Obstacle Avoidance, Visual Automatic Target Tracking, and Task Allocation for Small Unmanned Air Vehicles

Jeffery Brian Saunders
Brigham Young University - Provo

OBSTACLE AVOIDANCE, VISUAL AUTOMATIC TARGET TRACKING, AND TASK ALLOCATION FOR SMALL UNMANNED AIR VEHICLES

by

Jeffery B. Saunders

A dissertation submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Electrical and Computer Engineering

Brigham Young University

August 2009

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a dissertation submitted by

Jeffery B. Saunders

This dissertation has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Randal W. Beard, Chair
Date	Clark N. Taylor
Date	D.J. Lee
Date	Timothy W. McLain
Date	James K. Archibald

ABSTRACT

OBSTACLE AVOIDANCE, VISUAL AUTOMATIC TARGET TRACKING, AND TASK ALLOCATION FOR SMALL UNMANNED AIR VEHICLES

Jeffery B. Saunders

Department of Electrical and Computer Engineering

Doctor of Philosophy

Recent developments in autopilot technology have increased the potential use of micro air vehicles (MAVs). As the number of applications increase, the demand on MAVs to operate autonomously in any scenario increases. Currently, MAVs cannot reliably fly in cluttered environments because of the difficulty to detect and avoid obstacles. The main contribution of this research is to offer obstacle detection and avoidance strategies using laser rangers and cameras coupled with computer vision processing. In addition, we explore methods of visual target tracking and task allocation.

Utilizing a laser ranger, we develop a dynamic geometric guidance strategy to generate paths around detected obstacles. The strategy overrides a waypoint planner in the presence of pop-up-obstacles. We develop a second guidance strategy that oscillates the MAV around the waypoint path and guarantees obstacle detection and

avoidance. Both rely on a laser ranger for obstacles detection and are demonstrated in simulation and in flight tests.

Utilizing EO/IR cameras, we develop two guidance strategies based on movement of obstacles in the camera field-of-view to maneuver the MAV around pop-up-obstacles. Vision processing available on a ground station provides range and bearing to nearby obstacles. The first guidance law is derived for single obstacle avoidance and pushes the obstacle to the edge of the camera field-of-view causing the vehicle to avoid a collision. The second guidance law is derived for two obstacles and balances the obstacles on opposite edges of the camera field-of-view, maneuvering between the obstacles. The guidance strategies are demonstrated in simulation and flight tests.

This research also addresses the problem of tracking a ground based target with a fixed camera pointing out the wing of a MAV that is subjected to constant wind. Rather than planning explicit trajectories for the vehicle, a visual feedback guidance strategy is developed that maintains the target in the field-of-view of the camera. We show that under ideal conditions, the resulting flight paths are optimal elliptical trajectories if the target is forced to the center of the image plane. Using simulation and flight tests, the resulting algorithm is shown to be robust with respect to gusts and vehicle modeling errors.

Lastly, we develop a method of a priori collision avoidance in assigning multiple tasks to cooperative unmanned air vehicles (UAV). The problem is posed as a combinatorial optimization problem. A branch and bound tree search algorithm is implemented to find a feasible solution using a cost function integrating distance traveled and proximity to other UAVs. The results demonstrate that the resulting path is near optimal with respect to distance traveled and includes a significant increase in expected proximity distance to other UAVs. The algorithm runs in less than a tenth of a second allowing on-the-fly replanning.

ACKNOWLEDGMENTS

I want to express my gratitude to Professor Randy Beard for his help and guidance during the research and writing of this work. I would also like to thank my graduate committee for their time and effort in reading and editing my research. In addition, I would like to thank my coworkers in the MAGICC lab, including Derek Kingston, Andrew Eldredge, Derek Nelson, Stephen Griffiths, David Johansen, Bryce Ready, Travis Millet, and Benjamin Heiner. Thank you to my parents Bruce and Laura Saunders for their support in this.

This research was partially supported by the Air Force Research Laboratory, Munitions Directorate under SBIR contract No. FA 8651-07-C-0094 to Scientific Systems Company, Inc. and Brigham Young University.

Table of Contents

List of Figures

Chapter 1

Introduction

Fixed-wing micro air vehicles (MAVs) are increasingly common in military and civilian use and their cost continues to decrease even as their functionality increases. During the past ten years, autopilots for MAVs have become possible due to the decreasing size of the necessary sensors as well as the decrease in size and power requirements of integrated circuits. The size and low power requirements of these aircraft offer the potential of autonomous surveillance [2], terrain mapping [3], search missions [4], and mapping [5, 6] to name a few. The low detection probability, high functionality, and low cost create military and civilian uses not previously possible. However, small aircraft have significant weight limitations that prevent many sensors, such as radar and ladar, to be carried onboard. Sensors commonly used on commercial aircraft to detect obstacles and terrain are not feasible on MAVs. Since MAVs fly at low altitudes, they require sensors to detect obstacles and terrain, and to avoid collisions. The sensors need to be lightweight with low power requirements. This research explores two lightweight devices to detect obstacles, laser rangers and EO/IR cameras, and contributes to the current body of literature by developing methods of reactive obstacle avoidance using laser rangers, and vision-based reactive obstacle avoidance. In addition it also contributes by exploring vision-based target tracking, and task allocation for teams of UAVs.

Reactive obstacle avoidance is defined here as obstacle avoidance in the zero to three second time frame. It is commonly used in conjunction with a high level path planner, such as Rapidly-Expanding Random Trees (RRT) [7, 8]. Assuming an a priori terrain map is available, the path planner creates a path from an initial configuration to a final configuration, incorporating any nonholonomic constraints of the vehicle and any known obstacles. The goal of reactive obstacle avoidance is to avoid any pop-up obstacles on the path. Pop-up obstacles are obstacles that are detected but not included on the terrain map. The reactive planner avoids obstacles while the waypoint path planner creates a new path around pop-up obstacles. Chapter 2 develops obstacle avoidance using laser rangers and Chapter 3 develops vision-based reactive avoidance techniques.

Laser rangers are non-scanning devices for range estimation. While the lack of scanning capabilities on a MAV makes collision avoidance difficult to guarantee, they are lightweight and feasible for use on MAVs. We develop two reactive guidance strategies based on a laser ranger. The first generates waypoints around detected obstacles. There is no guarantee of obstacle detection or avoidance. However, the method requires little processing power and it is implemented on the autopilot. The second guidance strategy oscillates the MAV around a waypoint path, essentially scanning for obstacles. It guarantees obstacle detection and avoidance at the expense of the oscillation.

As an alternative to laser rangers, cameras with vision processing offer range-to-target and bearing-to-target estimation to obstacles in the field-of-view of the camera [9, 10]. Cameras are lightweight and low power, making them feasible for MAVs. Vision processing is available on a ground station if it is not available onboard. We develop two guidance strategies based on range-to-obstacle and bearing-to-obstacle estimates. Both strategies are based on movement of the obstacle in the camera field-of-view. The first is derived for single obstacles. The guidance law pushes the obstacle to the edge of the camera field of view. By maintaining the obstacle at the edge of the camera field-of-view, we show that the MAV maneuvers around the obstacle. After the MAV passes the obstacle, it continues on the original waypoint path. The second guidance strategy is derived for two obstacles. It uses movement of the obstacles in the camera field-of-view to balance the obstacles on opposite edges of the field-of-view, essentially flying between the obstacles. Since more than two obstacles may be encountered, we present two algorithms to decide which two obstacles to maneuver between. The first algorithm operates in 2D at a constant altitude, commanding roll angles to avoid obstacles. The second operates in 3D, commanding pitch and roll angles to avoid obstacles. Both single and multiple obstacle guidance strategies are demonstrated in simulation and flight tests.

In addition to obstacle avoidance, target tracking is a desirable feature for MAVs. The low audio signal of a MAV in flight allows it to observe targets without detection. Target tracking is defined here as maintaining a target in the field-of-view of an onboard camera with sufficient resolution for visual detection. Current research in target tracking uses gimballed cameras to observe targets [11–16] or variable-azimuth cameras [13, 14, 17]. Gimbals are not always feasible on MAVs, due to weight limitations. We explore a feedback loop guidance strategy based on movement in the camera field-of-view to maneuver the MAV such that the target does not leave the camera field-of-view. The contribution of Chapter 4 is a guidance strategy using a strapdown camera in a constant wind. It is demonstrated in simulation and flight tests.

The final chapter of this research addresses task allocation for multiple vehicles. In the COUNTER scenario, a team of MAVs are given a set of targets to observe, using onboard sensors. The allocation algorithm must assign the MAVs to observe targets in a sequential order such that fuel consumption is minimized, arrival to targets is staggered, and collision is avoided. The problem is posed as a combinatorial optimization problem. The optimal solution requires extensive computation. We show that by using a branch and bound search, a near optimal solution is possible in less than a second, allowing on-the-fly replanning.

The research contributions of this work are as follows.

- developed a 2D obstacle avoidance strategy to generate waypoint paths around obstacles in real time,
- developed a 2D obstacle detection and avoidance strategy using a laser ranger and circular paths that guarantees obstacle detection and avoidance,
- developed a 2D visual obstacle detection and avoidance strategy for single obstacles,

- developed a 2D and 3D visual obstacle detection and avoidance strategy for multiple obstacles,
- derived a visual target tracking strategy for strap-down cameras,
- formulated a task allocation algorithm for teams of MAVs incorporating distance traveled, collision, and staggered arrival to task, and
- described a simulation environment designed for fast research and prototyping in an educational environment.

The organization of this research is as follows. Chapter 2 discusses reactive obstacle avoidance using a laser ranger and Chapter 3 derives reactive obstacle avoidance using a camera. Target tracking is discussed in Chapter 4. The task allocation algorithm is addressed in Chapter 5. A conclusion and future directions of research are discussed in Chapter 6. Additionally, the simulation environment for this research and its advantages for development are introduced in Appendix A.

