the goal, and, because of the dynamics in the vehicles, the vehicles can maintain their highest speed by flying smooth paths rather than by slowing down to turn sharp corners.

Second, the part of the cost function that tries to minimize the system's energy usage is reflected in two different ways. The link vehicles travel the shortest distance possible to meet the communications constraints at their final position, and they fly at velocities below the maximum and even reduce to a very low velocity towards the end of their trajectories. The lead vehicle, on the other hand, flies at maximum speed for the entire mission. While this uses a lot of energy, the lead vehicle is the only vehicle that can directly control the duration of the mission, which is the other element of the cost function. This behavior shows that, when the cost function is properly weighted, the optimization can properly balance between two conflicting cost function elements.

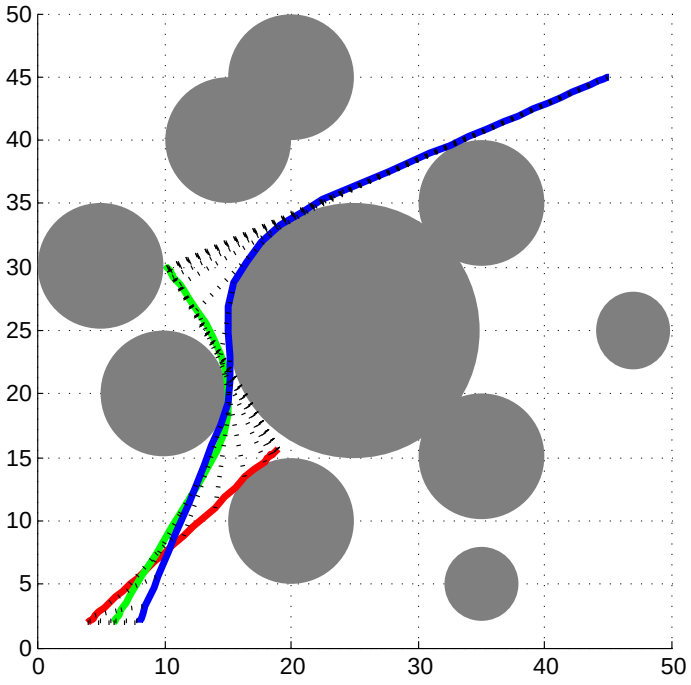


Figure 2-3: $\mathcal{GPOPS}$ Solution, Scenario 1

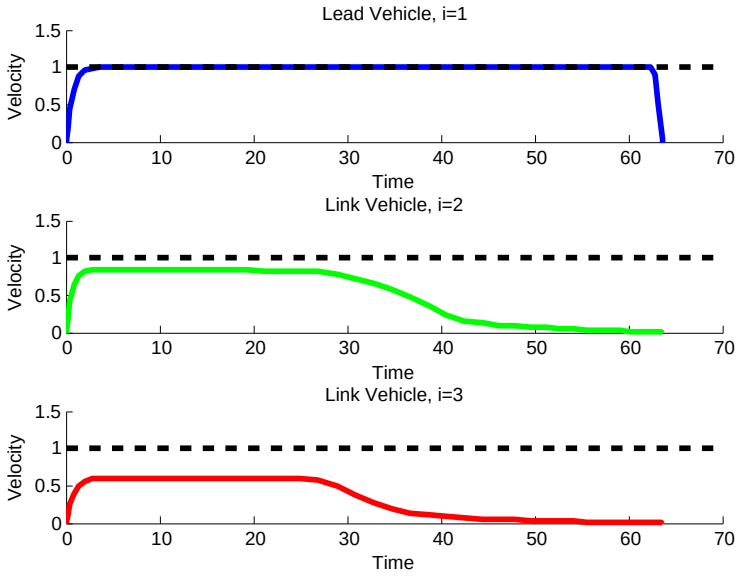


Figure 2-4: Velocity Profiles of $\mathcal{GPOPS}$ Solution, Scenario 1

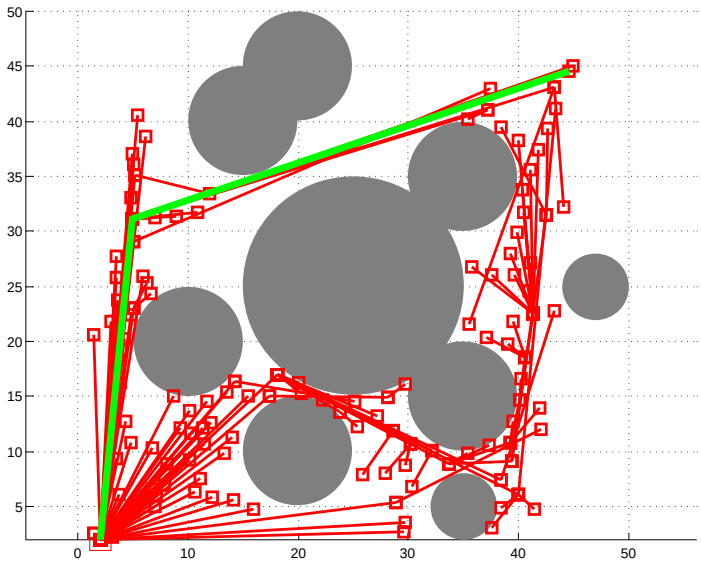


Figure 2-5: RRT-Backtrack Solution, Scenario 2

2.4.2 Second Scenario

The second scenario removes the left-most obstacle on the map, thereby enabling the vehicles to complete the mission with just one link vehicle instead of two, albeit in more time.

Rapidly-Exploring Random Tree Search

In this scenario, the search finds and selects the new path that only requires two vehicles. This path has two long straight legs, which are optimal for communications purposes. It also finds more paths along the bottom of the map and no path through the middle gap. This shows the variability, but also the breadth of the RRT search. If more paths are desired, then the RRT can be run for a longer period of time until more nodes and paths are created.

Optimization

One goal of this mission is to accomplish it with the minimum number of vehicles needed. The RRT has clearly demonstrated that in this second scenario, a solution with only two vehicles is feasible. However, in some cases the RRT might find a

path that has more than the minimum number of required edges. In this case, it is desirable for the optimization to realize that fewer vehicles are needed, and then eliminate the extra ones from the solution.

To demonstrate this behavior, the initial solution given to $\mathcal{GPOPS}$ in scenario 2 used the path shown in Figure 2-5 but with three vehicles rather than just the required two. The resulting vehicle paths are shown in Figure 2-6 and the velocity profiles are shown in Figure 2-7. The optimization recognizes that the red link vehicle is unneeded and does not move it from its initial position near the base. As a result, a post-processing step can remove from the solution any vehicle that does not move.

The lead vehicle still flies along the shortest path through the chosen gap while the green link vehicle swings wide to the left to increase its field-of-view towards the lead vehicle without having to move fast to keep up with it.

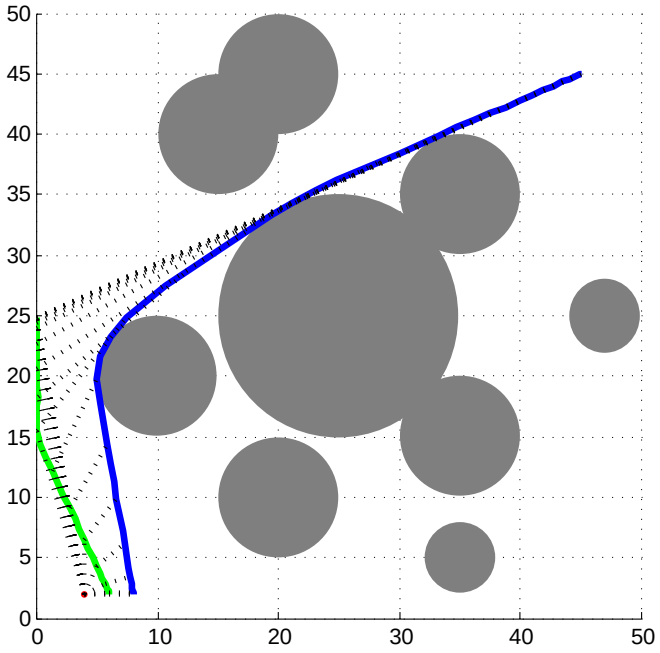


Figure 2-6: $\mathcal{GPOPS}$ Solution, Scenario 2

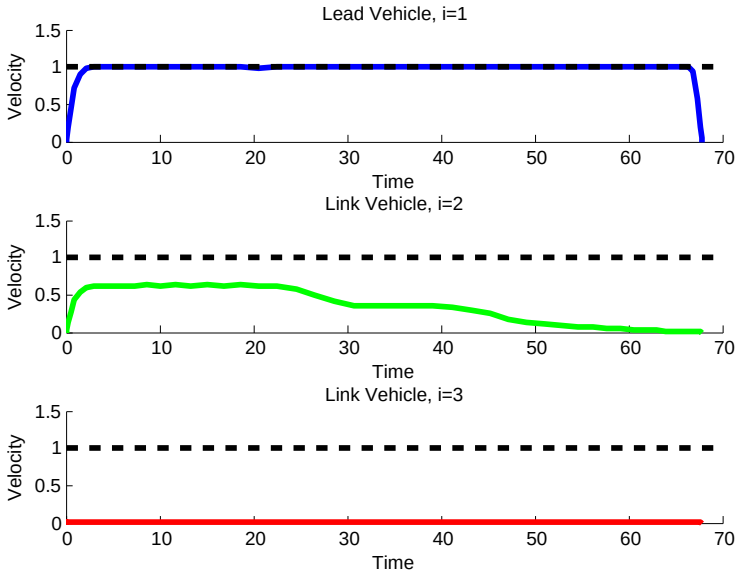


Figure 2-7: Velocity Profiles of $\mathcal{GPOPS}$ Solution, Scenario 2

THIS PAGE INTENTIONALLY LEFT (ALMOST) BLANK

Chapter 3

Real-Time Reconfiguration

This chapter addresses the problem of communications chain path planning when the target of the lead vehicle changes often and the path needs to be replanned often. Consider a UAV following a target vehicle along a mountainous road. Knowing that the UAV will eventually fly out of line-of-sight, the operator deploys additional linking UAVs behind the lead UAV. Using their knowledge of the local terrain and the signal strength of the communications links between the UAVs, they coordinate their planned paths to maintain a communications chain between the lead UAV and the operator.

While the previous chapter addressed a similar problem, the approach presented there was not flexible to replanning, and it could not easily handle a changing target location. The algorithm presented here attempts to solve these problems by trading overall optimality for flexibility and real-time implementability. These two factors may, in many circumstances, actually be more important than optimality.

Using a local obstacle map, each vehicle plans its path over a short horizon using an optimization. The lead vehicle attempts to minimize its distance to the target that it is trying to follow, while the cost function for the linking vehicles promotes a vehicle configuration that is favorable for communication now and in the future. The constraints for all vehicles ensure that the vehicles remain in a feasible (connected) configuration.

The path planning for the vehicles is both decentralized, with each vehicle plan-

ning its own path, and coordinated, with the sharing of certain information between planners. Specifically, the optimization is performed sequentially; the solution from one planner is passed to the next planner in the chain, which takes this new information into account to create its own plan. This chapter will more specifically lay out the problem and then discuss each of the various parts of the planner, including the representation of the environment map, the cost functions and constraints, and information sharing between vehicles. Simulation and flight test results of the algorithm present here are shown in Chapter 4.

3.1 Problem Statement

Supposed a team of unmanned vehicles is in a general configuration where the lead vehicle is performing a task that requires unbroken communications back to a base station along range-limited, line-of-sight communications links. This communications chain between the lead vehicle and the base station is provided by link vehicles, each of which continuously adjusts its position to maintain line-of-sight to the vehicle ahead and the vehicle behind (in the chain). As the lead vehicle moves to accomplish its mission, the link vehicles adjust their position to maintain the required communications link while hindering the lead vehicle's performance as little as possible. The vehicles are assumed to be holonomic point masses with maximum velocities, and the ordering of the vehicles in the chain is fixed.

3.1.1 Notation

The state of each vehicle i is denoted by the vector $\mathbf{x_i} = [x\ y\ z]^T$, where x, y, and z are the east-west, north-south, and up-down positions of each vehicle in the environment, respectively. The vector $\mathbf{xh_i} = [x\ y]^T$ is used for the horizontal component of the position of the vehicle. The lead vehicle has index $i = 1$, the link vehicle just behind the lead vehicle has index $i = 2$, and so on up to $i = N$. The base station is considered to be fixed at a known location $\mathbf{x_{N+1}}$.

Each environment, in general, has obstacles that can not be traversed by the ve-

hicles and that block the communications links between vehicles. The set of obstacles is denoted by $\mathcal{O}$ and the environment is represented by the binary map function

$$\mathbf{M_b}(x, y, z) = \begin{cases} 1, & \text{if } [x\ y\ z]^T \in \mathcal{O}, \\ 0, & \text{o.w.} \end{cases} \tag{3.1}$$

3.2 Optimization

The overall system of vehicles is attempting to minimize the distance between the lead vehicle and the given target location, but to accomplish this, each vehicle solves a local optimization over a short planning horizon. Encoded in the cost function of the optimization is a heuristic that improves the performance of the vehicle without explicitly planning past the planning horizon. This approach is taken because quantifying the effect of each link vehicle's movement on the lead vehicle's cost is difficult, especially in a distributed manner, but quantifying the effect on neighboring vehicles is much more manageable. This thesis hypothesizes that if each link vehicle is always made to provide a good, robust communications link to the next vehicle, then the overall communications chain will achieve a good configuration that achieves the desired goal.