Chapter 2

Obstacle Avoidance Using a Laser Range Finder

2.1 Reactive Avoidance Using Waypoint Path Generation

2.1.1 Introduction

An important problem in the design of fixed-wing miniature air vehicles (MAVs) is obstacle avoidance. There are two general classes of obstacles: a priori known obstacles (e.g. from a terrain map) for which a trajectory can be generated before flight, and pop-up obstacles encountered during flight. The objective of this Chapter is to present reliable computer algorithms that can avoid pop-up obstacles. We assume a waypoint path planner creates paths around pop-up obstacles while the reactive planner avoids collision in the zero to three second time frame. In this Chapter we focus on fixed wing miniature air vehicles (MAV) with wingspans less than 48 inches.

Many methods of path planning and obstacle avoidance have been developed. Potential fields is an easy to implement and popular method that creates a trajectory by using attractive (toward the goal) and repulsive (away from obstacles) forces. The result is a path that moves away from obstacles while still moving toward a goal. The problem arises with the existence of local minima away from which a path cannot be generated. Variants have been developed which attempt to remove local minima such as an escape force [18] or simulated-annealing [19]. However, these still are not guaranteed to find a path to the goal given the existence of such a path.

Probability Road Map (PRM) planners create a path by connecting random configurations in the configuration space into a feasible path to a goal [20]. PRM planners consist of two phases. The first is a learning phase. The planner randomly generates points in the configuration space until a desired number of points is reached.

Phase two connects the points in such a way that the cost of the path is minimized. The algorithm allows freedom on how that cost is determined. PRM planners can easily be used in holonomic scenarios where movement is not constrained. For non-holonomic scenarios however, PRM planners may not be sufficient. The constraints may be such that connection of feasible random points in the configuration space occurs too rarely. PRM planners also have problems with respect to completeness and computation time. There is no guarantee of completeness and no limit on computation time for an optimal path. Some of these problems have been partially overcome in subsequent work [21–25].

Rapidly-exploring Random Trees [26] [27] (RRT) is a good alternative to PRM planners for non-holonomic systems. RRT builds a tree starting from an initial configuration. Each branch on the tree takes into account the configuration from which it is starting, and randomly generates a new feasible configuration. The result is a tree that is completely within the constraints of the system. A path can then be found by finding the lowest cost path through the tree. Recent work has added a bridge technique to find paths through narrow spaces for PRM and RRT [28]. This significantly reduces computation time for areas with narrow passageways. However, RRT planners still suffer from lack of completeness and no limit to computation time to find an optimal path.

Cell decomposition [29] divides the configuration space into cells which can be labeled as obstacle-free or with-obstacle. A collision free path can be found by connecting the cells from the initial configuration to the final configuration. Region division into cells becomes a problem as obstacles become less trivial. It is desirable to have the smallest number of cells possible while still avoiding obstacles. A real-time algorithm using cell decomposition was developed in [30]. Cell decomposition has recently been combined with probabilistic planning to produce probabilistic cell decomposition [31] (PCD). The algorithm uses cells to direct the probabilistic planner to more "interesting" areas. The result is a feasible path that requires less computation time.

In nearly all previous work, perfect knowledge of surrounding obstacles and terrain is assumed. In MAVs, hardware to detect obstacles to such accuracy is not feasible. Real situations must allow for error and lack of information of the dimensions of an obstacle.

To plan around obstacles they must first be detected, and detection is a problem for small aircraft. Most sensors capable of reliable obstacle detection in large areas, such as radar or laser scanners, are too heavy for small aircraft. One possible lightweight sensor is a camera. Work has been done on using cameras and computer vision to detect obstacles. Cameras are lightweight and possible to mount on a MAV. In a paper by Sharp et al. [32], vision is used to land a helicopter, i.e. ground detection. A 3D model is created from computer vision in papers by Kanade et al. [33] and Sinopoli et al. [34] which is used for navigation and obstacle avoidance. Computer vision can be used for obstacle avoidance using state of the art technology and algorithms. However, computer vision requires a large amount of computation which usually requires an off-board computer. We would like all of the computation to be self-contained onboard the MAV.

In this Chapter we propose a solution that uses a lightweight, low-cost laser ranger and a computationally efficient algorithm to detect and avoid obstacles. Our approach uses a priori known terrain maps to generate an initial waypoint path through the known obstacles, and a reactive planner that responds to pop-up obstacles detected by the laser ranger. An a priori planner plans waypoint paths by exploring a terrain map using a Rapidly-Exploring Random Tree (RRT) [35]. The MAV tracks the planned trajectory using standard trajectory tracking techniques. The results of this chapter have been published in [36, 37].

The reactive planner is used to dynamically avoid obstacles. We assume that a simple single point laser ranger that operates as slow as two Hertz is mounted parallel to the longitudinal axis of the MAV. Using only the returned distances, the MAV can avoid detected obstacles in its path while minimizing the deviation from the planned trajectory. The algorithm operates in real-time and is demonstrated in flight tests.

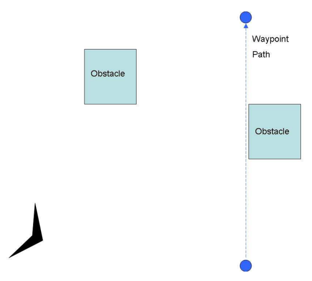
2.1.2 Reactive Path Planner

The reactive path planner consists of two parts. The first part is the waypoint path tracker. It is used by the obstacle avoidance algorithm and the a priori path planner to track waypoint paths and uses the method described in [38]. The second part of the reactive planner is the obstacle avoidance path planner which considers all of the detected objects and plans waypoint paths around them.

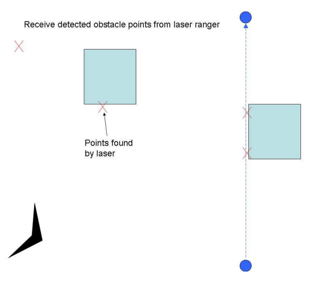
We make a few assumptions about the environment. First we assume level flight and no wind. This implies that the optical axis of the laser ranger is the flight path of the MAV and that the laser will detect obstacles at the current altitude of the MAV. We assume that the laser ranger runs at 2-3 Hertz. This allows it to run at low power and be small enough to mount on a MAV. We also assume that the laser has detected enough of the obstacle to successfully avoid it. Our last assumption is the existence of available memory to store laser ranger data.

Consider the pop-up threat scenario shown in Figure 2.1. Figure 2.1(a) shows the starting position of the MAV as it enters a situation where obstacle avoidance is required. We assume the MAV has a forward ground velocity and tracks the given waypoint path. We also assume the laser ranger has detected enough points on objects to safely avoid the obstacle. The MAV detects the obstacles in Figure 2.1(b). Notice that not all of the points are in front of the MAV. These points represent past obstacles that the laser ranger detected but are no longer relevant. These are removed as in Figure 2.1(c). To avoid the obstacles, we propose generating avoidance triangles around the obstacles and using the sides of the triangles as possible waypoint paths. Since the dimensions of the obstacles are unknown, they are estimated by construction of a triangle around the scanned points, as shown in Figure 2.1(d). Notice that some of the triangle sides intersect. If sides intersect, then they are not feasible paths around the obstacles. Figure 2.1(e) removes the intersecting sides. The remaining sides are all possible waypoint paths around the detected obstacles.

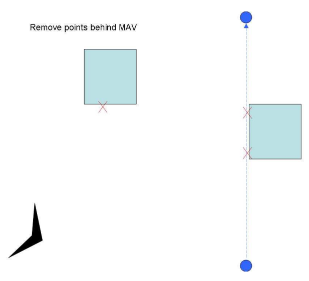
The MAV cannot always track each path generated before hitting an obstacle. In addition, some paths can be tracked faster than others. Using an estimation method described later, we can estimate at what point $\mathbf{r}$ the MAV can track each path. Using **r**, we can determine whether the path can be tracked before hitting the obstacle. Infeasible paths can be removed as in Figure 2.1(f). The remaining paths can be assigned a cost using the distance from the MAV to **r**, and a path selected based on this cost. The final selected path is shown in Figure 2.1(h)



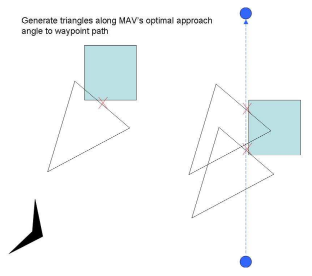
(a) MAV and waypoint path

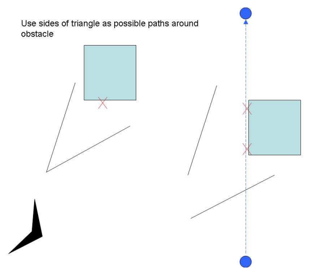


(b) Detected Obstacles

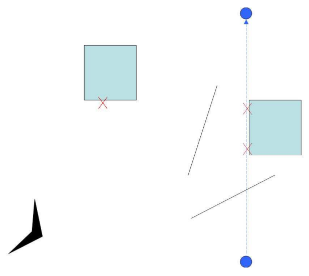


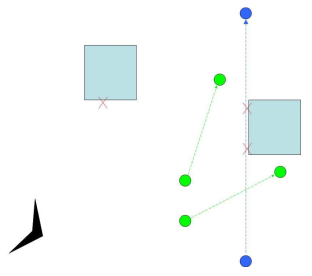
(c) Remove obstacles behind MAV

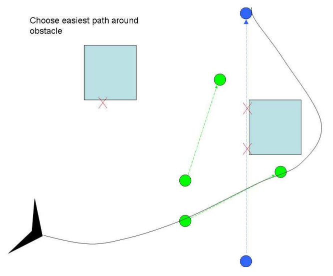

Generate triangles along MAV's optimal approach angle to waypoint path
(d) Generated triangles



(e) Possible waypoint paths


(f) Remove infeasible paths


(g) Two possible paths



(h) Choose path with smallest cost

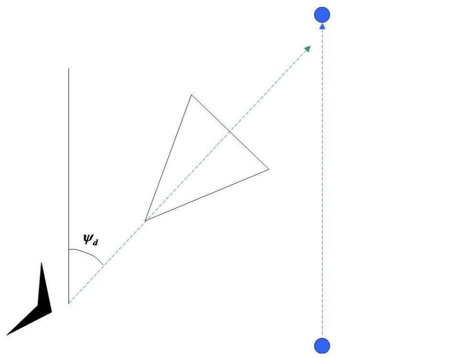
Figure 2.1: Reactive planner example. The MAV must detect and avoid obstacles while approaching a pre-planned waypoint path.

The steps to avoiding obstacles by generating intermediate waypoint paths make use of the waypoint trajectory tracking method described in Ref. [38]. The first step is to calculate the desired approach angle (ψ_d) to the current waypoint path. Using ψ_d, the algorithm can consider the direction which progresses the MAV toward the waypoint path. The second step removes all points behind the MAV and farther than $z_l + 2R$ from the MAV, where z_l is the height of the triangles generated around obstacles and R is the minimum turning radius of the MAV. We remove points over a certain distance from the UAV to prevent unneeded processing in determining the best path around obstacles. The third step is triangle generation. Given that the MAV cannot detect the exact size of the obstacle, the algorithm makes a buffer region around the point using a triangle. The triangle is isosceles and the angle bisector of the triangle has the same heading as ψ_d as shown in Figure 2.2(a). Make a triangle for each point. The result will be multiple triangles that may overlap, but are all of equal size as in Figure 2.2(b). Notice that z_w and z_l are two important dimensions of the triangles (Figure 2.2(c)). z_l is the distance the MAV has to maneuver around the obstacle. If the MAV has a large turning radius, then z_l will have to be large as well. z_w is the buffer region between the obstacle point and the new waypoint path. As expected obstacle size grows, so will z_w. There is further analysis on this in Section 2.1.3.

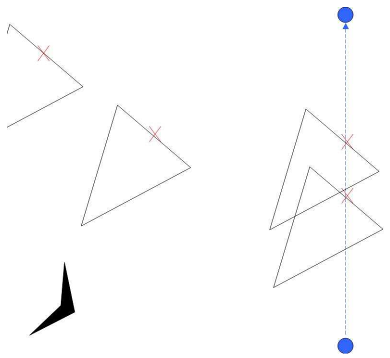
The next step is to remove intersecting sides. There will be at least two remaining sides for a pair of triangles after the intersection elimination since each equivalent side is parallel. No matter how the triangles are arranged, at least a set of non-parallel sides will remain. By our assumptions we know all the points around an obstacle, therefore those paths also avoid obstacles.

There are two or more remaining triangle sides that can be used as waypoint paths around the obstacle. The remaining task is to choose the best path. Each path is within $z_l + 2R$ to the MAV and deviates equally from ψ_d. Unfortunately, there will still be some paths that cannot be tracked before colliding with an obstacle. We need a cost function that estimates how fast the MAV can track each path. A good cost function is simply the distance of the MAV to the first point on the waypoint path.

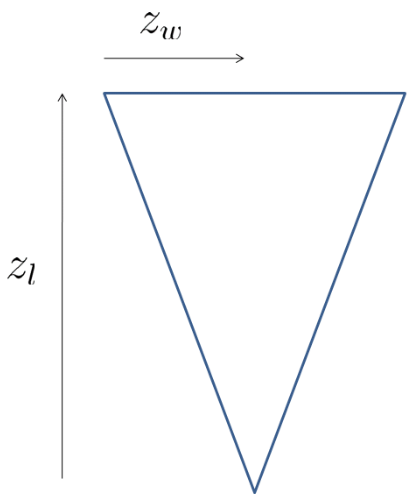
There are two cases to consider to estimate the point at which the MAV can track the waypoint path. The first case is when the waypoint path is very close, within a turning radius R of the MAV. The second case is when the path is further away



(a) Approach Angle to waypoint path (ψ_d).



(b) Generate triangles around points.



(c) Triangle dimensions.

Figure 2.2: Triangle generation algorithm. Triangle sides are used as possible waypoint paths around obstacles.

than R. To determine which case to use, first determine whether the MAV needs to turn left or right to track the path. This can be done by calculating the cross product of the velocity vector $\mathbf{v}$ of the MAV by the waypoint being approached minus the MAV position vector $\mathbf{u}$ (e.g. $\mathbf{v} \times (\mathbf{w_2} - \mathbf{u})$). If the cross product is positive, the MAV must turn left, otherwise right. Now create a circle to the left or right of the MAV as needed using the turning radius. The circle should be positioned to represent the turn of the MAV. Translate the circle and waypoint path such that the center of the circle is at the origin. This simplifies the calculations. We must find where the circle intersects the waypoint path. First take the equation of the waypoint path:

$$y_i = m(x_i - u_x) + u_y, \tag{2.1}$$

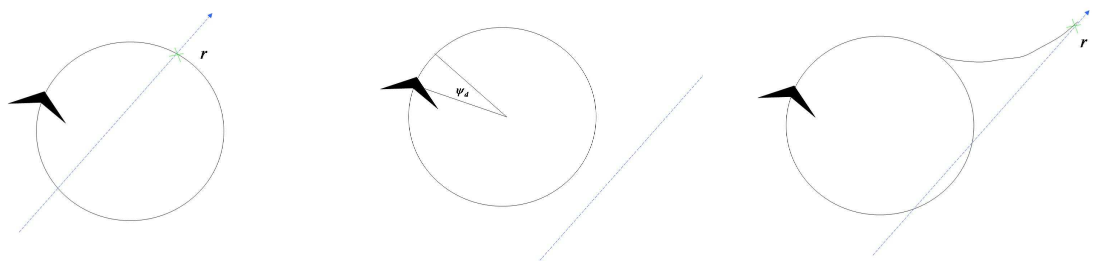
where x_i and y_i are the intersection points of the waypoint path and circle, m is the slope, and u_x and u_y are the position coordinates of the MAV. The equation of the circle is:

$$R^2 = x_i^2 + y_i^2. \tag{2.2}$$

Substituting and solving for x_i we get

$$x_i = \frac{1}{2(1+m^2)}(2m^2 w_{2x} - 2mw_{2y} + 2\sqrt{-m^2 w_{2x}^2 + 2mw_{2x}w_{2y} - w_{2y}^2 + R + m^2 R}). \tag{2.3}$$

y_i can be obtained by substituting x_i back into Equation (2.1). In the case that the square root is negative, the circle and waypoint path do not intersect and we know that we must use case two. If it is not negative, then we must find where x_i and y_i are on the circle. Translate the waypoint path such that it crosses through the center of the circle. Take the cross product of the waypoint path and x_i and y_i. If the cross product is negative, use case two. Otherwise use case one. Case one consists of using x_i and y_i as the coordinate **r** which indicates the position at which the MAV will track the waypoint path as indicated in Figure 2.3(a).



(a) Case 1. (b) ψ_d. (c) Case 2.

Figure 2.3: Waypoint path interception.

Case two projects the MAV position onto the waypoint path and adds an estimated distance. The estimated distance is calculated by finding ψ_d, the angle of the circle across the distance needed to fly before flying parallel to the waypoint path (Figure 2.3(b)). That distance plus another two turning radii along the waypoint path from the projection are the estimated distance to $\mathbf{r}$, the point of tracking the waypoint path. This is illustrated in Figure 2.3(c).

The cost function is the distance from the MAV to $\mathbf{r}$. The smallest cost is the path that can be tracked in the shortest distance. Choose the path with the smallest cost as the new waypoint path.

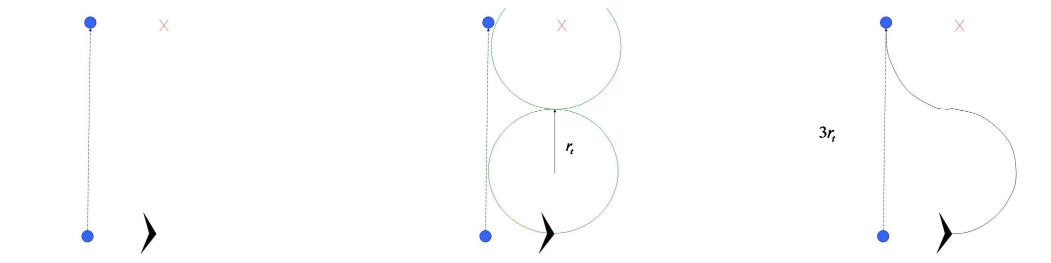
The reactive path planner is summarized by the following steps,

1. calculate the desired heading angle to the waypoint path ψ_d,
2. remove all points behind the MAV and all points over $z_l + 2R$ distance away from the MAV,
3. generate triangles with dimensions z_w and z_l along the angle ψ_d,
4. remove intersecting sides of triangles. The remaining sides are possible waypoint paths,
5. calculate the estimated point of tracking for each of the waypoint paths ($\mathbf{r}$). The distance to this point is the cost for the corresponding waypoint path, and
6. choose the path with the smallest cost as the new waypoint path.

2.1.3 Analysis

This section presents conditions under which the UAV is guaranteed to track a generated path from the reactive planner. The constraints of a fixed wing air vehicle and limited processing efficiency severely limit finding feasible paths for all configurations, but under certain conditions, successful avoidance can be guaranteed.