The communications model assumes a spherical/disk model where the links between vehicles are limited by both range and line-of-sight, and the signal is at a nominal strength within these constraints and zero outside of the constraints. In other words, for there to be a direct connection between two vehicles, the line between them must be free of obstacles and the two vehicles must be within a specified range of each other. These conditions can be written as

$$\|\mathbf{x_{i+1}} - \mathbf{x_i}\|_2 \leq R_{\max} \quad \forall \quad i = 1, \ldots, N, \tag{3.2}$$

where $R_{\max}$ is the maximum range of the communications equipment, and

$$(1 - \lambda) \cdot \mathbf{x_{i+1}} + \lambda \cdot \mathbf{x_i} \notin \mathcal{O} \quad \forall \quad \lambda \in [0, 1], \quad i = 1, \ldots, N. \tag{3.3}$$

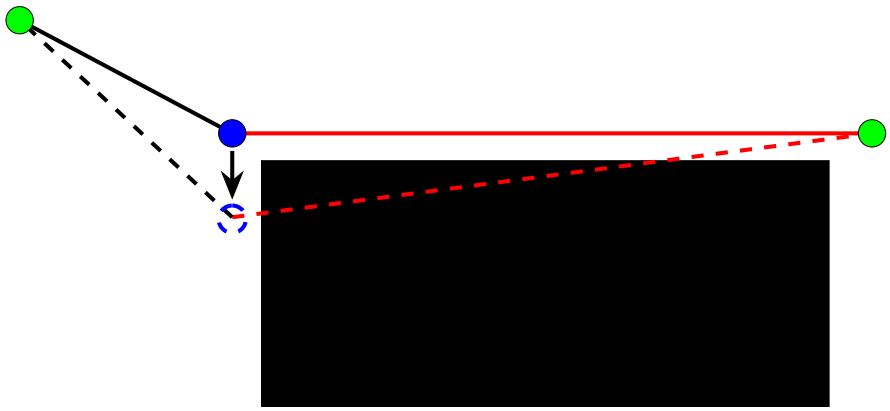


Figure 3-1: Robustness to Movement of Vehicle Ahead (blue) – Black Link is Robust, Red Link is not Robust

For the vehicles to provide a communications link at the current time, it is only important to keep the line-of-sight free of obstacles at the current time. However, this may not be robust to future movements of adjacent vehicles. Therefore, the optimization performed by each planner needs to take into account where the vehicle ahead or behind might move. For example, if the line-of-sight between two vehicles must pass through a gap between two obstacles, then it is more robust for that line-of-sight to pass through the middle of the gap rather than along one side of it. The former placement allows either vehicle to move its position perpendicularly to the line-of-sight without immediately risking breaking the communications connection. Figure 3-1 illustrates this with a lead vehicle (in blue) near the corner of an obstacle. While both link vehicle positions (in green) provide an adequate communications link at the current time, as soon as the lead vehicle moves south, a link vehicle placed on the right loses line-of-sight to the lead vehicle, but the left link vehicle position is robust to this movement. The vehicle's movement could be due either to disturbances or to planned movement.

3.2.1 Cost Function

The cost function of the optimization acts as a heuristic by making each vehicle's communications link to the next vehicle in the chain as flexible as possible to future movement of that vehicle. Based on the above observation about the line-of-sight,

along with conditions 3.2 and 3.3, the cost functions used are

$$\min J_1 = \|\mathbf{x_t} - \mathbf{x_1}\|_2 \tag{3.4}$$

$$\begin{aligned} \min J_i = {} & \|\mathbf{x_i} - \mathbf{x_{i-1}}\|_2 \cdot \int_0^1 \mathbf{M}((1-\lambda)\cdot\mathbf{x_i} + \lambda\cdot\mathbf{x_{i-1}})\, d\lambda \\ & + \alpha \cdot \|\mathbf{xh_i} - \mathbf{xh_{i-1}}\|_2 \\ & + \beta \cdot d\mathbf{x_i}, \qquad \text{s.t. } \alpha, \beta \geq 0, \quad \forall\ i \in 2, \ldots, N \end{aligned} \tag{3.5}$$

where $\mathbf{x_t}$ is the target location for the lead vehicle, $d\mathbf{x_i}$ is the speed of the vehicle, and $\mathbf{M}$ is a modified map function that will be discussed in detail in Section 3.3.

The line integral term in Eq. 3.5 is a measure of how close the line-of-sight between vehicle i and $i-1$ is to obstacles. The closer the line-of-sight passes near obstacles, the higher this term, and the less robust the communications link is. In other words, this term in the optimization tries to maximize the distance between the line-of-sight and obstacles.

The second term tries to move each vehicle as close as possible to the vehicle ahead of it. In general, the closer two vehicles are, the more robust their link is because it is less likely that a movement on the part of either vehicle will move the pair into a configuration where the line-of-sight is blocked by an obstacle. The term α (typically around 0.5) can be used to tune the relative importance of these two terms. The line integral is also minimized when the vehicles are close, but this second term only operates on the horizontal distance between two vehicles. It has been determined through simulation that if the full 3-dimensional vector is used in the second term, then the vehicles rarely choose a path that increases altitude if the vehicle ahead doesn't also choose such a path, as this would increase the distance between the two vehicles. However, in many situations it is beneficial to fly at an altitude that is higher than the vehicle ahead, and so using only the horizontal position vector for the second term doesn't penalize this movement. Furthermore, α is generally chosen such that the first term dominates the second term when the line-of-sight passes near obstacles.

The third term penalizes motion, which is used to reduce thrashing in the solution. In some cases, there may be many locally optimal or nearly optimal solutions for a vehicle rather than one unique solution. Without this damping term, vehicles may cycle between several positions, which is not desired. With this damping term, a vehicle will only move to a new position if there is a non-trivial decrease in cost. The factor β is chosen to be small (0.05-0.1) so as not to dominate the cost function.

The cost function for each vehicle only considers the link to the vehicle ahead, and does not consider the vehicle behind. Having only one vehicle evaluate the cost along each link simplifies the optimization because there are half as many expensive cost function evaluations, and because the goal of the optimization is to provide a good communications service to the vehicle ahead, not the vehicle behind. The link to the vehicle behind will, however, be considered in the optimization's constraint set.

3.2.2 Constraints

The optimization is also subject to certain constraints with the main goal of keeping the system in a feasible, connected configuration. The first two constraints maintain a clear line-of-sight to the vehicles ahead and behind. These constraints can be written as in Eq. 3.3. The next two constraints, written as in Eq. 3.2, make sure that adjacent vehicles are within range of each other. To avoid collisions between vehicles, additional minimum separation constraints are added. Lastly, the vehicles are required to plan paths that are dynamically feasible.

3.3 Environment Map

As explained in the previous section, one term in the cost function of the linking vehicles (Eq. 3.5) is a line integral from one vehicle to the vehicle in front of it that integrates the value of the map along that line. If only the binary map $\mathbf{M_b}$ were used for this, the line integral would be zero for all feasible links and non-zero for infeasible communications links. This does not sufficiently reflect the desired properties of the cost function because it does not reflect closeness between the line-of-sight and any

obstacle it passes.

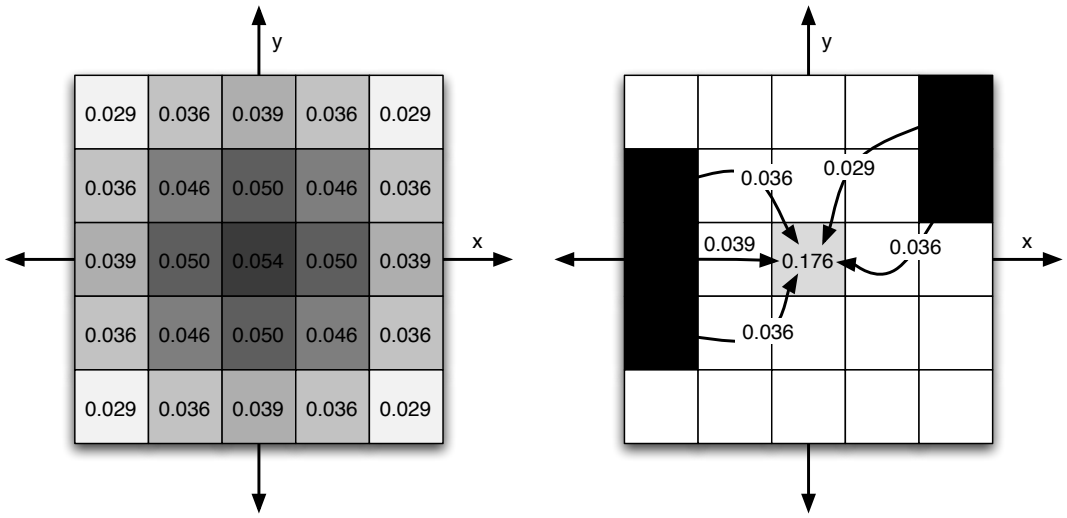
To achieve the desired response from the line integral term, each point on the map needs to contain information about the points surrounding it. The rationale for this is that if either vehicle in the link moves, then the link itself could move into a neighboring part of the map. Thus, if the link is made to pass through parts of the map that have no obstacles nearby, then it will be robust to movement of the link. Furthermore, it would be best to incorporate this "neighborhood" information in general, rather than specifically for each link. This allows the computation of neighborhood information to be performed once, rather than each time the cost function is computed, thus allowing for lower online computational complexity.

One method for incorporating this neighborhood information is with a convolution, specifically a three-dimensional convolution for a three-dimensional map. The two input parameters in the convolution are the original binary map and a *convolution kernel* K. The kernel is a function with the same dimension as the map. The convolution operation moves the kernel to each point in the map and then maps the integral of the product of the map and the kernel to that point. This can be written as

$$\mathbf{M}'(x,y,z) = \mathbf{M_b} * K \equiv \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbf{M_b}(\tau,\upsilon,\phi)K(x-\tau,y-\upsilon,z-\phi)d\tau d\upsilon d\phi. \quad (3.6)$$

Performing this convolution operation creates a "blurred" version of the original binary map, and the line integral in the map essentially becomes a penalty function where passing close to an obstacle is penalized.

The two-dimensional convolution operation on a discretized map is illustrated in Figure 3-2. Figure 3-2(a) shows a bounded kernel. A true Gaussian function has a domain of $(-\infty,\infty)$ for all the arguments, but the kernel shown here is truncated to a finite domain. Figure 3-2(b) shows what happens at one point in the convolution. The grid represents the binary environment, where the black squares have a value of 1. The sum of the product of the kernel and the underlying section of the map is mapped to the central cell.



(a) Bounded Gaussian Kernel (b) Convolution on One Cell

Figure 3-2: Convolution Operation

For this algorithm, a trivariate Gaussian kernel $K \sim N(0, \mathbf{\Sigma})$ is chosen to put more emphasis on obstacles close to a given point on the map and less emphasis on distant obstacles, which are less likely to have an effect on a communications link passing through a point. Also, by using a Gaussian kernel, the new map values remain scaled between zero and one[1].

The covariance matrix $\mathbf{\Sigma}$ is a tuning parameter and is chosen to be a diagonal matrix with $\sigma_{11} = \sigma_{22}$ and $\sigma_{33} < \sigma_{11}, \sigma_{22}$. The weight along the vertical dimension (σ_{33}) is chosen to be less than the two weights along the horizontal dimensions because a typical environment is usually smaller along the vertical dimension than along the other two dimensions.

The last modification that must be made to the new map $\mathbf{M}$ is that all parts of the map that original had a value of 1 in $\mathbf{M_b}$ should maintain that same value, rather than use the convoluted value. This operation is written as

$$\mathbf{M}(x, y, z) = \begin{cases} 1, & \text{if } \mathbf{M_b}(x, y, z) = 1 \\ \mathbf{M}'(x, y, z), & \text{if } \mathbf{M_b}(x, y, z) = 0. \end{cases} \tag{3.7}$$

[1]The hyper-volume under a trivariate Gaussian is 1. For two-dimensional environments, a bivariate Gaussian is used, and the volume under it is also 1.

This map is the one that will be used by the optimization algorithm.

3.4 Considerations for Urban Environments

Urban environments with tall buildings require some special consideration with respect to the cost function and the environment map convolution. If the communications chain is passing in between buildings at a low altitude, it can easily become wrapped around a building or become unnecessarily circuitous. One possible approach for the team of vehicles is to gain altitude and pass individual chain links up and over buildings and then straighten and shorten the chain. However, doing so often requires a momentary increase in the line integral term of the cost function before achieving a decrease, and if the planning distance is not great enough, then this solution may not be found.

This problem is solved by two modifications that are employed in parallel. First the convolution is modified to give the buildings sloped sides. Rather than performing one three-dimensional convolution, a separate two-dimensional convolution is performed at each altitude with a decreasing kernel size as altitude is increased. As a result, the buildings will have the largest border at the bottom and a small one at the top. This allows the line-of-sight integral to have a lower value at higher altitudes, which offsets the factors that increase the cost with altitude.

Secondly, another term is added to the link vehicle cost function to encourage altitude gain when needed. This produces the modified link vehicle cost function

$$\tilde{J}_i = J_i + \gamma \cdot (z_{\max} - z_i), \quad \text{s.t. } 0 \leq \gamma \leq \gamma_{\max}, \tag{3.8}$$

where $z_{\max}$ is the maximum altitude that the vehicle can fly to and $\gamma_{\max}$ is a tuning factor. The term γ is a measure of how circuitous the communications chain is, and is calculated as follows:

$$\gamma = \min\left(\gamma_{\max}, \frac{d_{chain} - \|\mathbf{xh_1} - \mathbf{xh_{N+1}}\|_2}{\|\mathbf{xh_1} - \mathbf{xh_{N+1}}\|_2}\right). \tag{3.9}$$

Here d_{chain} is the length of the chain and $\|\mathbf{xh_1} - \mathbf{xh_{N+1}}\|_2$ is the horizontal, straight-line distance from the lead vehicle to the base. Both of these values can be computed by the system from the information that is shared between vehicles. When the length of the chain is much greater than the distance of the lead vehicle from the base, it is indicative of the chain being "stuck" on a building, and so the γ term dynamically increases and creates an incentive for the link vehicles to gain altitude. When the chain becomes unstuck, the gamma term returns to a lower value and the gradient is reduced. $\gamma_{\max}$ is typically chosen to be 0.4 in the scenario shown in Chapter 4.

3.5 Algorithm Architecture

The various elements discussed above must be assembled together into one planning and optimization unit. Each vehicle in the team executes its own independent planner and share required information over a communications link that is established to the vehicle ahead in the chain and to the vehicle behind. This algorithm is shown in its two variations in Algorithms 4 and 5.

Each vehicle is initialized with the environment map; it is assumed that the environment map is known *a priori*. Once the vehicle is initialized, each vehicle creates a communications link to the vehicle ahead of it and behind it over which it will transmit information for coordination and the information from the lead vehicle's sensor. Also, a separate thread is spawned to run the vehicle's low level controller, which receives the plan through a block of shared memory.

The optimization is performed periodically and concurrently with a vehicle controller that implements the plan calculated by the optimization. The planning is done in a receding horizon fashion, which is a well-established method for performing planning and control when long-term planning is computationally difficult or when long-term information is unavailable or unreliable [4]. Planning only for a short period of time and then executing over an equally short (or even shorter) period of time allows the planner to compensate well for new information introduced into the system. In the case of the communications reconfiguration problem, this might mean

Algorithm 4 Reconfiguration Planner (Link Vehicle)

```
Initialize(M)
Socket_A ← Accept_Connection() {link from vehicle ahead}
Socket_B ← Connect() {link to vehicle behind, except last vehicle}
Current_Plan ← NULL
Spawn_Controller_Thread(Ptr_to_Current_Plan)
loop
  Recv_Plan_Req(Socket_A) {Blocking Call}
  Send_Plan(Socket_A)
  Plan_A ← Recv_Plan(Socket_A) {Blocking Call}
  Send_Plan_Req(Socket_B)
  Plan_B ← Recv_Plan(Socket_B) {Except last vehicle}
  Current_Plan ← Optimize(Plan_A, Plan_B)
  Send_Plan(Current_Plan, Socket_B) {Except last vehicle}
end loop
```

Algorithm 5 Reconfiguration Planner (Lead Vehicle)

```
Initialize(M)
Socket_B ← Connect() {link to vehicle behind}
Current_Plan ← NULL
Spawn_Controller_Thread(Ptr_to_Current_Plan)
loop
  Wait(Opt_Timer) {block until optimization timer expires}
  Send_Plan_Req(Socket_B)
  Plan_B ← Recv_Plan(Socket_B)
  Current_Plan ← Optimize(Target, Plan_B)
  Send_Plan(Current_Plan, Socket_B)
end loop
```

a new target location for the lead vehicle to follow. It would not make much sense for vehicles to compute a long-term plan, which is computationally difficult, if a completely new plan is likely to become necessary a short time later. One disadvantage of using a receding horizon approach is that the planner may be blind to something that will strongly affect the plan in the future past the planning horizon. However, this negative effect can be mitigated by using an appropriate heuristic, which is a primary focus of this algorithm and this thesis.

By running the optimization and the controller concurrently, the vehicle can keep moving along the current plan while the optimization creates a new plan. To ensure that the controller never has to wait for a new plan, the planning horizon, which is the length of the plan that is created, is set to be longer than the time between successive planning cycles. Of course, this inter-planning time must itself be longer than the execution time of the optimization. The exact times for these periods varies with the scenario; various situations will be discussed in Chapter 4.

As mentioned previously, this planning algorithm is a coordinated algorithm, which is to say, it incorporates coordination between adjacent vehicles in the communications chain. Higher levels of coordination or cooperation can lead to better solutions, but generally at the cost of computational difficulty. A range of coordination/cooperation levels were considered, from simply sharing each vehicle's current position to sharing the complete local cost function with adjacent vehicles. The lowest level of sharing was tested, but it was determined that more sharing could lead to better results with very little increased demand on inter-vehicle communication and onboard computation.

Sharing the full local cost function was also considered. In this approach, each vehicle computes its local cost function conditioned on the cost function that it receives from the vehicle ahead[2]. The last vehicle then chooses its best solution and passes this choice forward in the chain. Each successive vehicle then chooses a final plan conditioned on the plan received from the vehicle behind. In this way each vehicle's answer becomes conditioned on the choices made by each adjacent vehicle.

[2]The lead vehicle computes an unconditional cost function

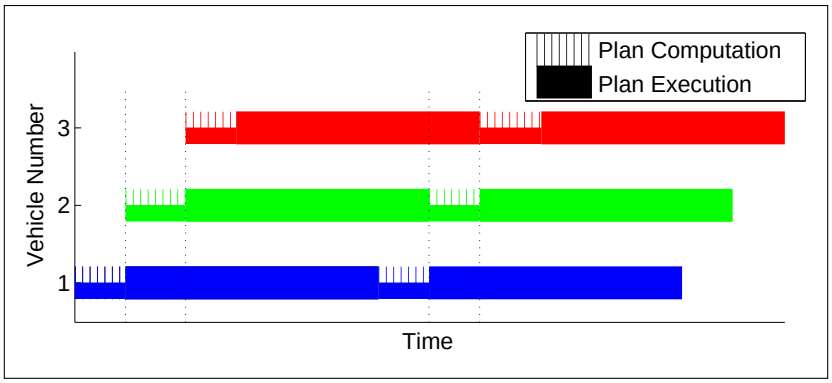


Figure 3-3: Sequential Optimization and Planning Timeline

While this approached shows promise, after some testing it was determined that this approach is computationally intractable. Whereas an unconditional cost function generally has three dimensions (x, y, and z), the conditional cost function has to take into account all possible solutions of the vehicle ahead, thereby making the function a six-dimensional function.

The chosen solution was to perform the optimization in a framework similar to Gauss-Seidel optimization [26]. In this approach, each vehicle, starting with the lead vehicle, performs its own optimization and then passes its best answer to the vehicle behind it in the chain. This triggers the next vehicle to begin its optimization, such that each vehicle sequentially performs its optimization based on the result of the vehicle ahead of it. The advantage of this approach is that each optimization is based on the future plan of the vehicle ahead, rather than its current position. This avoids the pitfalls of the most basic approach, which uses each vehicle's current state. However, there is little to no increased communications requirement. The previous plan of the vehicle behind is used because no better information is available. A timeline of this approach is shown in Figure 3-3.

3.5.1 Planner

The planner computes a path for the vehicle over a short distance that can be traveled within the planning horizon time. The plan terminates with the vehicle in an invariant state (e.g. it is not moving). The plan is feasible with respect to the obsta-

cle constraints and the line-of-sight constraints. The planner checks these constraints using the environment map and the plans that it receives from the adjacent vehicles. Also, the planner uses the optimization described in Section 3.2 to minimize the cost at the invariant state. The cost function, described in Section 3.2.1, serves as the heuristic for the receding horizon planner. As described, the cost function gives the vehicle ahead in the chain the largest range of feasible motion.

3.6 Properties of the Line Integral Term

The line integral term, when calculated on the convoluted map, exhibits some nice properties for the communications linking problem. The main goal of the linking vehicles is to maintain a large *symmetric* field-of-view towards the next vehicle ahead. This allows the vehicle ahead the greatest freedom of movement. If the vehicle ahead is the lead vehicle, then it has the greatest freedom to move towards the goal, and if the vehicle ahead is a link vehicle, then it has the greatest freedom to move to a position beneficial to the vehicle ahead of it. In this way, if a link vehicle moves to a better position then all the vehicles ahead may benefit with an increased range of feasible motions.

3.6.1 Symmetric Field-of-View

The field-of-view (FOV) from vehicle i to vehicle $i-1$ is defined as the solid angle of the largest obstacle-free spherical cone[3] with radius equal to the distance between vehicles i and $i-1$ projected from vehicle i towards vehicle $i-1$. By definition, and because of the convexity of the cone, all points in this cone are within line-of-sight of both vehicles. The *symmetric* field-of-view (SFOV), drawn in Figure 3-4, is defined as the solid angle of the largest obstacle-free spherical cone whose axis of revolution extends from vehicle i to vehicle $i-1$. In other words, the symmetric spherical cone

[3]The algorithm is applicable to both three-dimensional and two-dimensional situations. This thesis typically describes the algorithm in terms of three dimensions, but many of the examples and figures are two-dimensional.

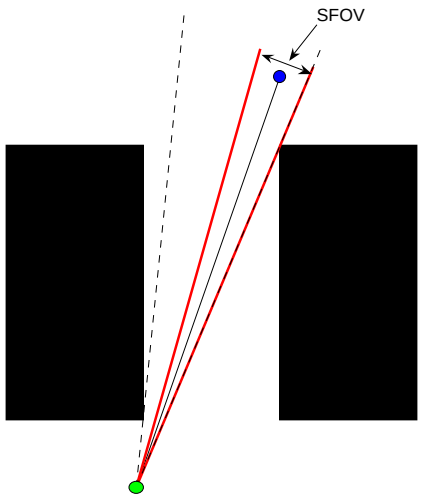


Figure 3-4: Symmetric Field-of-View (Overhead View)

is symmetric about the line segment between the two vehicles. The solid angle of the symmetric FOV will always be less than or equal to that of the general FOV.

A main assumption being made in the path planning algorithm is that the movement of all the vehicles is only known for a small increment of time. Thus, the planner must account for the possibility of the vehicle ahead moving in any possible direction beyond this small increment of time. This is why a symmetric field-of-view is desirable. If the normal field-of-view would be used, the optimization might find a solution where there is a very large field-of-view with one vehicle at the very edge of the base of the cone. Unless it is known that the vehicle ahead will move towards the center of this cone, this solution is not robust since the vehicle could move in the other direction and quickly lose line-of-sight with the other vehicle. A symmetric field-of-view resolves this issue by maximizing the minimum field-of-view in any direction. This issue is illustrated in Figure 3-1.

3.6.2 Field-of-View Through a Gap

Consider two obstacles that are close together but with a gap between them that is large enough for a vehicle to pass through. If there is a vehicle on each side of the gap, then the obstacles forming the gap will form the feature that most restricts the

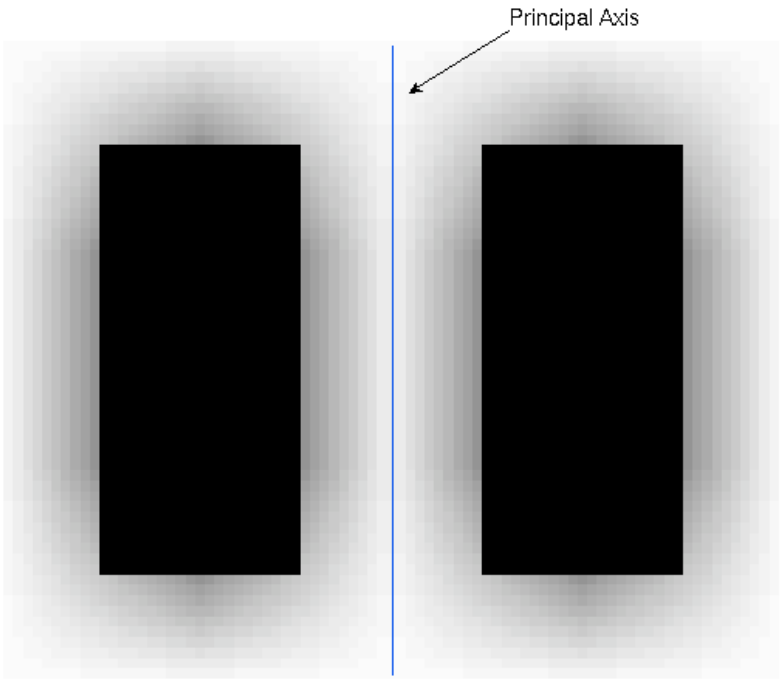


Figure 3-5: Obstacle Gap (Overhead View)

field-of-view (FOV) from one vehicle to the other. Such an obstacle feature might be two tall and closely-spaced buildings in a city, or two hills with a valley between them in mountainous terrain.

Principal Axis In examining the gap shown in Figure 3-5, it can be seen that the convolution naturally produces a straight "valley" between the two obstacles where the value of the map is at a local minimum. This valley will be called the *principal axis* of the gap because it characterizes the direction of the gap[4].

Field-of-View Optimization Now, consider the optimization problem that maximizes the SFOV subject to the constraint that one of the vehicles is fixed and that the distance between the two vehicles is also fixed as in Figure 3-5. The FOV cone is projected from the movable vehicle. The optimal solution to this problem is the one where the line-of-sight passes directly through the middle of the gap. With this orientation, the gap has the largest apparent width as seen from the optimizing vehicle.

[4]In three dimensions, this may generalize to a plane for certain obstacle geometries.

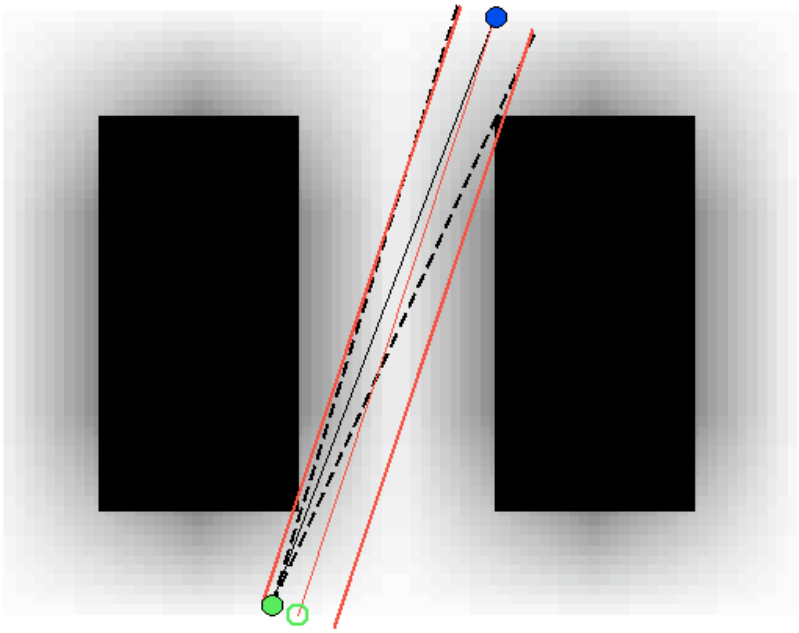


Figure 3-6: Cone Field-of-View vs. Column Field-of-View

This example is examined in more mathematical detail in Appendix A.

The previous example was a relatively simple example, and under different assumptions the same explanation for the optimal solution may not be possible. Consider the setup in Figure 3-6. In this case, the two obstacle features that impinge on the field-of-view cone are the two corners on either side of the gap. Because these corners are not equidistant from each of the vehicles, the optimal solution offsets the line-of-sight slightly towards the corner that is closer to the movable vehicle. However, this solution is dependent on the exact vehicle configuration. Calculating this configuration-specific optimal solution is not as simple as performing a line integral on a convoluted map. The latter approach achieves a solution that is close to optimal, in that it still aligns itself with the principal axis of the gap, but can do this with pre-computed information.

3.6.3 Field-of-View Around Corners

Next, consider the lead vehicle moving along the edge of an obstacle near a corner as shown in Figure 3-7. As the vehicle approaches the corner of the obstacle, one

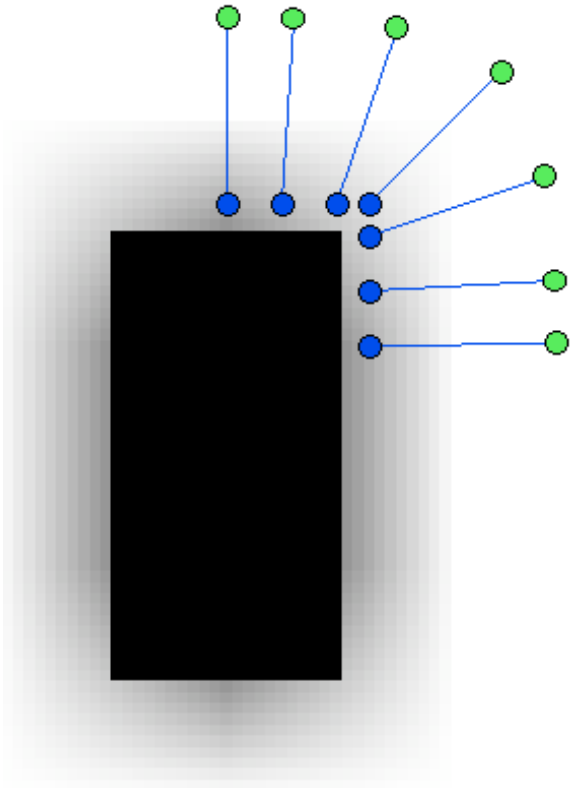


Figure 3-7: Line-of-Sight Alignment at a Corner

possible move for it is to continue following the edge of the obstacle and move around the corner. It is important for the link vehicle to anticipate this possibility and provide communications coverage around the corner in advance of the lead vehicle actually moving there. Naturally, there are situations when this is not possible, but in general a link vehicle using the given cost function will anticipate this maneuver.