Consider Figure 2.4(a). The red X represents a detected obstacle and the blue dashed line is the desired waypoint path to track. The MAV is angled such that the obstacle is slightly in front, thus not removing from consideration. To avoid the obstacle, the MAV must turn nearly 180 degrees toward the waypoint path and then turn back 90 degrees to begin tracking the path. This can be considered a worst case scenario. Any other scenario requires less distance to track the path.



(a) Worse case scenario (b) Turning Radius (c) Quickest path

Figure 2.4: Analysis of worst case scenario to show sufficient conditions to guarantee obstacle avoidance.

Theorem 2.1.1 *Given a MAV with turning radius R, parameter $z_l \geq 3R$, and assuming (1) no wind, (2) sufficient points have been detected on a given obstacle, (3) that the MAV is z_l distance away from the obstacle, and (4) the MAV can track the partial orbital paths to the waypoint path, then the MAV can successfully track the waypoint path generated around that obstacle.*

Proof: Consider the situation where the MAV is approaching a set of obstacles that requires it to track a path requiring at least a 90 degree heading change as depicted in Figure 2.4(b). To track the path, the MAV must turn 180 degrees toward the path, and then turn another 90 degrees to track the path, as shown in Figure 2.4(c). The MAV may cover up to $3R$ distance before tracking the path around the obstacle. The parameter z_l is the distance from an obstacle where a path begins. Therefore, if the UAV is at least $3R$ away from the obstacle, the MAV is guaranteed to avoid it. ■

2.1.4 Simulations Results

Virtual World

The goal of this Section is to describe the simulation environment. The simulator is the primary testing method for our architecture making it an important aspect in understanding the results. The simulator generates random cities through which the MAV can be flown. The parameters that can be changed are street width, building width, mean building height, and building height variance. The buildings created are placed on a flat terrain and can be detected by a simulated laser ranger. The buildings are rectangular and protrude vertically from the flat terrain surface.

The physics engine of the simulator models aerodynamic forces, gravitational forces, forces due to control surfaces, forces due to thrust, and forces due to wind. The MAV aerodynamic coefficients have been tuned to match the airframes used in our hardware testbed.

The simulator also emulates the Kestrel autopilot used at BYU [39]. It receives a .dll file of compiled Kestrel autopilot code and runs in real time in to match the speed and timing of the MAV autopilot. The controls and sensors are run by the autopilot in the simulator in the sense that they are executed in hardware as algorithms. This method makes the simulation a close match to actual flight tests facilitating rapid prototyping.

Reactive Planner

Avoiding a single obstacle allows the algorithm parameters like z_w and z_l to be appropriately tuned. The parameters can be adjusted based on convergence speed to the desired waypoint, desired distance from obstacle during avoidance, and desired distance from obstacle to begin avoidance. In the simulation depicted in Figure 3.15, the MAV was directed to fly on a waypoint path directly through a building. The building is represented as a solid gray box in the image with a width of 20 meters. The solid white line is the waypoint path. The dotted line is the actual path flown by the MAV. The parameters of the algorithm are:

$$R = 30,$$
$$z_l = 3R,$$
$$z_w = 35.$$

where R is the minimum turning radius of the MAV. The MAV used the virtual laser range finder running at three Hertz to detect the position of the obstacle and effectively avoided it.

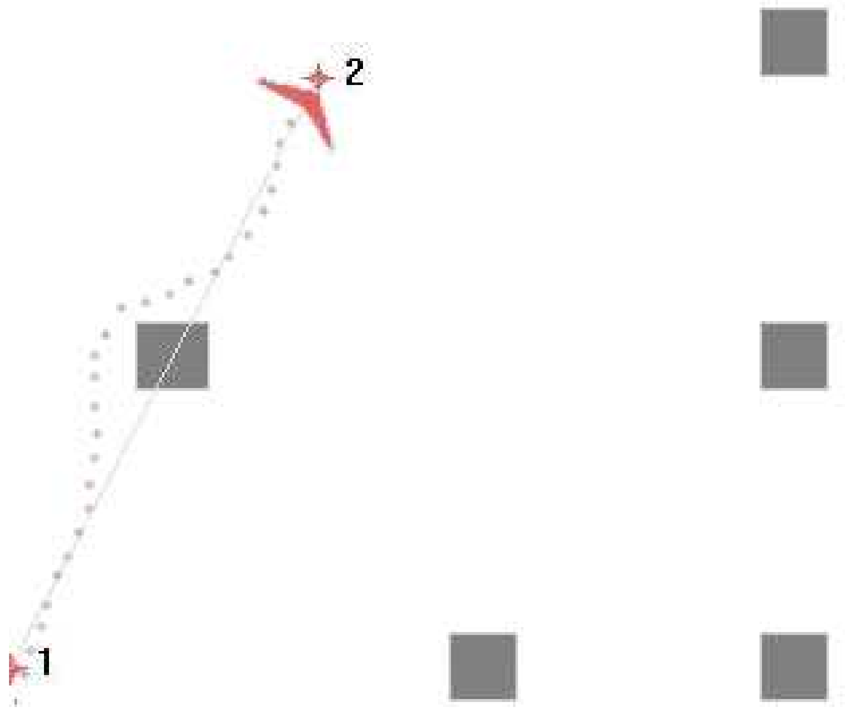


Figure 2.5: Single obstacle simulation

2.1.5 Flight Test Results

A successful test of the reactive planner required at least one obstacle high enough at which to fly the MAV without a need to change altitude low to the ground. This prevents the laser ranger from detecting points on the ground that might be mistakenly considered obstacles and allows for short altitude drops during turns. We used the laser ranger in Figure 2.6 from Optics Planet [1] with dimensions approximately 3 inches by 4 inches.

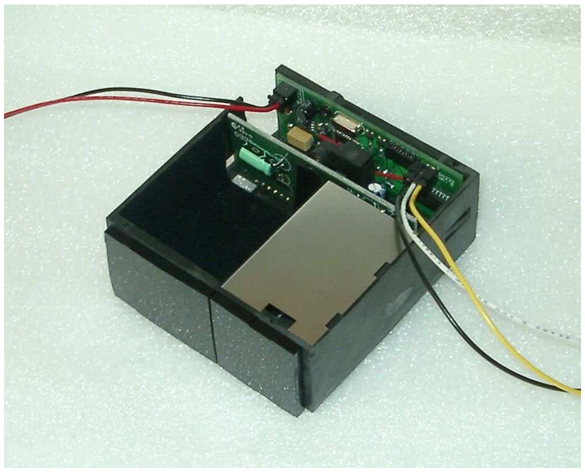


Figure 2.6: We used the laser ranger available from Optics Planet [1] to detect obstacles.

We chose an isolated building on the BYU campus that is 50 meters high and 35 meters square. Other surrounding buildings rose to about 20 meters which separated the building as a single obstacle. We planned a path from the south side of the building to the north at an altitude of 40 meters. The MAV had no previous information on the location and dimensions of the building. The turning radius of the MAV is approximately $R = 30$ meters. We set $z_l = 90$ meters as suggested by Section 2.1.3 and $z_w = 55$ meters to ensure that one detected point on the building generated a path sufficiently close to the side of the building. A GPS telemetry plot of the results is shown in Figure 2.7.

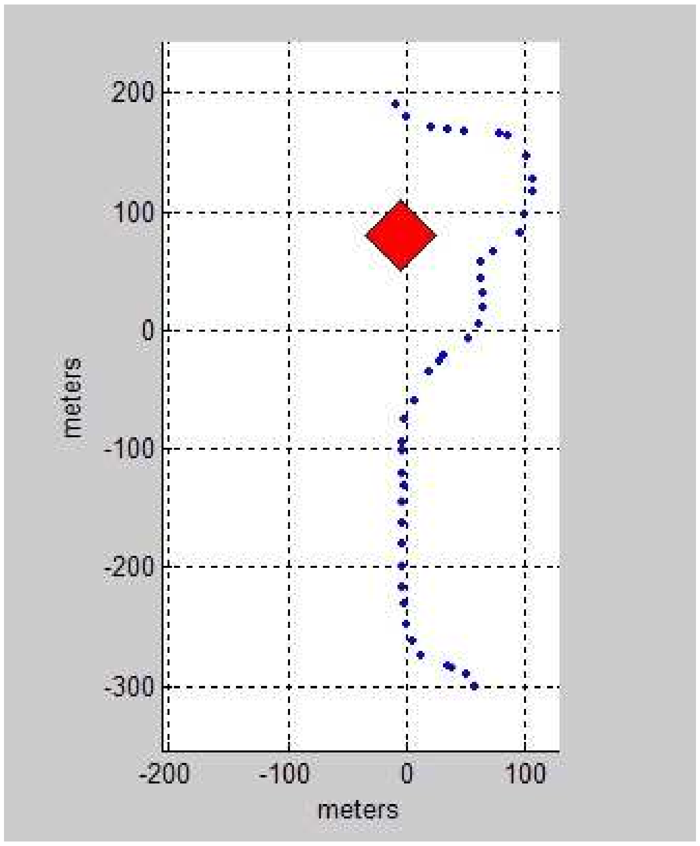


Figure 2.7: Flight test results plot of a MAV avoiding a building with a planned path through the building.

As the MAV approached the building, the laser ranger detected the building and calculated its position. When the MAV came within 90 meters of the building, the reactive planner generated a path around the building and the MAV began to track the path. Notice that as the MAV passed the building, it once again began to track the original waypoint path that went through the building. The MAV successfully avoided the building without human intervention.

2.2 Circular Paths

Our goal in this section is to incorporate a laser ranger sensor into an obstacle avoidance algorithm such that obstacle detection and obstacle avoidance are guaranteed. We assume each MAV is equipped with a laser ranger in the longitudinal direction of the body frame. A single laser ranger is light enough to mount on a MAV and allows ranging for obstacles directly in front of the MAV. We will derive a series of circular paths around a waypoint path such that the MAV may scan obstacles near the waypoint path and possible escape routes to avoid detected obstacles. Our goal is to guarantee obstacle detection and extend the derived methods of obstacle avoidance to real time and account for frequent changes in the environment.

2.2.1 Problem Description

We want to generate an oscillating path along which a MAV may traverse to scan potential obstacles. The desired path must conform to the non-holonomic constraints of a fixed wing aircraft and enable the MAV to scan possible new paths ahead. Consider the first order set of kinematic equations of a fixed wing MAV,

$$
\begin{aligned}
\dot{r}_n &= V \cos \psi, && (2.4)\\
\dot{r}_e &= V \sin \psi, && (2.5)\\
\dot{\psi} &= \frac{g}{V} \tan \phi, && (2.6)\\
\dot{V} &= \alpha_V (V^c - V), && (2.7)\\
\dot{\phi} &= \alpha_\phi (\phi^c - \phi), && (2.8)
\end{aligned}
$$

where $(r_n, r_e)^T$ is the position of the aircraft, V, V_c, ϕ, and ψ are the velocity, commanded velocity, roll, and heading, g is the gravitational constant, and α_ϕ and α_V are parameters. Using these equations, a constant roll angle ϕ_d results in a constant heading change $\dot{\psi}$. The aircraft flies a circular trajectory with a turning radius R. With a mounted laser ranger, the MAV scans nearby regions outside the circular path.

We would like to use the scanning abilities of the circular path by incorporating a series of circular oscillations in waypoint paths such that the MAV scans obstacles on or near a waypoint path and any possible escape routes that can be flown to avoid obstacles. Let ψ_d be the heading of the waypoint path, let R be the desired radius of the circular paths, and let ψ_m be the maximum deviation from ψ_d in the circular

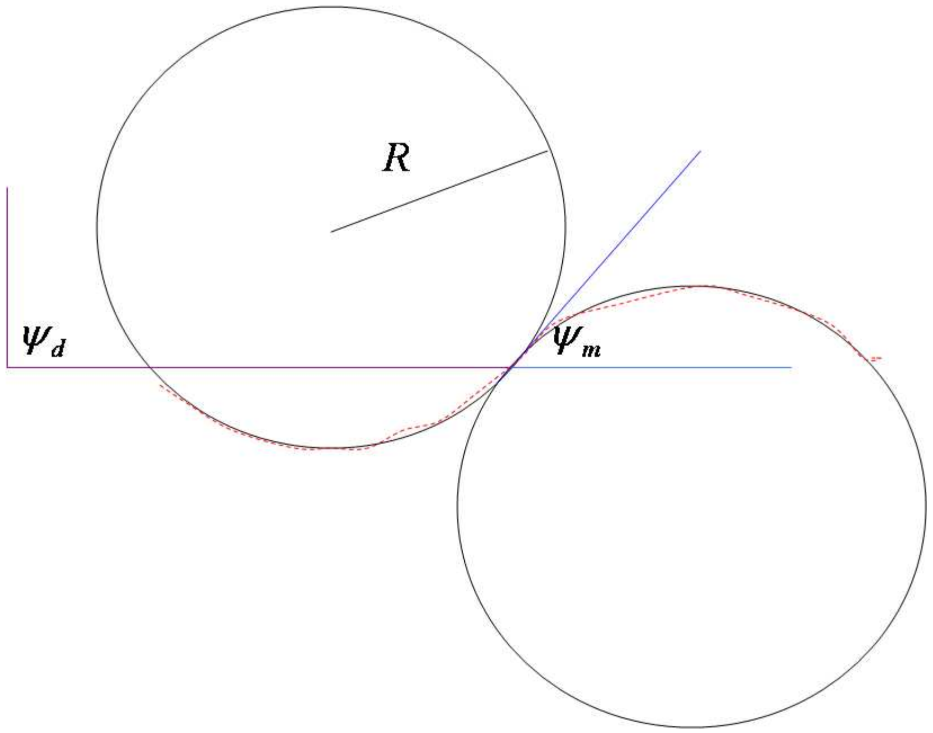


Figure 2.8: Geometry of a circular path around a waypoint path. The path of the MAV is shown in red.

oscillation as shown in Figure 2.8. Let the circular paths be arranged in such a way that they intersect at exactly one point, and the angle of the tangent line at that point be ψ_m from the waypoint path heading ψ_d. We want the MAV to track one of the circular paths until it reaches the intersection point of another circular path, where it will then begin to track the new circular path. The result is a series of oscillating circular paths around a waypoint path. We assume that the time to change roll angle is zero for simplicity.

2.2.2 Analysis

To analyze the conditions of the path that result in successful scanning, we need the properties of the path. Consider Figure 2.9. $\overline{bd}$ is the desired path of travel. In other words, its orientation is ψ_d. The angle $\angle dbg$ is ψ_m by definition. We can construct a line segment $\overline{fg}$ such that it is tangent to circles a and c, passing through the point where the circles intersect at point b. $\overline{ab}$ is a radius of circle a ending at

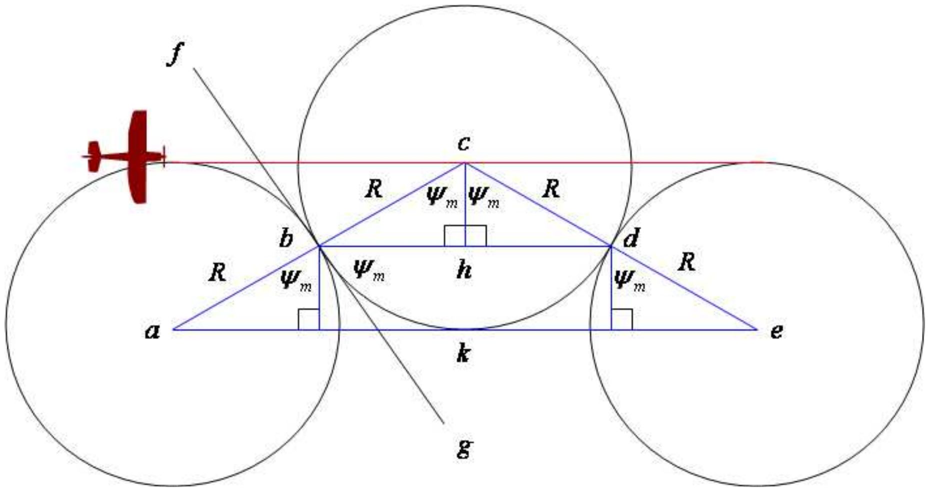


Figure 2.9: The measurements of the circular path can easily be found using geometry.

point b, where the circle a and c intersect. $\overline{bc}$ is a radius of circle c to point b. The line segment $\overline{ac}$ begins at the center of circle a and ends at the edge of circle c. $\overline{ac}$ passes through point b and is perpendicular to $\overline{fg}$. The length of $\overline{ac}$ is twice the radius of the circle, or $2R$. Using complementary angles, we know that $\angle cbd = \frac{\pi}{2} - \psi_m$. $\angle bch = \psi_m$ using the sum of angles in a triangle. Using similar logic, we can find the other angles as labeled in Figure 2.9.

Using our knowledge of the angles, we find the lengths of the edges of the triangles in Figure 2.9. We know that $\ell(\overline{bh}) = R\sin(\psi_m)$, and $\ell(\overline{ch}) = R\cos(\psi_m)$, where $\ell(x)$ is the length of x. We can now find the distance from the center of one circle to the center of another circle. The distance from a to e is $4R\sin(\psi_m)$. The distance from the center of a circle to a center n circles ahead in the path is $2nR\sin(\psi_m)$, where n is an even number. In Figure 2.9, the distance from a to e can be calculated using $n = 2$, resulting in $4R\sin(\psi_m)$.

To avoid obstacles successfully, the MAV needs to scan complete escape routes that may be flown in the case of obstacle detection. We will use the circles in the description of the path as possible escape paths from obstacles. These circles need to be scanned completely before the MAV starts traversing them to guarantee the escape route is obstacle free. We assume for a moment that ψ_d does not change.

Theorem 2.2.1 *Let ψ_m be the maximum deviation from a desired heading ψ_d in a circular scanning path. Let n be the number of circles ahead to scan and L be the range of the laser ranger. If $\psi_m > \tan^{-1}\left(\frac{\sqrt{n}}{n}\right)$ and $L \geq \sqrt{c^2 + 2Rc}$ where $c = 2R\sqrt{n^2 \sin^2(\psi_m) + \cos^2(\psi_m)}$, then a circle n circles ahead in the circular scanning path will be completely scanned.*

Proof: Refer to Figure 2.10(a). Let $\overline{ab}$ denote a tangent line of circles e and d. Let θ be the angle of $\overline{ab}$ measured from ψ_d. This represents the laser ranger of the MAV. We need to verify the laser passes this location. Draw the radii $\overline{ae}$ and $\overline{bd}$ to the intersection points of the tangent line on circles e and d respectively. The tangent line and radii are perpendicular to each other. Construct the line segment $\overline{ed}$ between the centers of the circles. This line is perpendicular to the radii and parallel to $\overline{ab}$. $\overline{ab}$ then has the same length and orientation as $\overline{ed}$. Construct the line segment $\overline{ef}$ parallel to the direction of travel, and create the line segment $\overline{df}$ perpendicular to it. $\angle def = \angle bag$ because of the parallel lines $\overline{ab}$, $\overline{ag}$, $\overline{ed}$, and $\overline{ef}$. $\angle edf = \frac{\pi}{2} - \theta$ using the sum of triangle angles. Using the geometry discussed earlier, we know that $\ell(\overline{ef}) = 2nR\sin(\psi_m)$ and $\ell(\overline{df}) = 2R\cos(\psi_m)$. The angle $\angle cab = \frac{\pi}{2} - \theta$ by complementary angles. We have shown that triangles abc and def have three equal angles and one equal side, therefore they are equivalent triangles. We can now use the lengths of the sides of triangle def on triangle abc. An equation for the relation of θ and ψ_m can be constructed as

$$\frac{\pi}{2} - \theta = \tan^{-1}\left(\frac{2nR\sin(\psi_m)}{2R\cos(\psi_m)}\right). \tag{2.9}$$