The gradient of the map is perpendicular to the face or edge of an obstacle. This property causes the link vehicle to try to keep the communications link perpendicular to the obstacle face. This, however, changes at the corner, where the gradient has to change directions from one obstacle face to the other. The result is a map gradient near the corners that causes the link vehicle to assume a position that has line-of-sight to both faces that form the corner.

3.7 Feasibility of Solution

The theorem presented in this section addresses a key property of the algorithm, namely that the team of vehicles remain in a feasible configuration for all time with respect to the line-of-sight constraints. Feasibility with respect to the obstacle constraints is addressed in [26].

Definition: Plan A vehicle's path in space and time that is defined for all time but that ends in an invariant state. The plan for vehicle i created at time t is denoted as p_i^t.

Theorem 3.1 *If each vehicle has a plan that is feasible for all time, then any new plan created by a vehicle will also be feasible for all time, and it will not invalidate any other vehicle's plan.*

Proof At the beginning of the planning cycle at time t, vehicle 1, the lead vehicle[5], receives p_2^{t-1} from vehicle 2. Vehicle 1 makes a new plan, p_1^t, that is feasible for all time with respect to p_2^{t-1}. Vehicle 1 passes p_1^t to vehicle 2. Vehicle 2 receives p_3^{t-1} from vehicle 3. Vehicle 2 makes a new plan, p_2^t, that is feasible for all time with respect to both p_1^t and p_3^{t-1}. Vehicle 2 passes p_2^t to vehicle 3. This repeats until vehicle N receives p_{N-1}^t from vehicle $N-1$. Vehicle N makes a new plan, p_N^t, that is feasible for all time with respect to p_{N-1}^t and the location of the base station $\mathbf{x_{N+1}}$. This completes planning cycle t.

As described in Section 3.5.1, each plan p_{i-1}^t and p_{i+1}^{t-1} reaches an invariant state before the end of the planning horizon for vehicle i at time t. Therefore the feasibility check performed in the optimization checks the feasibility of plan p_i^t with respect to the invariant states of p_{i-1}^t and p_{i+1}^{t-1} and thus implicitly checks the feasibility for all time.

Plans p_{i-1}^t and p_{i+1}^{t-1} are feasible for all time with respect to p_i^{t-1}. Because constraints are binary, p_i^{t-1} is feasible for all time with respect to p_{i-1}^t and p_{i+1}^{t-1}.

[5]recall that the vehicles in the chain are numbered sequentially, starting with vehicle 1 as the lead vehicle, and ending with vehicle N

Vehicle i can make the decision $p_i^t = p_i^{t-1}$ because p_i^t is a subset of p_i^{t-1} and is therefore also feasible. Each vehicle always has a feasible decision. ∎

3.8 Convergence of Solution

This section presents a theorem that shows that the system converges to a unique constrained local minimum.

Theorem 3.2 *The lead vehicle, and thus the system, converges to a unique local minimum, which may be a constrained minimum.*

Proof At the beginning of each planning cycle, the cost is C_1. By Theorem 3.1, at the beginning of each planning cycle the team is in a feasible configuration. The lead vehicle's cost function is the system cost function, which is the distance from the lead vehicle to the target location. By Theorem 3.1, one feasible plan is to maintain the current position. Therefore, the cost is upper-bounded by C_1. If a plan with a lower cost is feasible, the planner will choose that plan. Therefore, the planner will reduce the cost until it is unable to due to constraints. This solution is a constrained local minimum. Due to the damping term in the cost function, the vehicle will not choose another solution that would otherwise have an equal value. Therefore the chosen solution is an isolated local minimum. ∎

3.9 Summary

This chapter introduced a new distributed, receding-horizon path planning algorithm. The algorithm uses a map of the environment and an optimization to choose the path. The cost function of the optimization acts as the heuristic for the planner by optimizing the vehicle's position for the future movement of the vehicle ahead of it in the chain. The chapter also addressed the issue of feasibility and convergence of the algorithm. A proof of the feasibility of the team's configuration with respect to

line-of-sight constraints was given, as well as a proof that the system converges to an isolated local, and possibly constrained, minimum.

THIS PAGE INTENTIONALLY LEFT (ALMOST) BLANK

Chapter 4

Implementation for Real-Time Surveillance Mission

The algorithm described in Chapter 3 was implemented both in simulation and in the Aerospace Controls Lab's RAVEN flight testbed. The simulation implementation allows for rapid testing of the algorithm in many different scenarios with various environments and a varying number of vehicles. The testbed implementation allows for verification of the algorithm's performance under real-life disturbances that can be difficult to predict or model in simulation.

4.1 The RAVEN System

Flight tests were conducted in the Aerospace Controls Lab's RAVEN (Real-time indoor Autonomous Vehicle test ENvironment) [18, 40], a multi-vehicle testbed allowing for rapid-prototyping of high-level mission management and path planning algorithms (see Figure 4-1(a)). This capability is achieved by using a very accurate Vicon MX motion capture system [41] to produce high bandwidth state estimates of numerous aerial and ground vehicles, as well as in-house vehicle controllers to provide low-level control and stabilization of the vehicle hardware.

The motion capture system uses cameras (Figure 4-1(b)) to detect lightweight reflective dots on the vehicles, such as quadrotors (Figure 4-1(c)), and uses these to

(a) RAVEN Flight Facility

(b) Vicon MX Camera

(c) Quadrotors

Figure 4-1: RAVEN Elements

calculate the vehicles' position and orientation within the 25 by 30 foot test room. This data is transmitted via ethernet to each vehicle's ground based control computer, which in turn commands its vehicle through a COTS R/C transmitter or wireless modem [18, 40]. Along those same lines, the path planning software presented in this chapter is also run off-board, allowing the use of COTS vehicle hardware with minimal requirements for onboard computational capacity. This off-board computation replicates the exact type of computation that would be performed onboard each vehicle, and it is performed off-board simply to ease the integration process given the payload restrictions of the current vehicles.

4.2 Software Implementation Details

The software for the simulation and testbed implementations was written in the C, C++, and MATLAB programming languages. Each language was chosen for its strengths and for the ease of integration with the other languages. Specifically, the optimization function and certain utility functions were written in C, infrastructure and vehicle simulation code was written in C++, and the user interface and base station were written in MATLAB [30]. Furthermore, to correctly model a real system, all the software was written so that a separate executable is run for each vehicle and for the user interface. The separate executables then communicate with each other over a TCP/IP network and through Berkeley sockets.

When a test is started, the base station creates a three-dimensional matrix to represent a discretized map of the environment. The values in the matrix range from 0 to 1, where 0 denotes an obstacle-free cell, 1 denotes a cell with an obstacle, and values in between represent the "blurred" values obtained from performing the convolution. Since the environment is stored as simple binary values denoting where obstacles are, the base station performs the convolution on the map and then sends it to each of the vehicles. Each vehicle is initialized with this map, which it uses to do its path planning and to compute the cost function.

During the test, the vehicles act autonomously except for the lead vehicle which receives updated target locations from the base station. Also, the user interface has the ability to directly poll each of the vehicles for its position. However, this is done solely for simulation and testing purposes, and does not play any role in the algorithm itself.

4.2.1 Cost Function

The cost function must be approximated since, in general, evaluating the continuous integral in the cost function would be difficult. Recall that the cost function has the term

$$\int_0^1 \mathbf{M}((1-\lambda)\mathbf{x_i} + \lambda\mathbf{x_{i-1}})\, d\lambda.$$

With a discretized map of the environment, this cost is approximated with the sum

$$\sum_{j=0}^{P-1}\left[\left(\mathbf{M}\left(\left(1-\frac{j}{P}\right)\mathbf{x_i}+\left(\frac{j}{P}\right)\mathbf{x_{i-1}}\right)\right.\right.$$
$$\left.\left.+\mathbf{M}\left(\left(1-\frac{j+1}{P}\right)\mathbf{x_i}+\left(\frac{j+1}{P}\right)\mathbf{x_{i-1}}\right)\right)\frac{\|\mathbf{x_{i-1}}-\mathbf{x_i}\|_2}{2(P-1)}\right], \quad (4.1)$$

where P is the number of discretization points along the integral approximation and $\mathbf{M}$ is a function that returns the value of the map at its argument. For increased accuracy, the function $\mathbf{M}$ interpolates between the discretized points of the map, rather than just returning the nearest neighbor value.

4.2.2 Line-of-Sight Test

Another important function that the optimization must perform is a line-of-sight check to adjacent vehicles as well as along the planned path. Both of these checks relate to feasibility, communications link feasibility in the first case, and path feasibility in the second case. To perform this check, a deterministic sampling approach is used. The line segment being checked is sampled at evenly spaced points along the segment. If any point on the line is above a threshold value (chosen to be 0.9), then the two endpoints of the line segment are not in line-of-sight of each other.

4.2.3 Optimization Timing and Inter-Vehicle Communication

Two key aspects of the optimization are the Gauss-Seidel style sequential optimization [26] and the accompanying inter-vehicle communication. At the beginning of each optimization cycle, the lead vehicle begins its optimization. Once it is done, it passes its result to the vehicle behind it, and so on down the line. However, for the lead vehicle to begin its optimization it needs the position and most current plan of the vehicle behind it (for feasibility checks).

One simple solution would be to start the optimization with the last vehicle by

having it pass forward its current position, and so on until each vehicle (except the lead vehicle) has passed forward its current position. However, there are some important flaws with this approach. Keep in mind that the vehicles do not necessarily stop moving when they perform their optimization. Instead, they keep moving along their previous plan. Thus, it is important for a vehicle to receive the position and plan of the vehicle behind it just before it begins the optimization routine, and this is achieved by having each vehicle request, at the appropriate time, this information from the vehicle behind it.

Inter-vehicle communication also serves as the timing and synchronization mechanism for the optimization. Only the lead vehicle has to determine when to start the optimization in each cycle since it is the only vehicle that doesn't have a vehicle ahead to trigger it. When the optimization timer on the lead vehicle expires, it requests the position and plan of the vehicle behind, and then starts optimizing using the value it receives along with the current target location that it needs to move to. Once the optimization is complete, the lead vehicle sends its final result to the link vehicle behind it, which in turn is triggered to start the optimization. This triggering works its way down the line to the last vehicle. The advantage of this setup is that it avoids using timers on all vehicles, which simplifies the code that is being run. While this chapter only describes the algorithm with one iteration per optimization cycle, multiple iterations could be performed during each cycle. This concept is discussed in Section 5.2.

4.3 Simulation Results

This section presents the results of simulations of unmanned vehicle systems that use the reconfiguration algorithm presented in Chapter 3. Both urban and mountainous littoral environments and environments with varying length and time scales are shown in order to demonstrate the applicability of this algorithm to multiple situations.

Table 4.1: Optimization Parameters for Littoral Scenario

Parameter	**Value**
UAV/USV Speed	10 m/s
Horiz. Planning Radius	1000 m
Vert. Planning Distance	350 m
α	0.5
β	0.05
γ_{max}	0.0
T_{opt}	105 s

4.3.1 Littoral Environment

This scenario shows how a military expeditionary unit might use a heterogeneous team of unmanned vehicles to reconnoiter a coastline with fjords or other steep terrain. To remain stealthy but also flexible, the unit deploys both unmanned *surface* vessels (USV) and unmanned *aerial* vehicles (UAV). The USVs can get closer to the target of interest while remaining stealthy, while the UAV can act as a (relatively) long-range communications relay by flying at an appropriate altitude above obstacles.

This scenario is demonstrated in an environment created from a digital elevation model obtained from NOAA's GLOBE database [16]. The terrain is shown in Figure 4-3; darker colors represent higher terrain. It contains a coastline with numerous islands and peninsulas of various heights, ranging from nearly flat near the northwest corner, to very steep and high near the southeast corner.

The operator of the vehicles (the expeditionary unit) is located at the northwest corner of the map, and the target of interest is the area at the southeast corner of the map, as well as the path enroute to this location. As mentioned previously, both USVs and UAVs are used. Specifically, two USVs are deployed as the lead and first link vehicle, and one UAV is deployed to act as the second link vehicle. This allows the surface vessels to stealthily approach the target, while allowing the UAV to remain farther away from the target area. The optimization parameters used for this scenario are shown in Table 4.1.

The operations area shown in Figure 4-3 is 71 km from east to west by 85 km

from north to south and the maximum terrain elevation is 1400 m, which is also the altitude limit for the UAV.

The evolution of the three vehicles' paths is shown in Figure 4-3 and the vertical profile of the vehicles is shown in Figure 4-2. Initially, the three vehicles convoy together and remain in a tight group (Figure 4-3(a)). Eventually, however, the UAV reaches its communications range limit and doesn't proceed away from the base anymore (Figure 4-3(b)). Initially it tries to remain at a low altitude, as well as at its current location. However, as the two USVs continue towards the goal, terrain begins blocking the line-of-sight between the USVs and the UAV, which causes the UAV to gain altitude in order to have a better view of the USVs (Figure 4-2). Soon afterwards the two USVs start to travel along the channel on the eastern side of the environment. The two land features that form this channel are the two tallest features on the map, and the one to the west easily blocks communication between the link USV and the UAV. To compensate for this, the UAV readjusts its position by flying to the other side of the north-central island to obtain a better line-of-sight towards the link USV, and the link USV stops proceeding further down the channel due to line-of-sight and range limits (Figure 4-3(d)). However, the lead USV can continue to the target location and achieve its objective.

4.3.2 Urban Environment

This second scenario demonstrates the use of the reconfiguration algorithm in an urban environment with buildings of various heights. This time, the lead vehicle is an unmanned *aerial* vehicle (UAV), but due to sensor limitations, it generally still needs to fly low to be effective. The link vehicles are also UAVs and are allowed to fly at any altitude up to a maximum specified altitude. The optimization parameters for this scenario are shown in Table 4.2.