We want to know the angle where $\psi_m = \theta$, this is the smallest angle for ψ_m to ensure the laser passes this tangent line. Substituting ψ_m for θ and solving we get

$$\psi_m = \tan^{-1}\left(\frac{\sqrt{n}}{n}\right). \tag{2.10}$$

This is the angle at which the laser scans the edge of the circle. ψ_m may be larger and still cross the tangent line, therefore,

$$\psi_m \geq \tan^{-1}\left(\frac{\sqrt{n}}{n}\right). \tag{2.11}$$

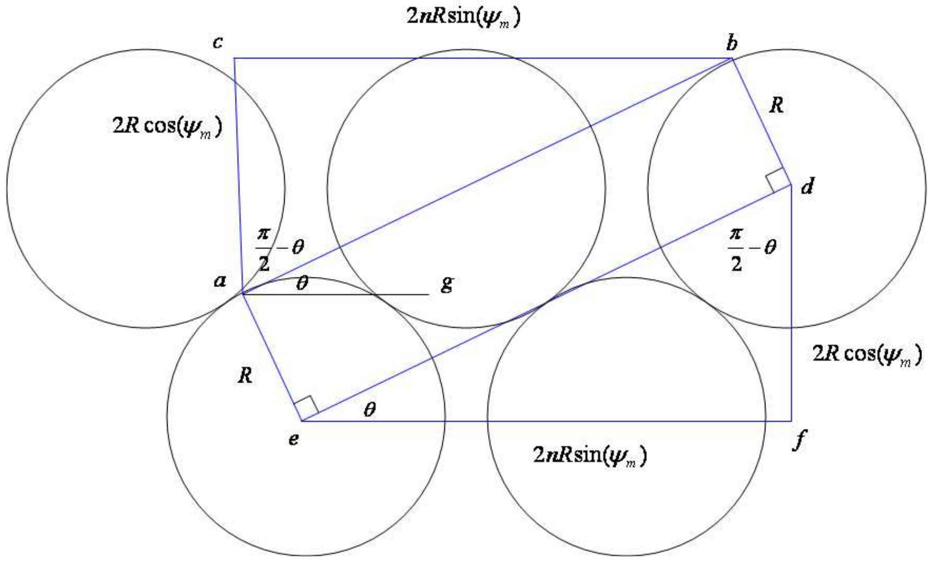
We know if $\psi_m \geq \tan^{-1}\left(\frac{\sqrt{n}}{n}\right)$, then the laser ranger begins to scan circle d. We need to show the laser ranger completes the scan. Consider Figure 2.10(b). $\overline{ab}$ is a tangent line of circles e and d with $\overline{ea}$ and $\overline{db}$ as radii of the circles to the tangent points. $\overline{ac}$ is parallel to the direction of travel (orientation ψ_d). α is the angle of the tangent line from the direction of travel. $\overline{ab}$ represents the orientation the laser ranger must achieve to completely scan circle d. We know that if $\psi_m \geq \alpha$, then circle e will be completely scanned.

In triangle abc, $\angle acb$ is $\frac{\pi}{2}-\alpha$. This implies $\angle dcj = \frac{\pi}{2}-\alpha$. By the sums of angles in a triangle, $\angle cdj = \alpha$. The length $\ell(\overline{di}) = 2R\cos\psi_m$ was derived above. Using the angle $\angle bdf$, we know $\ell(\overline{df}) = \ell(\overline{eh}) = R\cos\alpha$. Using the difference of $\ell(\overline{ji})$ and $\ell(\overline{di})$, we can find $\ell(\overline{dj}) = 2R\cos\psi_m - R\cos\alpha$. Using $\ell(\overline{dj})$, then $\ell(\overline{dc}) = \frac{2R\cos\psi_m - R\cos\alpha}{\cos\alpha}$. $\ell(\overline{cb})$ is easily seen now as:

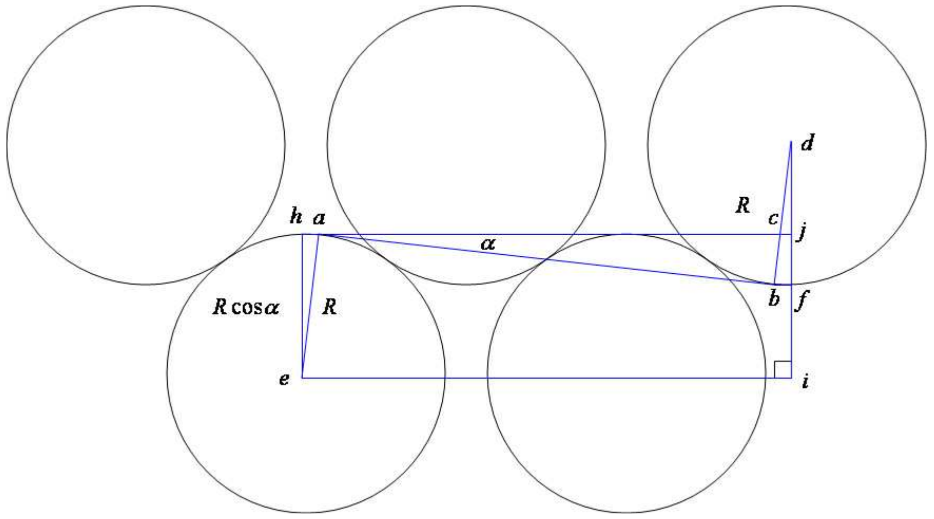
$$\ell(\overline{cb}) = R - \left(\frac{2R\cos\psi_m - R\cos\alpha}{\cos\alpha}\right) = 2R\left(1 - \frac{cos\psi_m}{\cos\alpha}\right). \tag{2.12}$$

Notice that in Equation 2.12 we require that $\frac{cos\psi_m}{\cos\alpha} \leq 1$ for the length to be positive. This implies $\cos\psi_m \leq \cos\alpha$. Given triangle abc, we know $0 \leq \alpha \leq \frac{\pi}{2}$. Given $0 \leq \psi_m \leq \pi$, then $\cos\psi_m \leq \cos\alpha \Rightarrow \psi_m \geq \alpha$. Thus the laser always achieves position $\overline{ab}$. If $\psi_m \geq \tan^{-1}\left(\frac{\sqrt{n}}{n}\right)$, then circle e is completely scanned.

The next step is to determine a sufficient laser range to ensure a circular path is completely scanned. Consider Figure 2.12(a). We can find the area the laser scans as the MAV tracks circle a. The area is a circle centered at a with a radius of $\sqrt{R^2 + L^2}$.



(a) $\overline{ab}$ represents the position the laser ranger must achieve to start scanning circle e. $\psi_m > \tan^{-1}\left(\frac{\sqrt{n}}{n}\right)$ results in the laser ranger scanning a circular route n circles ahead in the path.



(b) $\overline{ab}$ represents the position the laser ranger must achieve to finish scanning circle d. We can find that $\alpha < \psi_m$ regardless of the chosen value of ψ_m.

Figure 2.10: Geometry used in derivations of the requirements of the circular path to guarantee scanned paths.

Let circle e be a circle we desire to scan n circles ahead in the path. The distance from a to the center of circle e is

$$\ell(\overline{ae}) = \sqrt{(2Rn\sin(\psi_m))^2 + (2R\cos(\psi_m))^2}. \tag{2.13}$$

The distance from a to any point on circle e has an upper bound of $\ell(\overline{ae}) + R$. To guarantee that the laser has sufficient range to scan circle e, we can set the upper bound of the distance from a to any point on circle e equal to the radius of the circle scanned by the laser and solve for L,

$$\sqrt{R^2 + L^2} \geq \sqrt{(2Rn\sin(\psi_m))^2 + (2R\cos(\psi_m))^2} + R. \tag{2.14}$$

To simplify, let $c = \sqrt{(2Rn\sin(\psi_m))^2 + (2R\cos(\psi_m))^2}$. Then

$$\sqrt{R^2 + L^2} \geq c + R. \tag{2.15}$$

The required range of the laser to scan circle e is

$$L \geq \sqrt{c^2 + 2Rc}. \tag{2.16}$$

■

We have shown thus far the required conditions on the circular paths to guarantee obstacles will be detected before collision. In addition, these obstacles will be detected while tracking an obstacle free circle. In the case an obstacle is detected, two options are obvious. The first is to turn around using the circular orbit currently being flown and back track. The second is to fly the current circular orbit in its full 360 degrees in an attempt to scan for a new obstacle-free path in a different direction.

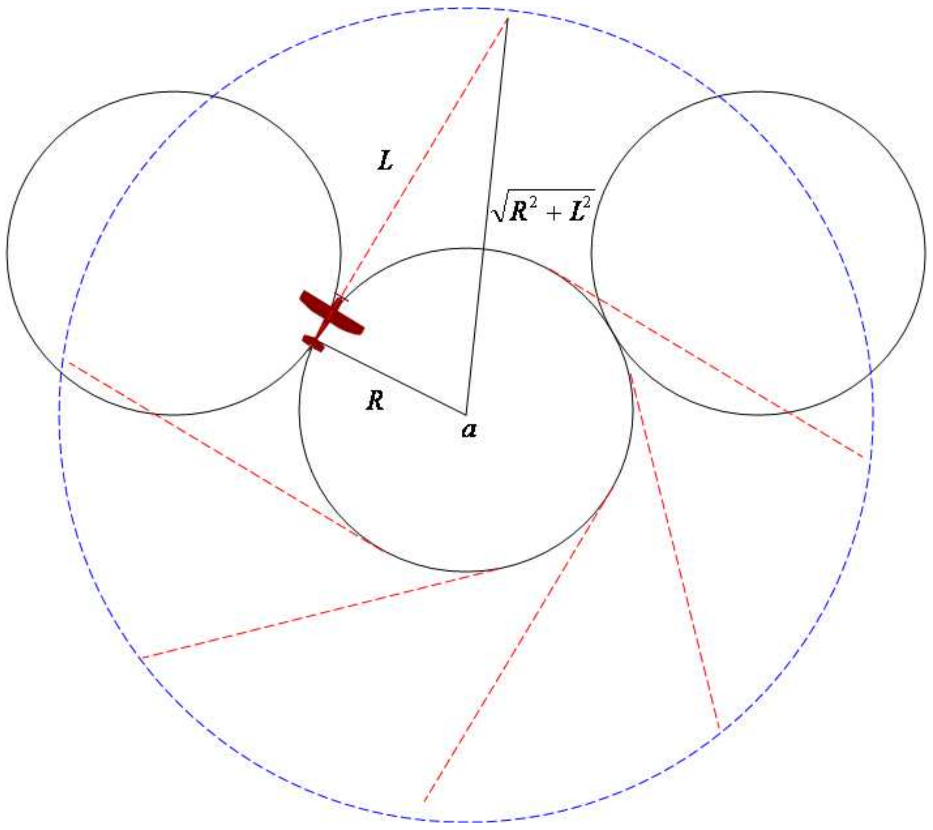
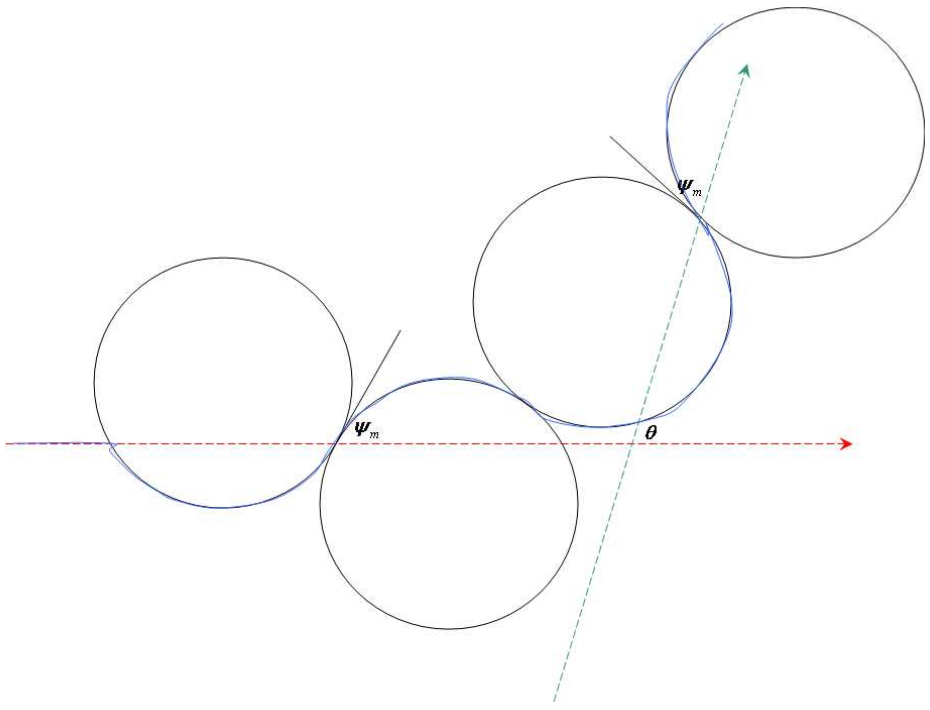


Figure 2.11: The blue circle is the area scanned by the laser as the MAV tracks circle a.

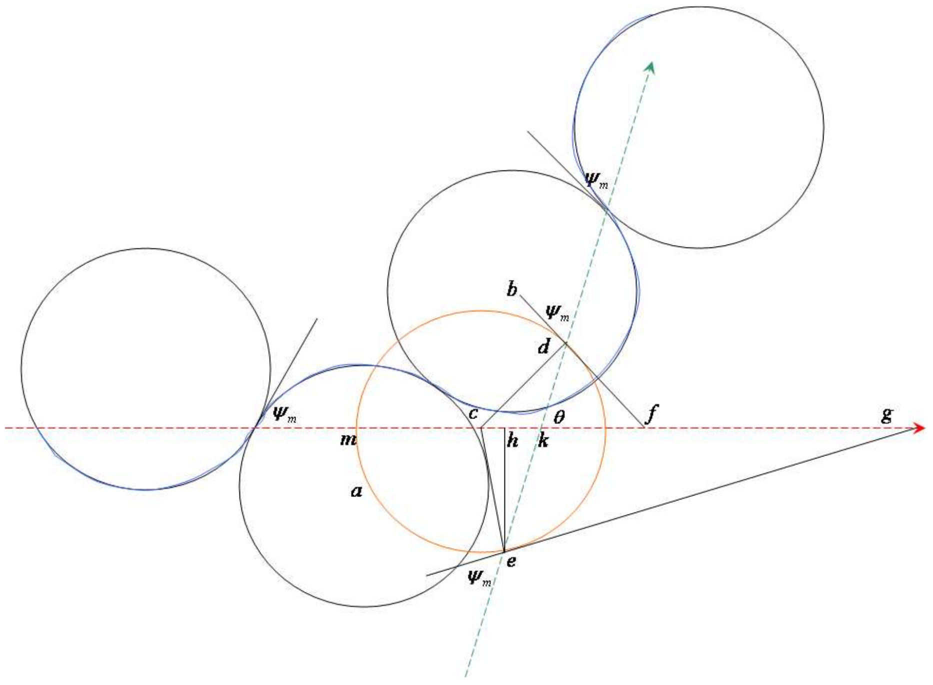
Once another obstacle free path is found, the circular paths can continue in a new direction. Either method guarantees obstacle avoidance.

2.2.3 Waypoint Path Transitions

An important part of flying waypoint paths is the transition between paths at waypoint junctions. We need to devise a method of smooth transition from one path to another. Consider Figure 2.12(a). The red dashed line represents the direction of travel of a MAV. The green dashed line is a new waypoint path the MAV must transition to. As the MAV approaches the new path, it must change direction by tracking circles of its current path until reaching a tangent with a circle of the new path, after which it can track the new path. This is represented in the path from circle a to circle b. We call the first circle of the new path the *transition circle*, shown as circle b.



(a) Simple figure of a waypoint path transition is shown and the path a MAV flies through it.



(b) Derivation for finding the center of the first circle on the new waypoint path.

Figure 2.12: The path of a MAV is shown during a waypoint change. The red dashed line represents the first waypoint path, and the green dashed line represents the second waypoint path. The blue line shows the path of a MAV flying the waypoint path transition. The goal is to find the center of circle b which is the first circle for the MAV to track on the second waypoint path.

The difficult part in the new path is finding the location of the transition circle. The derivation for finding the center of circle b is illustrated in Figure 2.12(b). We want to find circle b given the change in heading of θ and an intersection point of both paths with an offset of $(x, R\cos(\psi))$ from the center of circle a. Note that x may be negative here if the offset is to the left. The following theorem shows how to find the location of the center of circle b.

Theorem 2.2.2 *Let two waypoint paths intersect with angle θ. Let circle a be the circle on the first waypoint path nearest the intersection point without crossing it. Let x be the distance of a to the intersection point along the waypoint path. The location of the center of the transition circle is*

$$b_x = \frac{-(2\tan\theta + 2q) \pm \sqrt{(2\tan\theta + q)^2 - 4(\tan^2\theta + 1)(q^2 - 4R^2)}}{2\tan^2\theta + 2}, \tag{2.17}$$

$$b_y = b_x \tan\theta + q, \tag{2.18}$$

where

$$q = \tan\theta\left(-x + R\cos\left(\frac{\pi}{2} - \theta + \psi_m\right) + \frac{R\sin\left(\frac{\pi}{2} - \theta + \psi_m\right)}{\tan\theta}\right) + R\cos(\psi_m). \tag{2.19}$$

Proof: Refer to Figure 2.12(b) for illustration of this proof. First translate circle b along the new waypoint path $\overline{de}$ such that the center of the circle lies on the first waypoint path $\overline{cf}$, this is circle c. Line segments $\overline{df}$ and $\overline{eg}$ are tangent lines of circle c at angles of ψ_m along the direction of travel. $\overline{cd}$ and $\overline{ce}$ are radii of circle c. By opposite angles from ψ_m, $\angle edf = \angle deg = \psi_m$. By adding the sum of the angles in a triangle, $\angle egc = \theta - \psi_m$. Using the right triangle $\triangle ceg$ and summing the angles, we find $\angle ecg = \frac{\pi}{2} - \theta + \psi_m$. Construct the line segment $\overline{eh}$ perpendicular to $\overline{cf}$. Using the two right triangles formed, we can find the following,

$$\ell(\overline{ch}) = R\cos\left(\frac{\pi}{2} - \theta + \psi_m\right), \tag{2.20}$$

$$\ell(\overline{eh}) = R\sin\left(\frac{\pi}{2} - \theta + \psi_m\right), \tag{2.21}$$

$$\ell(\overline{hk}) = \frac{R\sin\left(\frac{\pi}{2} - \theta + \psi_m\right)}{\tan\theta}, \tag{2.22}$$

$$\ell(\overline{mc}) = x - \ell(\overline{ch}) - \ell(\overline{hk}) \tag{2.23}$$

$$= x - R\cos\left(\frac{\pi}{2} - \theta + \psi_m\right) - \frac{R\sin\left(\frac{\pi}{2} - \theta + \psi_m\right)}{\tan\theta}, \tag{2.24}$$

$$\ell(\overline{am}) = R\cos(\psi_m). \tag{2.25}$$

Knowing the location of c, we can shift circle c back along the waypoint path until it arrives at circle b, which can also be described as when $\ell(\overline{ac}) = 2R$. First create the equation of the line which crosses through c and runs parallel to $\overline{de}$,

$$b_y = \tan\theta\left(b_x - x + R\cos\left(\frac{\pi}{2} - \theta + \psi_m\right) + \frac{R\sin\left(\frac{\pi}{2} - \theta + \psi_m\right)}{\tan\theta}\right) + R\cos(\psi_m). \tag{2.26}$$