The area of operation depicted on the map is 6 km by 8 km, and the altitude of the UAVs is limited to 1100 m. The operator is located at the southwest corner and the lead UAV is deployed to the northeast corner before returning back to the operator. The UAVs are range limited to two kilometers, and this will play a role in

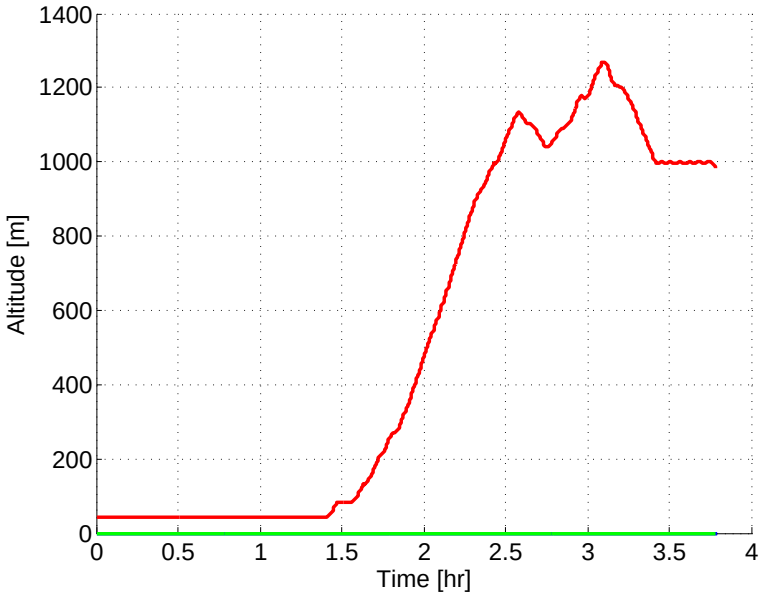


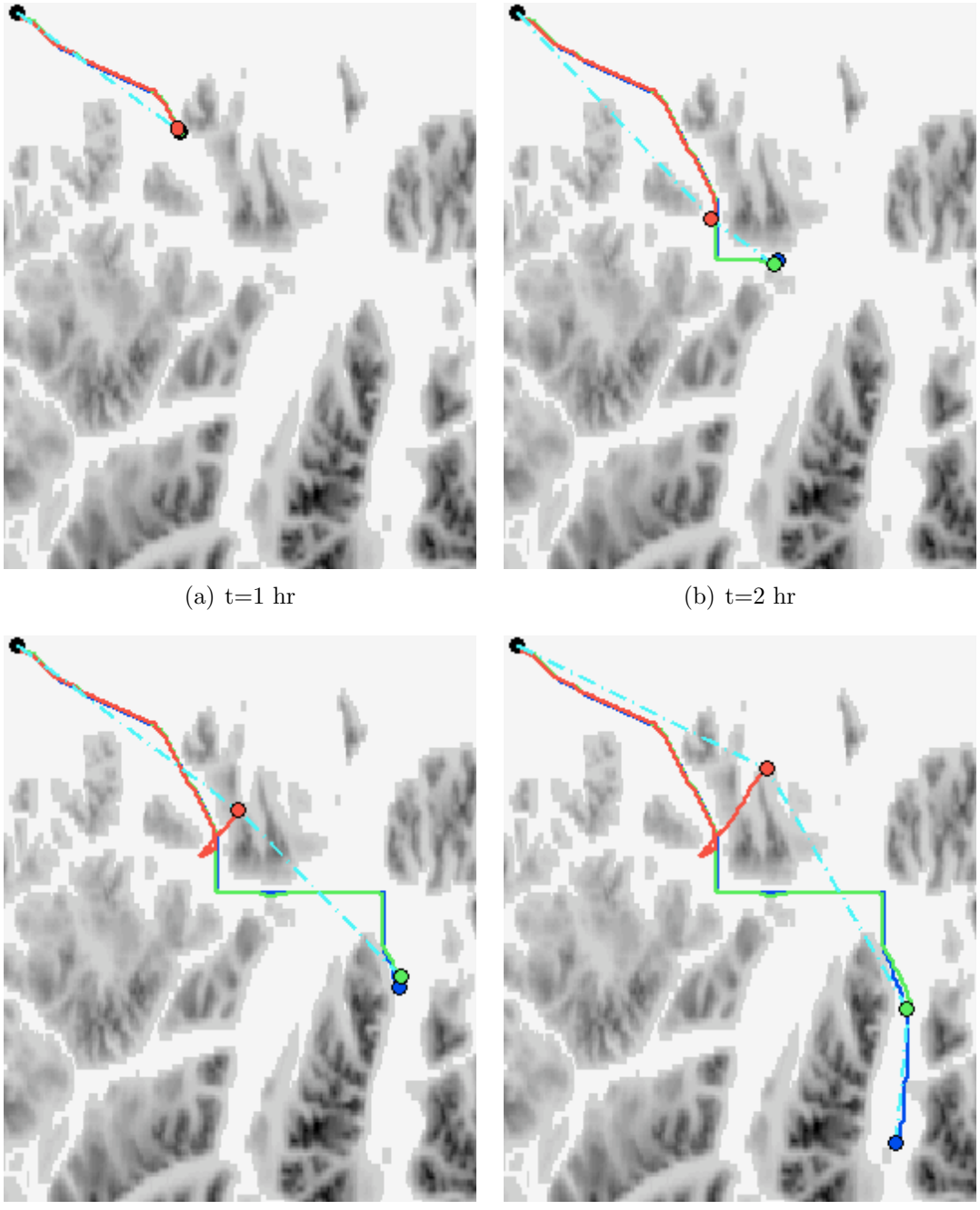
Figure 4-2: Vertical Profile of Vehicles – Littoral Scenario

Table 4.2: Optimization Parameters for Urban Scenario

Parameter	**Value**
UAV Speed	13.9 m/s
Horiz. Planning Radius	300 m
Vert. Planning Distance	100 m
α	0.5
β	0.05
γ_{max}	0.4
T_{opt}	11 s

how the link vehicles provide the communications service.

Initially, all the vehicles stay low to the ground because they can provide adequate communications service to the lead vehicle from this altitude (Figures 4-4 and 4-5(a)). However, as the lead UAV travels further away, the communications chain becomes more and more circuitous, which forces the link UAVs to reconfigure by gaining altitude and providing the communications link over the buildings, rather than between them (Figure 4-5(b)). Next the lead UAV is tasked to fly to a high altitude to get an overview of the eastern part of the city before returning to low-level flying (Figure 4-5(c)). The lead UAV then flies down low between buildings. At this point,



(a) t=1 hr

(b) t=2 hr

(c) t=3 hr

(d) t=3 hr 50 min

Figure 4-3: Horizontal Vehicle Path Evolution – Littoral Scenario

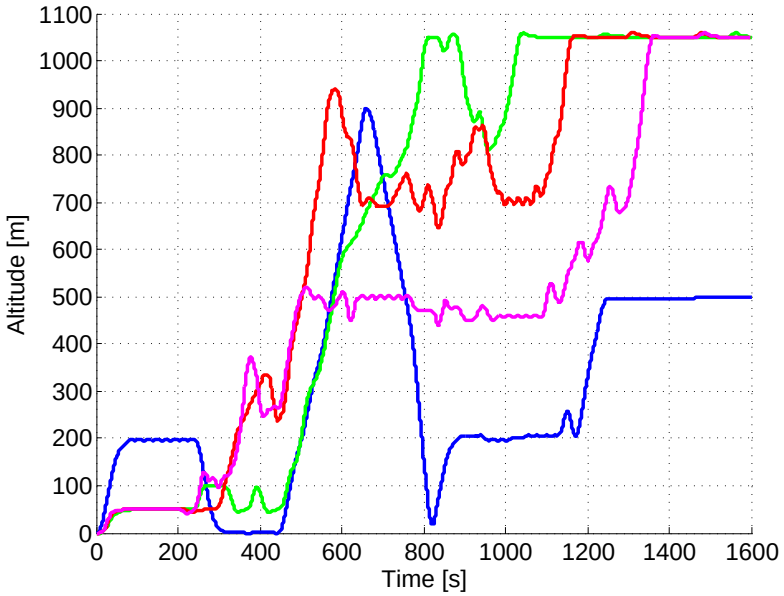
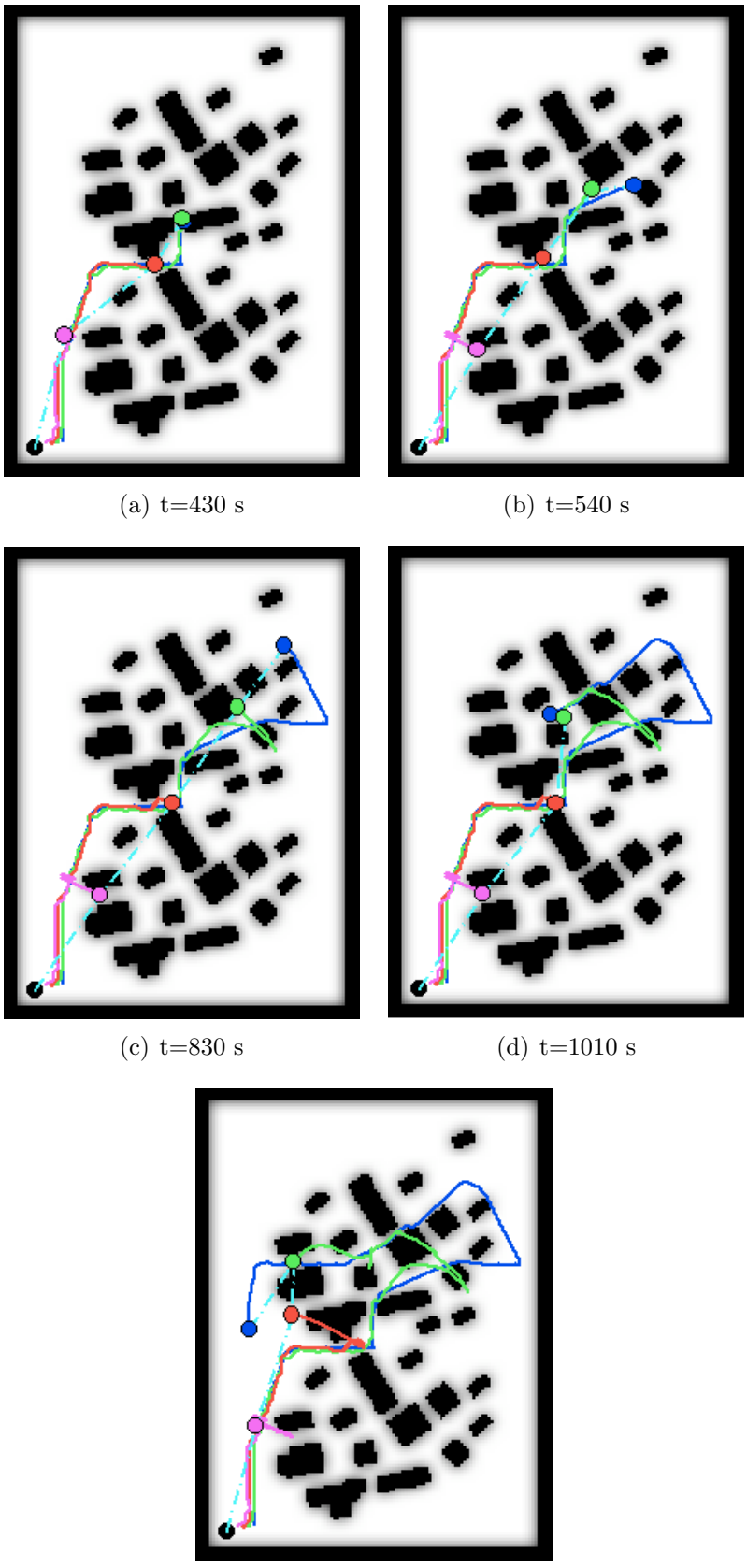


Figure 4-4: Vertical Profile of Vehicles – Urban Scenario

the chain is stretched to its maximum length, and the green link vehicle is just able to provide a communications link from above the lead UAV (Figure 4-5(d)). Finally, when the lead UAV is tasked to return to base, it exits the city to the west. However, there is a tall building (T-shaped building) blocking the communications chain. As the chain becomes more circuitous, the *gamma* term is dynamically increased, which forces the link UAVs to gain altitude and free the communications chain by going over the building before returning back to base too (Figure 4-5(e)).

This scenario shows the flexibility of the reconfiguration algorithm to deal with circuitous routes through a city, especially when aided by the modifications made for urban environments. When possible, the link vehicles provide the communications service from a low altitude between buildings, but when the communications chain gets stretched to its length limit, the UAVs reconfigure and provide the communications link via a straighter route over the top of the buildings.



(a) t=430 s (b) t=540 s

(c) t=830 s (d) t=1010 s

(e) t=1260 s

Figure 4-5: Horizontal Vehicle Path Evolution – Urban Scenario

Table 4.3: Optimization Parameters for Building Exploration Scenario

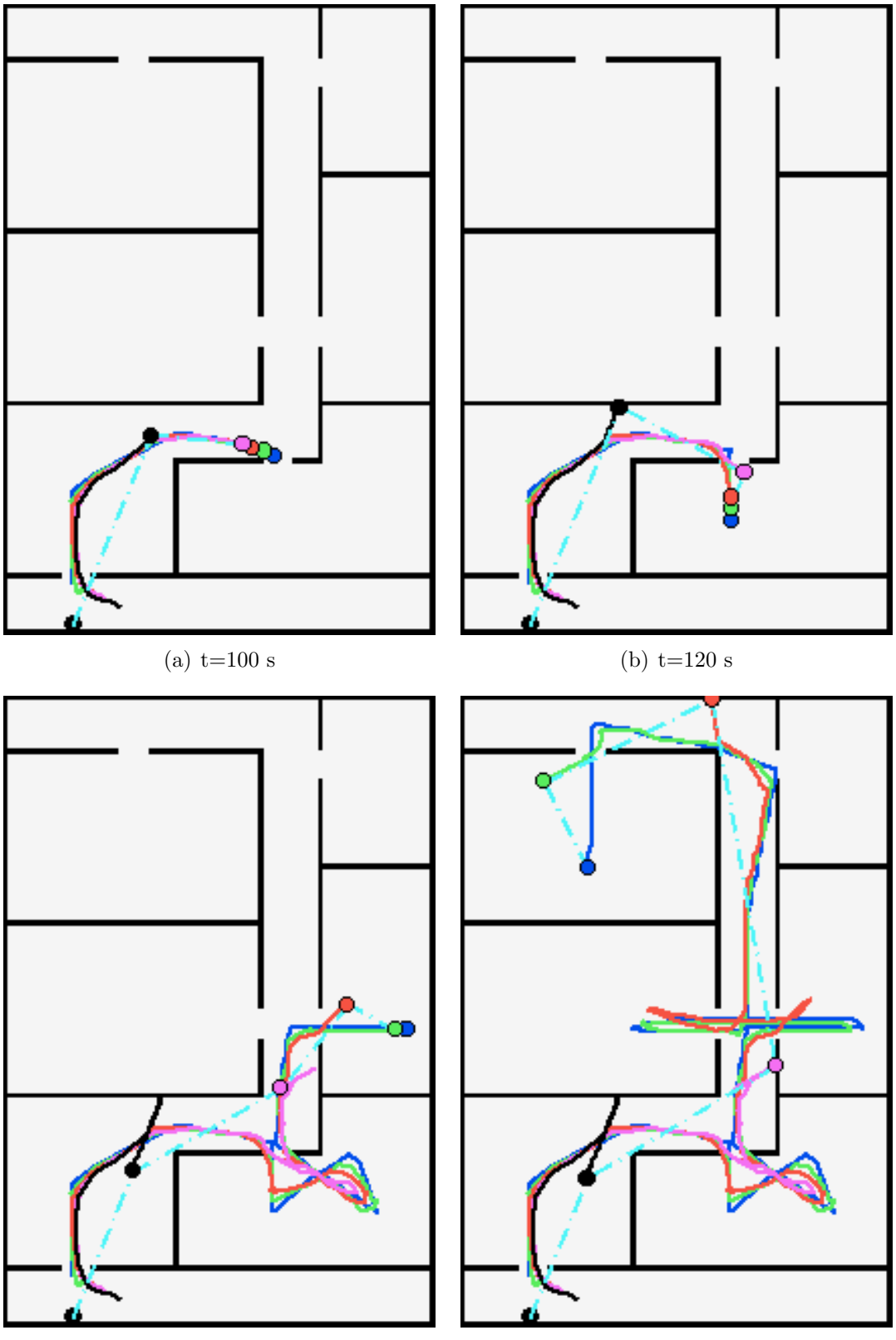
Parameter	**Value**
UGV Speed	0.75 ft/s
Planning Radius	3 ft
α	0.1
β	0.01
T_{opt}	3 s

4.3.3 Building Exploration

This scenario applies the reconfiguration algorithm to a team of unmanned ground vehicles (UGV) exploring a building, which is similar to the scenario considered in [32]. As usual, the team is set up in the same configuration with one lead UGV and several link UGVs (four in this case). Communications range does not play a factor in this scenario, but due to the winding corridor and the narrow doorways, line-of-sight blockage is a significant issue. The lead vehicle is sent along a path that explores each room in the building, starting with the closest one and ending with the farthest one. The UGV operator remains outside of the building at the southern part of the map. The optimization parameters used for this scenario are shown in Table 4.3. The building is 75 ft. wide by 100 ft. long. Corridors are 10 ft. wide and doorways are 5 ft. wide.

Because this scenario involves only UGVs and because the building has a ceiling, the problem becomes a two-dimensional problem. This does not mean that the problem is easier; in fact, because the vehicles can not pass up and over obstacles, it can be a more difficult problem.

Four snapshots of the vehicles' paths are shown in Figure 4-6. Initially the five vehicles convoy in together, but soon the tail vehicle (black) must stop so as not to lose line-of-sight with the base (Figure 4-6(a)). A short time later, the other four vehicles enter the first room. Passing through this doorway greatly restricts the field-of-view from the black vehicle to the magenta vehicle. To compensate, the black vehicle adjusts its position northwards to get a better view through the doorway



(a) t=100 s (b) t=120 s

(c) t=280 s (d) t=550 s

Figure 4-6: Horizontal Vehicle Path Evolution – Building Exploration Scenario

(Figure 4-6(b)). Once the team continues down the hallway, the black vehicle must once again adjust its position in the opposite direction to get the best field-of-view (Figure 4-6(c)). The magenta vehicle now also can't stay with the convoy and hangs back. To get the best view into the second room, it moves as far to the west of the corridor as possible. At the end of the exploration sequence, the team achieves the configuration shown in Figure 4-6(d). Once again, the magenta vehicle has adjusted its position to gain the best field-of-view around the corner in the corner, thereby allowing the red vehicle to advance as far forward as possible. This scenario shows the ability of the algorithm to reconfigure the team of vehicles in real time to give the lead vehicle as much freedom of movement as possible.

4.4 Flight Test Results

This section presents the results of a flight test of the reconfiguration algorithm performed in the RAVEN flight testbed. The main purpose of this demonstration is to show how the system behaves with real-world dynamics and disturbances from wind and other factors. It also validates that the reconfiguration algorithm can be run in real-time.

The operations area consists of an area that is 5.34 m from east to west, 8.65 m north to south, and 2 m high. Several obstacles are placed in the room and modeled in the environment map. The two large obstacles in the northern part of the map closest to the base are very tall obstacles that can not be flown over. They form a gap through which the vehicles must travel. The group of obstacles to the south range from 1 m to 1.2 m in height and also form two gaps. The environment is discretized into cells with side of 0.15 m length. Because the obstacles have vertical sides, the convolution was performed in the way described in Section 3.4.

Three vehicles are used for the test, one model truck acting as the lead vehicle, and two quadrotors acting as link vehicles. The quadrotors are limited to a minimum altitude of 0.3 m. Other parameters are shown in Table 4.4. The two quadrotors are flown autonomously by autopilots in the RAVEN system based on waypoints

Table 4.4: Optimization Parameters for RAVEN Flight Test

Parameter	**Value**
Vehicle Speed	0.2 m/s
Horiz. Planning Radius	0.5 m
Vert. Planning Distance	0.25 m
α	0.5
β	0.05
γ_{max}	0.0
T_{opt}	3 s

received from the reconfiguration algorithm. The lead truck is controlled manually, which models one concept of operation where the operator dynamically directs the path of the vehicle. This also demonstrates the ability of the algorithm to deal with uncertainty in the lead vehicle's path.

After the two link vehicles perform their takeoff, they hover at a height of 0.5 m. Once the lead vehicle starts moving through the first gap, the two link vehicles fall in behind (Figure 4-8(a)). As the lead vehicle continues along its path, the rear link vehicle (red) begins to gain altitude because of the vertical cost gradient caused by the convolution (Figure 4-7). The rear link vehicle also reaches its line-of-sight constraint. The other link vehicle (green) stays at its low initial altitude and follows the lead vehicle (Figure 4-8(b)). At $t = 60$ s the red link vehicle has a temporary incursion into the obstacle due to a disturbance (Figure 4-8(c)). However, the map has slightly enlarged representations of the obstacle to provide a safety border and so the vehicle does not contact the actual obstacle in the flight test room. After this incursion, the vehicle remains outside of the obstacle area for the remainder of the test.

Next, the lead vehicle travels through the southern obstacle gap (Figure 4-8(d)). At this point, the green link vehicle gains altitude to go up and over the obstacles (Figure 4-7). Finally, the the simulation ends with the lead vehicle traveling back to the base and the green link vehicle catching up with it (Figure 4-8(e)).

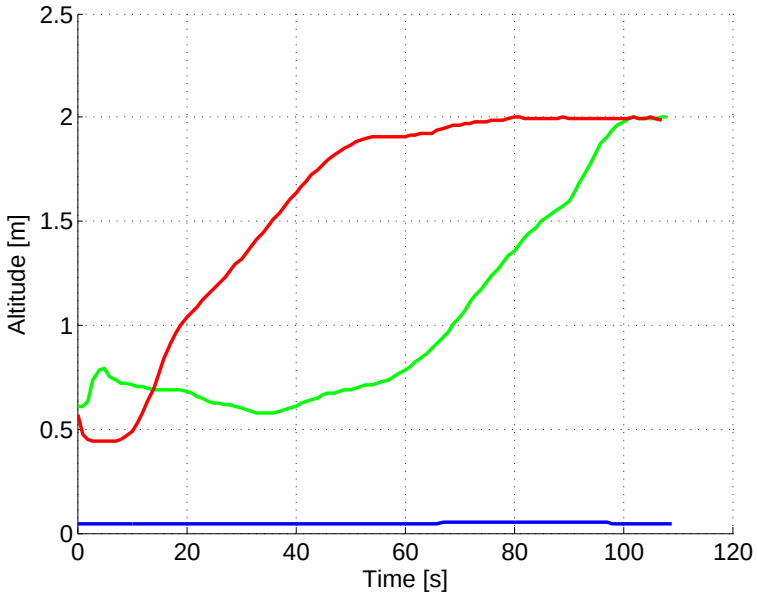
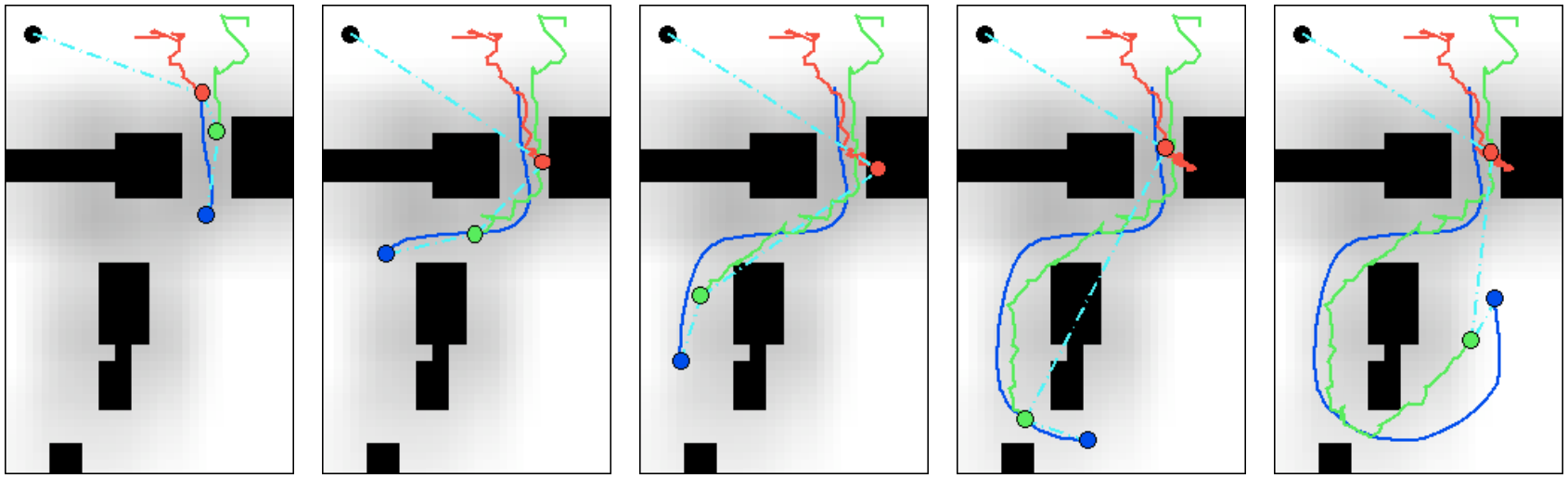


Figure 4-7: Vertical Profile of Vehicles – RAVEN Flight Test

4.5 Comparison of Deployment and Reconfiguration Algorithms

While the two algorithms presented in Chapter 2 and Chapter 3 differ, they address similar problems. This section attempts to compare the two by modifying the cost function of the deployment algorithm so that the modified algorithm behaves as is desired for the reconfiguration algorithm. Because the deployment algorithm provides an optimal answer (within the limits of the optimization package), its solution provides a baseline to which the reconfiguration algorithm can be compared.

Several factors in the reconfiguration algorithm can affect its outcome. First, the initial locations of the vehicles can be perturbed. Second, because the algorithm is distributed over several computers, any variations in the computation time (caused by other processes running on the same computers) or the communications time (caused by varying network traffic), might also affect the solution. By running the reconfiguration algorithm several times with various perturbations, a comparison can be made between the baseline solution provided by the modified deployment algorithm, and the various solutions provided by the reconfiguration algorithm.



(a) t=20 s (b) t=40 s (c) t=60 s (d) t=80 s (e) t=110 s

Figure 4-8: Horizontal Vehicle Path Evolution – RAVEN Flight Test

4.5.1 Cost Function Modification

One main difference between the behavior of the deployment algorithm and the behavior of the reconfiguration algorithm is that the deployment algorithm attempts to minimize the energy used by the system over the entire mission. As a result, in the final configuration the link vehicles hang as far back as possible. The reconfiguration algorithm is different because it is always expecting to have to go further, and so the link vehicles always push as far forward as possible. The cost function for the deployment algorithm is given in Eq. 2.8, but is repeated here:

$$\min J = (t_f - t_0) + \alpha \int_{t_0}^{t_f} \sum_{i=1}^{N} ({F_{x,i}}^2 + {F_{y,i}}^2)\, dt.$$

The two terms in the cost function minimize the mission time and minimize the energy expenditure. This cost function is modified by removing the penalty on energy usage, and instead penalizing the distance between two vehicles. This gives a new cost function of

$$\min \tilde{J} = (t_f - t_0) + \int_{t_0}^{t_f} \sum_{i=1}^{N-1} \left[(N-i)\alpha \left((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 \right) \right] dt. \tag{4.2}$$

The factor α is chosen to be 0.01 so that the mission time dominates the cost function. Also, links further ahead are penalized more than the links further back in the chain. This is to prevent the middle vehicles from having multiple optimal solutions, and instead forces *all* the vehicles to move forward. While this cost function does not exactly replicate the cost function used in the reconfiguration algorithm, in practice it achieves a similar goal, especially with respect to the final configuration of the vehicles.

4.5.2 Simulation Results

To show the actual comparison between the two algorithms, a scenario is chosen, and the optimal solution is solved for by the deployment algorithm with the mod-

Table 4.5: Optimization Parameters for Comparison Simulations

Parameter	**Value**
Vehicle Speed	1 m/s
Planning Radius	3 m
α	0.5
β	0.05
T_{opt}	3 s

ified cost function. Next, the reconfiguration algorithm is run multiple times. The reconfiguration algorithm results show tests where the vehicles have the same initial positions as in the deployment algorithm, as well as tests where the initial position of the vehicles is perturbed. The main goal of these results is to show that the reconfiguration algorithm converges to the same path and final configuration from different initial conditions. The solution from the deployment algorithm is an optimal baseline solution that the other solutions can be compared to. The results also show that the reconfiguration algorithm solutions are close to this optimal solution, both in the path taken and in the final configuration that is achieved.

The scenario uses an environment that is similar to the ones in Chapter 2. As before, a two-dimensional problem is shown. The environment is 50 m by 50 m with four circular obstacle. The base is located at the southwest corner, and the goal is at the northeast corner. The deployment algorithm is given an initial solution that passes through the gap between the southwestern obstacle and the central obstacle, and the gap between the northwestern obstacle and the central obstacle. In the reconfiguration algorithm, the vehicle is given one intermediate waypoint which is just to the west of the central obstacle. The parameters used for the reconfiguration algorithm are shown in Table 4.5.

Figure 4-9 shows four different simulation cases, and in all the cases, the solid line shows the paths calculated by the deployment algorithm. The same solution is used as the baseline in all four cases. The paths shown by markers only are the paths of the vehicles when using the reconfiguration algorithm.

In the baseline solution, the lead vehicle (blue) follows a path that is very close

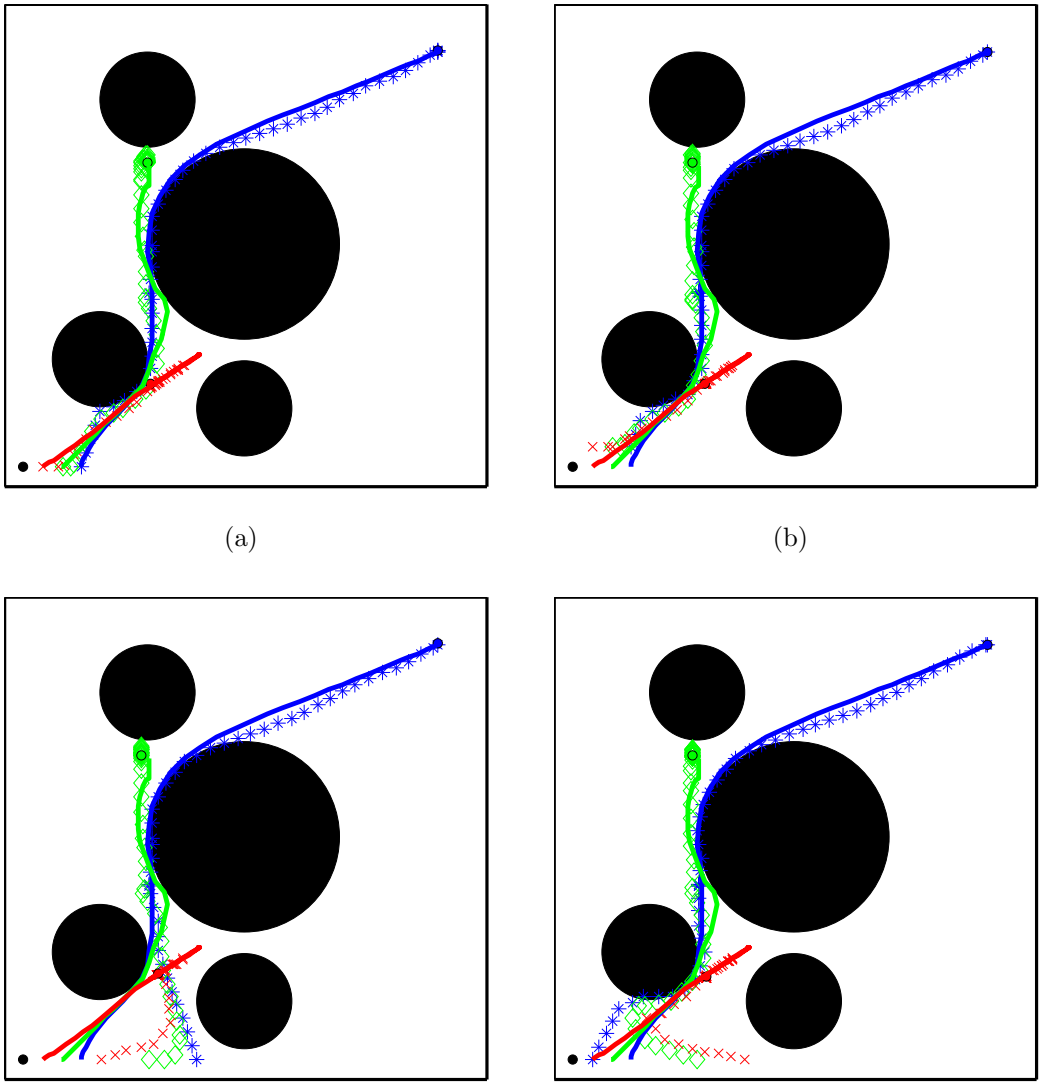
to the shortest time path from its starting location to the final goal. The green vehicle follows the lead closely at first, and then flies to its final location near the northernmost obstacle. This path maximizes the field of view from the first link vehicle to the lead vehicle by minimizing the distance between the two. However, near the end of the path, this link vehicle is constrained by the requirement to maintain line-of-sight with the rear link vehicle. The rear vehicle (red) initially flies northeast before returning to its final position next to the westernmost obstacle. From its final location it allows the middle vehicle to advance as far eastward as possible. For the northeastern part of the trajectory, it is constrained by the requirement to stay in line-of-sight with the base, which is why the rear link vehicle travels along this constraint.

Figure 4-9(a) shows the result of both algorithms when the vehicles are started from the nominal initial positions. The reconfiguration solution very closely follows the deployment solution. This validates that, given the same initial conditions, the two algorithms produce very similar solutions and that the reconfiguration solution is close to the optimal solution. The other results show that, even with varying initial conditions, the reconfiguration solution converges to the same paths and the same final configuration as in the optimal solution. This behavior is expected because both link vehicles approach the constrained local minimum of their local optimization and the lead vehicle flies along the shortest path to the target location.

In the scenario shown in Figure 4-9(b), the initial positions of all the vehicles are shifted north by a small distance. Because this perturbation is small, the paths quickly converge to the optimal path. In the scenario shown in Figure 4-9(c), the initial positions are shifted to the east and spread out. As expected, the lead vehicle flies directly to the the intermediate waypoint where it meets the optimal path. The link vehicles fly towards the lead vehicle to minimize their distance to the vehicle ahead. Once they reach their respective constraints, they fly along these constraints, thus minimizing their local cost functions. Finally, in the scenario shown in Figure 4-9(d), the starting locations of the lead vehicle and the rear link vehicle are interchanged and all the starting locations are spread out. Once again, the lead vehicle flies towards

its intermediate waypoint and the link vehicles exhibit the same behavior of flying towards the lead vehicle and then flying along their respective constraints.

In all the cases shown, the reconfiguration algorithm produces a solution where the lead vehicle flies along the shortest path to the goal and the link vehicles converge to the baseline paths. Also, all the vehicles in all the scenarios end at the same final location, which is the local minimum for each of their respective optimizations.



(a) (b)

(c) (d)

Figure 4-9: Vehicle Path Evolution – Comparison Simulations

Chapter 5

Conclusion

5.1 Summary

This thesis investigated path planning for teams of vehicles that form communications chains. The main challenge faced in this problem is meeting the constraint that adjacent vehicles maintain a clear line-of-sight between each other. In highly constrained environments, such as urban or mountainous areas, there are many obstacles that can block the line-of-sight.

This thesis presented two algorithms, the *deployment* algorithm and the *reconfiguration* algorithm, to address this path planning problem. Both algorithms addressed similar variations of the problem in two different ways, but generally with the same constraints.

5.1.1 Deployment Algorithm

The deployment algorithm, presented in Chapter 2, considered the problem of deploying a team of vehicles from a common base location to a final configuration with the lead vehicle at a specified target location and the other vehicles placed as necessary to provide a communications link from the lead vehicle to the base. Additionally, the team was constrained to provide this link at all times during the deployment.

This problem was formulated as an optimal control problem and solved using a

Gauss pseudospectral optimization algorithm. The initial guess to the optimization was created using a modified Rapidly-Exploring Random Tree algorithm. The unique feature of this initial guess approach is that the RRT solution only provides a small subset of the entire initial guess, namely the path of the lead vehicle. The rest of the states and the controls are created by a heuristic function.

Several simulations showed typical solutions obtained by the deployment algorithm. They also showed that the optimization inherently minimizes the number of vehicles used in the final solution, even if the initial guess uses more than the minimum number of vehicles. A modified version of the deployment algorithm was used in Chapter 4 for comparison with the reconfiguration algorithm.

5.1.2 Reconfiguration Algorithm

The reconfiguration algorithm, presented in Chapter 3, is an algorithm intended for real-time use onboard a team of vehicles creating a communications link. It achieves this goal by solving the path planning problem over a short planning horizon and performing this planning cycle frequently. The path planning itself is formulated as an optimization that minimizes a heuristic while meeting all the applicable constraints, such as the line-of-sight requirement. Each vehicle creates its own plan, but takes into account what adjacent vehicles are doing or planning to do.

One main contribution to this solution method was a new heuristic and cost function used in the receding horizon path planner. Because the planner plans over a short horizon, the heuristic must account for the future movement of the other vehicles. It does this by optimizing each vehicle's position in such a way that it allows the vehicle ahead of it in the chain to move with the greatest freedom while meeting the constraints. The lead vehicle in the chain simply tries to get as close to the goal as possible. The greatest freedom of movement is achieved by maximizing the symmetric field-of-view, which is a measure of how well one vehicle can see another vehicle.

A second contribution was the development of the environment representation used by the heuristic mentioned above. A binary obstacle map of the environment

was "blurred" out by performing a convolution on it with a Gaussian convolution kernel. This has the effect of adding a boundary to obstacles. The boundary has high values near the obstacle, and a lower value farther away from the obstacle. This encodes in the map information about how far a certain point on the map is from obstacles. This information is used in the heuristic calculation.

5.1.3 Simulations and Flight Tests

Chapter 4 showed simulations and flight tests of teams of vehicles implementing the reconfiguration algorithm. The simulations showed heterogeneous teams operating in various types of environments. In the first scenario, a team of unmanned surface vessels and unmanned aerial vehicles explored a mountainous coastal terrain. This scenario demonstrates the applicability of the algorithm to heterogeneous teams of vehicles where some of the vehicles are constrained to move in two dimensions only.

In the second scenario, a team of aerial vehicles explored an urban area with tall buildings. This scenario adds obstacles with vertical sides, which can be more difficult to deal with than obstacles with sloped sides. To deal with this additional difficulty, two modifications were made. First, the convolution step was altered to simulate slopes sides around buildings. Second, another term was added to the heuristic that promoted altitude gain in certain situations where it is beneficial to the team.

In the third simulation scenario, a team of ground vehicles explored the inside of a building. Because the scenario is two-dimensional, the movement of the vehicles is restricted more than in a three-dimensional problem, which can make it more difficult to find an appropriate solution.

Flight tests in MIT's RAVEN testbed demonstrated the reconfiguration algorithm in use on a real system of unmanned aerial vehicles. These tests validated that the algorithm can be used in real time systems and demonstrated how the system behaves under disturbances.

Lastly, this chapter compared simulations of the reconfiguration algorithm to a baseline solution provided by a modified version of the deployment algorithm. The deployment algorithm was modified to behave like the reconfiguration algorithm and

then the reconfiguration algorithm was simulated with various perturbed initial states. This comparison showed that the reconfiguration algorithm produces paths that are close to the optimal baseline path provided by the deployment algorithm, and that even with perturbed initial states, the reconfiguration algorithm converges to the same path and final configuration as in the baseline case.

5.2 Future Work

This thesis has developed two algorithms for unmanned vehicle systems. There are several interesting continuations and extensions of this work that can be pursued, some of which are discussed in more detail in the following sections. Of course, the implementation of these algorithms on more complex unmanned vehicle systems could further validate the applicability of the algorithms to real systems.

5.2.1 Incorporating Advanced Knowledge of Path

The basic version of the reconfiguration planning algorithm assumes that the plan is created over a short planning horizon and that the invariant state is in close proximity to the initial location of the optimizing vehicle. However, it might be possible, in some cases, to improve the performance of the algorithm by incorporating advanced knowledge of the lead vehicle path and/or the link vehicle paths and using a longer planning horizon. In this case, the "short distance" assumption no longer applies and the cost function cannot be evaluated only at the invariant state. Instead, the cost function needs to be evaluated at several points along the path. Presented below are two proposed modifications.

Average Cost

The new cost function must measure the cost along the entire path, and not just at one point. One option is to evaluate the original cost function at discrete points that define the plan, and then average these values. Taking this approach will provide a

good overall solution, but may allow points in the plan that have lower robustness to tradeoff with other points that have higher robustness.

Maximum Cost

Another cost function takes the maximum value of the original cost function evaluated at discrete points in the plan. This provides a solution that has a good communications configuration at all points in the plan, with the advantage that if the optimizer must replan at any point it is already in a robust configuration to do so.

5.2.2 Communications Chain Shortening

Section 3.4 discussed a modification of the reconfiguration algorithm's cost functions to keep the communications chain from getting "stuck" on tall buildings and becoming circuitous. While this worked well in the simulation scenarios shown, there may be other cases, such as complex two-dimensional environments where this approach does not work. In these cases, the communications chain could be shortened by having a vehicle communicate directly with a non-adjacent vehicle in the chain. While this is certainly the case when all the vehicles are close together at the beginning of the mission, it can also happen if the chain wraps itself around an obstacle and doubles back on itself (Figure 5-1). In such a situation, it is advantageous to shortcut the communications chain and have the vehicle communicate directly to a vehicle further back in the chain. This then frees up intermediate vehicles to reposition themselves at a better location before rejoining the chain, and results in a straighter chain that has more slack to extend further.

Despite the advantages of shortening the communications chain, there are several pitfalls that must be avoided. During the time that a vehicle is not part of the chain and is repositioning itself, the overall chain length is shorter, which could limit the versatility of the chain. Thus, a careful decision must be made about when the chain can be broken. The general criteria should ensure than the newly formed link is robust to being broken while the freed vehicles are repositioning themselves.

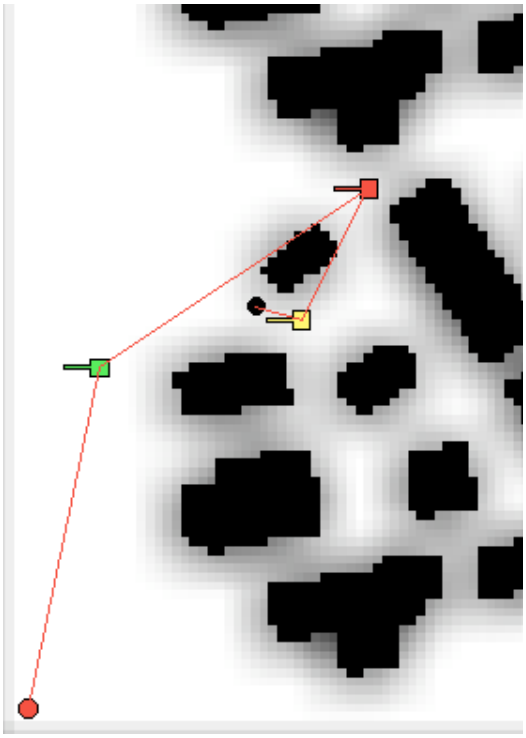


Figure 5-1: Communications Chain Wrapped Around Obstacle

The second aspect of this approach that must be considered is how the freed vehicles reposition themselves. The proposed solution is to have each vehicle fly to the vehicle ahead in the chain that is still part of the main chain. This puts it in the most versatile position from a communications perspective. From a really close distance, the field-of-view is essentially unlimited, and there are no range issues. Second, because the vehicle ahead is still in the chain, positioning the freed vehicle at the same location allows for straightforward insertion back into the communications chain.

In the example in Figure 5-1, the base is the red dot in the southwest corner and the small black circle is the lead vehicle. The link vehicles are represented by the green, red, and yellow symbols. There are two possible ways to shorten the chain. The single vehicle shortening would entail the red link vehicle breaking out of the chain and flying to meet up with the yellow vehicle's position. The yellow vehicle would temporarily communicate with the green link vehicle. The other option is for the lead vehicle, in black, to communicate directly with the green link vehicle and

have both the red and yellow link vehicles rendezvous with the lead vehicle before rejoining.

5.2.3 Non-Holonomic Vehicles

One limitation to the reconfiguration algorithm is that it assumes holonomic vehicles. An interesting and useful extension to the algorithm would be to allow for non-holonomic vehicles with a non-zero minimum speed such as airplanes. The planning algorithm would have to take these limitations into account both for planning and constraint satisfaction purposes.

5.2.4 Unknown Environment Map

Another limitation of the reconfiguration algorithm is that it assumes a known environment. While this may certainly be true in many cases, a system that does not fully rely on this assumption would be more useful. If a vehicle has sensors onboard that can sense obstacles, then the environment map can be updated online by updating the binary map and then redoing the convolution over the affected parts of the map.

THIS PAGE INTENTIONALLY LEFT (ALMOST) BLANK

Appendix A

Optimal Field-of-View

This appendix presents an example to show that the line integral term minimization presented in section 3.2.1 provides the greatest symmetric field-of-view. Recall the simple environment shown in Figure 3-5 that consists of two rectangular obstacles with a gap in between them. Also consider the associated binary map function:

$$\mathbf{M_b} = \begin{cases} 1 & \text{if } -3 \leq x \leq -1, \quad -2 \leq y \leq 2 \\ 1 & \text{if } 1 \leq x \leq 3, \quad -2 \leq y \leq 2 \\ 0 & o.w., \end{cases} \tag{A.1}$$

and the uniform kernel

$$K = \begin{cases} \frac{1}{4} & \text{if } -1 \leq x \leq 1, \quad -1 \leq y \leq 1 \\ 0 & o.w. \end{cases} \tag{A.2}$$

The convolution of the binary map and the kernel results in

$$\mathbf{M} = \begin{cases} \frac{(x+4)(y+3)}{4} & \text{if } -4 \le x \le -2, \; -3 \le y \le -1 \\ \frac{-x(y+3)}{4} & \text{if } -2 \le x \le 0, \; -3 \le y \le -1 \\ \frac{x+4}{2} & \text{if } -4 \le x \le -2, \; -1 \le y \le 1 \\ \frac{-x}{2} & \text{if } -2 \le x \le 0, \; -1 \le y \le 1 \\ \frac{-(x+4)(y-3)}{4} & \text{if } -4 \le x \le -2, \; 1 \le y \le 3 \\ \frac{x(y-3)}{4} & \text{if } -2 \le x \le 0, \; 1 \le y \le 3 \\ \frac{x(y+3)}{4} & \text{if } 0 \le x \le 2, \; -3 \le y \le -1 \\ \frac{-(x-4)(y+3)}{4} & \text{if } 2 \le x \le 4, \; -3 \le y \le -1 \\ \frac{x}{2} & \text{if } 0 \le x \le 2, \; -1 \le y \le 1 \\ \frac{-(x-4)}{2} & \text{if } 2 \le x \le 4, \; -1 \le y \le 1 \\ \frac{-x(y-3)}{4} & \text{if } 0 \le x \le 2, \; 1 \le y \le 3 \\ \frac{(x-4)(y-3)}{4} & \text{if } 2 \le x \le 4, \; 1 \le y \le 3 \\ 0 & \text{if } o.w. \end{cases} \tag{A.3}$$

For the optimization, the position of the lead vehicle will be fixed at $\mathbf{x_1} = [0\ 3]^T$ and the link vehicle will be constrained to the line segment $y_2 = -3, -2 \le x_2 \le 2$. The computation of the line integral can be split into two parts, $x_2 < 0$ and $x_2 \ge 0$.

The original line integral that was given has an integration variable λ that moves along the line. This calculation can be simplified by effecting a change of variables so that the new integration variable y' moves vertically. This allows the integration to be split into three additional parts: $-3 \le y' \le -1$, $-1 \le y' \le 1$, and $1 \le y' \le 3$, where y' is the variable of integration. The change of variable is done as follows:

$$y' = \lambda y_1 + (1-\lambda) y_2 \tag{A.4}$$

$$y' = \lambda \cdot 3 + (1-\lambda) \cdot -3 \tag{A.5}$$

$$dy' = 3ds + 3ds = 6ds. \tag{A.6}$$

Now the line integral (for $x_2 \geq 0$) can be written as

$$\begin{aligned} J_2 = \int_{-3}^{-1} \left[\frac{y'+3}{6}x_1 + \left(1 - \frac{y'+3}{6}\right)x_2\right]\left[\frac{y'+3}{6}y_1 + \left(1 - \frac{y'+3}{6}\right)y_2 + 3\right]\frac{1}{24}dy' \\ + \int_{-1}^{1} \left[\frac{y'+3}{6}x_1 + \left(1 - \frac{y'+3}{6}\right)x_2\right]\frac{1}{12}dy' \\ + \int_{1}^{3} - \left[\frac{y'+3}{6}x_1 + \left(1 - \frac{y'+3}{6}\right)x_2\right]\left[\frac{y'+3}{6}y_1 + \left(1 - \frac{y'+3}{6}\right)y_2 - 3\right]\frac{1}{24}dy', \end{aligned} \tag{A.7}$$

which simplifies to

$$J_2 = \frac{x_2}{12}, \quad x_2 \geq 0.$$

By symmetry, the cost for $x_2 < 0$ is $\frac{-x_2}{12}$. The minimum value of this function is at $x_2 = 0$.

Now that it has been shown that the minimum of the line integral is achieved at $x_2 = 0$, it needs to be shown that this produces the maximum symmetric field-of-view. For x_2 in the range $[-1, 1]$, the corners that most limit the field-of-view are $(-1, 2)$ and $(1, 2)$. The angle from $\mathbf{x_2}$ to $\mathbf{x_1}$ is given by $\psi = \arctan(6/(0 - x_2))$ and the angle to the left and right corners is given by $\phi_1 = \arctan(5/(-1 - x_2))$ and $\phi_2 = \arctan(5/(1 - x_2))$. The FOV to the left of the line-of-sight is $\phi_1 - \psi$ and the FOV to the right is $\psi - \phi_2$. The symmetric FOV is the minimum of these two angles. When $x_2 = 0$, $\psi = \frac{\pi}{2}$, $\phi_1 = 1.768$, and $\phi_2 = 1.373$. The respective derivatives at this point are

$$\begin{aligned} \frac{d\psi}{dx_2} &= \frac{6}{{x_2}^2 + 36} \\ \frac{d\phi_1}{dx_2} &= \frac{5}{{x_2}^2 + 2x_2 + 26} \\ \frac{d\phi_2}{dx_2} &= \frac{5}{{x_2}^2 - 2x_2 + 26}. \end{aligned}$$

From the above equations it can be seen that the FOV to the right of the line-of-sight is the smaller FOV when $x_2 > 0$ and conversely, the FOV to the left is smaller when $x_2 < 0$. Also, since the derivatives of ϕ_1 and ϕ_2 are larger than that of ψ,

the minimum FOV (the symmetric FOV) is maximized when $x_2 = 0$. Thus, the line integral cost function term maximizes the symmetric field-of-view from $\mathbf{x_2}$ to $\mathbf{x_1}$.

Bibliography

[1] Aerovironment. UAS: Wasp III. Available at http://www.avinc.com/uas_product_details.asp?Prodid=4, May 2009. 15

[2] G. S. Aoudé. Two-stage path planning approach for designing multiple spacecraft reconfiguration maneuvers and application to SPHERES onboard ISS. Master's thesis, Massachusetts Institute of Technology, September 2007. 18, 21, 25, 29

[3] G. S. Aoudé, J. P. How, and I. M. Garcia. Two-stage path planning approach for solving multiple spacecraft reconfiguration maneuvers. *Journal of Astronautical Sciences (accepted to appear)*, May 2009. 18, 29

[4] J. Bellingham, A. Richards, and J. P. How. Receding horizon control of autonomous aerial vehicles. In *Proceedings of the 2002 American Control Conference*, volume Vol. 5, pages pp. 3741–3746, 2002. 50

[5] D. A. Benson. *A Gauss Pseudospectral Transcription for Optimal Control*. PhD thesis, Massachusetts Institute of Technology, November 2004. 18, 21, 25

[6] D. A. Benson, G. T. Huntington, T. P. Thorvaldsen, and A. V. Rao. Direct trajectory optimization and costate estimation via an orthogonal collocation method. *Journal of Guidance, Control, and Dynamics*, Vol. 29(No. 6):pp. 1435–1440, November-December 2006. 25

[7] T. X. Brown, B. Argrow, C. Dixon, S. Doshi, R.-G. Thekkekunel, and D. Henkel. Ad hoc UAV ground network (AUGNet). In *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop, and Exhibit*. AIAA, 20-23 September 2004. 20

[8] E. M. Craparo. *Cooperative Exploration under Communication Constraints*. PhD thesis, Massachusetts Institute of Technology, September 2008. 20

[9] Defense Industry Daily. Raven UAVs winning gold in Afghanistan's "commando olympics". *Defense Industry Daily*, 2008. 16

[10] C. Dixon and E. W. Frew. *Advances in Cooperative Control and Optimization*, volume 369 of *Lecture Notes in Control and Information Sciences*, chapter Decentralized Extremum-Seeking Control of Nonholonomic Vehicles to Form a Communication Chain, pages 311–322. Springer Berlin / Heidelberg, 2007. 20

[11] C. Dixon and E. W. Frew. Maintaining optimal communication chains in robotic sensor networks using mobility control. In *RoboComm '07: Proceedings of the 1st international conference on Robot communication and coordination*, pages 1–8, Piscataway, NJ, USA, 2007. IEEE Press. 20

[12] P. Eng. Navy tests unmanned patrol boats. *ABC News* http://abcnews.go.com/Technology/FutureTech/Story?id=99511&page=1, June 2004. 16

[13] I. Garcia and J. P. How. Improving the efficiency of rapidly-exploring random trees using a potential function planner. In *Proc. and 2005 European Control Conference Decision and Control CDC-ECC '05. 44th IEEE Conference on*, pages 7965–7970, 12–15 Dec. 2005. 18, 29

[14] I. Garcia and J. P. How. Trajectory optimization for satellite reconfiguration maneuvers with position and attitude constraints. In *Proc. American Control Conference the 2005*, pages 889–894, 8–10 June 2005. 18, 29

[15] P. E. Gill, W. Murray, and M. A. Saunders. *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*. 25

[16] GLOBE Task Team, D. A. Hastings, P. K. Dunbar, G. M. Elphingstone, M. Bootz, H. Murakami, H. Masaharu, P. Holland, J. Payne, N. A. Bryant, T. L. Logan, J.-P. Muller, G. Schreier, and J. S. MacDonald. *The Global Land One-kilometer Base Elevation (GLOBE) Digital Elevation Model, Version 1.0*. National Oceanic and Atmospheric Administration, National Geophysical Data Center, 325 Broadway, Boulder, Colorado 80305-3328, U.S.A. http://www.ngdc.noaa.gov/mgg/topo/globe.html, 1999. 68

[17] A. Holmberg and P.-M. Olsson. Route planning for relay UAV. In *Proceedings of the 26th International Congress of the Aeronautical Sciences*, 2008. 21

[18] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. *IEEE Control Systems Magazine*, Vol. 28(No. 2):pp. 51–64, April 2008. 63, 64

[19] G. T. Huntington. *Advancement and Analysis of a Gauss Pseudospectral Transcription for Optimal Control*. PhD thesis, Massachusetts Institute of Technology, May 2007. 21, 25

[20] G. T. Huntington, D. A. Benson, , and A. V. Rao. Design of optimal tetrahedral spacecraft formations. *Journal of the Astronautical Sciences*, Vol. 55(No. 2):pp. 141–169, April-June 2007. 25

[21] G. T. Huntington, D. A. Benson, J. P. How, N. Kanizay, C. L. Darby, and A. V. Rao. Computation of boundary controls using a gauss pseudospectral method. In *2007 Astrodynamics Specialist Conference*, Mackinac Island, Michigan, August 2007. 25

[22] G. T. Huntington and A. V. Rao. Optimal reconfiguration of spacecraft formations using a gauss pseudospectral method. *Journal of Guidance, Control, and Dynamics*, Vol. 31(No. 3):pp. 689–698, May-June 2008. 18, 25

[23] A.S. Ibrahim, K.G. Seddik, and K.J.R. Liu. Improving connectivity via relays deployment in wireless sensor networks. In *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 1159–1163, Nov. 2007. 21

[24] A.S. Ibrahim, K.G. Seddik, and K.J.R. Liu. Connectivity-aware network maintenance via relays deployment. In *Proc. IEEE Wireless Communications and Networking Conference WCNC 2008*, pages 2573–2578, March 31 2008–April 3 2008. 21

[25] J.J. Kuffner Jr. and S.M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE International Conference on Robotics and Automation ICRA '00*, volume 2, pages 995–1001, 24–28 April 2000. 29

[26] Y. Kuwata. *Trajectory Planning for Unmanned Vehicles using Robust Receding Horizon Control*. PhD thesis, Massachusetts Institute of Technology, February 2007. 21, 53, 59, 66

[27] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How. Motion planning for urban driving using RRT. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages pp. 1681–1686, Nice, France, September 2008. 29

[28] S. M. LaValle. Rapidly-Exploring Random Trees: A new tool for path planning. Technical Report TR 98-11, Computer Science Dept., Iowa State University, October 1998. 18, 29, 30

[29] Xiangheng Liu, A. Goldsmith, S.S. Mahal, and J.K. Hedrick. Effects of communication delay on string stability in vehicle platoons. In *Proc. IEEE Intelligent Transportation Systems*, pages 625–630, 25–29 Aug. 2001. 19

[30] The Mathworks. MATLAB. Available at http://www.mathworks.com/, May 2009. 65

[31] H. G. Nguyen and J. P. Bott. Robotics for law enforcement: Applications beyond explosive ordnance disposal. In *SPIE Proc. 4232: Technologies for Law Enforcement*, Boston, MA, November 2000. 16

[32] H. G. Nguyen, N. Pezeshkian, M. Raymond, A. Gupta, and J. M. Spector. Autonomous communication relays for tactical robots. In *in Proceedings of the International Conference on Advanced Robotics (ICAR)*, 2003. 20, 74

[33] Office of the Secretary of Defense. Unmanned aircraft systems roadmap. Technical report, United States Department of Defense, 2005. 16

[34] Staff Sgt. R. Piper. Small UAV provides eyes in the sky for battalions. *Military.com*, 2005. 15

[35] A. V. Rao, D. A. Benson, G. T. Huntington, and C. Francolin. *Users Manual for GPOPS Version 1.3: A MATLAB Package for Dynamic Optimization Using the Gauss Pseudospectral Method*. 25

[36] T. Schouwenaars. *Safe Trajectory Planning of Autonomous Vehicles*. PhD thesis, Massachusetts Institute of Technology, February 2006. 20

[37] T. Schouwenaars, E. Feron, and J. P. How. Multi-vehicle path planning for non-line of sight communication. In *Proc. American Control Conference*, 2006. 20

[38] A. Srinivas. *Mobile Backbone architecture for wireless ad-hoc networks: algorithms and performance analysis*. PhD thesis, Massachusetts Institute of Technology, 2007. 20

[39] Northrop Grumman Integrated Systems. RQ-4 block 20 Global Hawk. Available at http://www.is.northropgrumman.com/systems/ghrq4b.html, May 2009. 15

[40] M. Valenti, B. Bethke, G. Fiore, J. P. How, and E Feron. Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Keystone, CO, August 2006. 63, 64

[41] Vicon. Vicon MX systems. Available at http://www.vicon.com/products/viconmx.html, July 2006. 63