Let

$$q = \tan\theta\left(-x + R\cos\left(\frac{\pi}{2} - \theta + \psi_m\right) + \frac{R\sin\left(\frac{\pi}{2} - \theta + \psi_m\right)}{\tan\theta}\right) + R\cos(\psi_m). \tag{2.27}$$

The equation can be simplified to

$$b_y = b_x\tan\theta + q. \tag{2.28}$$

We need to know the distance from a to any point on this line, and find where that equals $2R$. First set the distance equal to $2R$ and solve for b_x,

$$\begin{aligned}
2R &= \sqrt{(b_x \tan\theta + q)^2 + b_x} \\
4R^2 &= (\tan^2\theta + 1)b_x^2 + 2(\tan\theta + q)b_x + q^2 \\
0 &= (\tan^2\theta + 1)b_x^2 + 2(\tan\theta + q)b_x + (q^2 - 4R^2) \\
b_x &= \frac{-(2\tan\theta + 2q) \pm \sqrt{(2\tan\theta + q)^2 - 4(\tan^2\theta + 1)(q^2 - 4R^2)}}{2\tan^2\theta + 2}.
\end{aligned}$$

b_y can be found by substituting b_x back into Equation 2.28. The result is the location of circle b as an offset from the center of circle a. Remember that these coordinates are in reference to point a and the rotation of ψ_d. To move them to world coordinates, they must be translated by $\mathbf{a}$ and rotated by $\psi_d - \frac{\pi}{2}$.

$\blacksquare$

2.2.4 Conclusion

In the first half of this chapter we derived an obstacle avoidance strategy which uses a laser ranger to detect pop-up obstacles and generate new waypoint paths around them. It was tested by planning a waypoint path through a building. The MAV successfully detected and avoided the building and continued to track the original waypoint path after avoidance.

In the second half of this chapter we described a set of circular paths that oscillate around a waypoint path for the purpose of scanning obstacles with a laser ranger. The paths scan obstacles as well as escape paths that may be utilized to prevent collision. When an obstacle is detected, a feasible path to avoid that obstacle is guaranteed. The feasible path is the circle the MAV is flying when the obstacle is detected. A method of transition between waypoint paths was also developed.

With the introduction of this new class of paths, obstacle detection is guaranteed. Methods of obstacle avoidance other than the rudimentary methods mentioned in this Section can be explored in future work. The laser ranger acquires data in real time and previously studied methods may be adapted to update paths based on new laser data. Future work may also explore 3D scanning, 3D obstacle avoidance, and terrain mapping using the laser ranger.

Chapter 3

Vision-based Reactive Avoidance

3.1 Introduction

Fixed wing micro air vehicles (MAVs) are becoming common in military and civilian use as their functionality increases and their cost decreases. Advances in integrated circuits have allowed low-power microcontrollers and the necessary sensors have become small enough to permit miniature autopilots to be used on MAVs. Current autopilots are as small as two inches square, weigh less than an ounce, [40] and may be installed on aircraft with wingspans of less than 12 inches. The size and low power requirements of these aircraft offer the potential applications of autonomous surveillance [2], terrain mapping [3], search missions [4], and mapping [5, 6] to name a few. Small cameras can be mounted onboard, allowing for visual surveillance and observation. MAVs can avoid detection due to their weak audio signal and low energy emission. The low detection probability, high functionality, and low cost create military and civilian uses not previously possible.

Although MAVs are capable of navigating waypoints autonomously, they cannot yet fly safely through urban terrain where buildings and other obstacles create a hazardous flying environment. The sensors used for obstacle detection on large aircraft are too heavy for use on MAVs. Alternatively, cameras are small and lightweight enough to mount on a MAV, making computer vision a feasible option. Laser rangers have been used in the past to detect obstacles [36], but only provide a single range measurement in the body frame. The laser ranger may be mounted on a gimbal [41], requiring a larger airframe to support the additional weight and aerodynamic drag. Scanning laser range finders are commercially available, but are currently too heavy for use on MAVs.

Strapdown cameras supported by have the potential to provide time-to-collision, object segmentation, and object identification unavailable with other sensors. Vision-based obstacle detection is a topic of research and while it has not reached reliability sufficient for commercial aircraft systems, it continues to show potential [42–44]. We assume that vision processing provides object segmentation, bearing, and time-to-collision for obstacles. Using that information, we can maneuver the MAV to push obstacles to the edge of the camera field of view. Maintaining the obstacle on the edge of the field of view guarantees that the MAV is not on a collision course with the obstacle.

Obstacle avoidance and path planning algorithms have been extensively researched in literature. For example, probability road map (PRM) methods generate random points in the configuration space and connect the points to create a navigation map of the environment [45–47]. Another probabilistic planning technique is the Rapidly-Expanding Random Tree (RRT) [7,8], which is often used in conjunction with fixed-wing air vehicles. RRTs generate random points on the configuration space and connect them to a tree of other points such that the non-holonomic constraints of the vehicle are met. Points unable to connect to the tree are removed. While these methods and their many variants [27, 28, 35, 48–51] have shown considerable success, they require time to generate paths around obstacles. A reactive planner will produce fast results and react quickly to pop-up obstacles. While the reactive planner immediately avoids pop-up obstacles, a high level path planner can produce a new waypoint path to avoid obstacles over a longer planning horizon. Reactive obstacle avoidance methods have been shown in previous work by means of dynamic replanning [52], potential fields [18], simulated annealing [19], predefined maneuvers [53], and mixed integer linear programming (MILP) [54]. MILP has been used in obstacle avoidance techniques with success, however it requires extensive computation, limiting its use for reactive avoidance.

Potential fields is a common reactive obstacle avoidance technique [18, 53, 55], but unfortunately, it and many of its variants only guarantee a high probability of avoiding obstacles. Since collisions in flight can be catastrophic, aircraft require a guarantee of obstacle avoidance. We previously developed a reactive method by generating a path on one side of obstacles and an RRT method producing waypoint paths [36]. While the reactive method produced paths around obstacles, it was not based on a continuous control law making it susceptible to oscillations in the presence of multiple obstacles. In this chapter, we will describe a continuous guidance strategy that reacts to detected obstacles within the camera field-of-view and requires very little computational processing.

Cell decomposition is another popular path planning method in which a configuration space is decomposed into cells. A path is easily found in each cell using a a variety of methods. A sequence of cells is found to connect the initial conditions to the goal [29, 30]. Additional methods include visibility graphs [56] and Voronoi diagrams [57]. Each have varying success, but neither are designed for non-holonomic vehicles.

This chapter develops a reactive guidance strategy that is intended to be used within a tiered system of path planning as illustrated in Figure 3.1. The first tier reacts instantaneously to detected obstacles using a reactive guidance strategy based on a feedback control law. The control law maneuvers the MAV to push the obstacles to the edges of the camera field-of-view essentially avoiding immediate collision with obstacles. The reactive method is the focus of this chapter. The second tier plans a waypoint path around obstacles locally in the body frame of the MAV. The third tier creates a global path from the current configuration to the goal. Each tier acts at a different time scale and has increasing computation time. The reactive guidance tier responds to obstacles where the predicted time-to-collision is zero to three seconds and requires very little computation. The tier two planner reacts to obstacles with predicted time-to-collision three to ten seconds and plans a local path that can be overridden by the reactive planner. The tier three planner reacts to obstacles with predicted time-to-collision is greater than ten seconds and plans a global path to the goal. The global plan can be overridden by the local planner in the presence of pop-up obstacles.

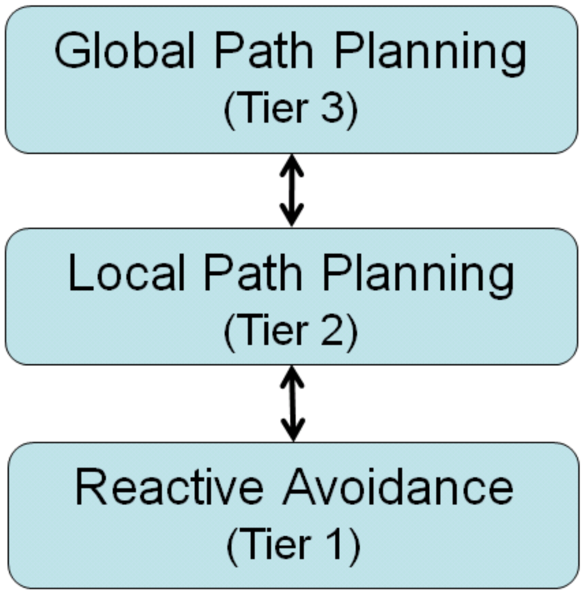


Figure 3.1: The tiered approach to path planning is shown. The first tier is the reactive layer to avoid obstacles where the time-to-collision is 0-3 seconds. The second tier plans paths in the body frame where time-to-collision is 3-10 seconds. The third tier plans paths from the initial configuration to the final configuration using global terrain maps.

The goal of this chapter is to develop a guidance strategy to maneuver MAVs around obstacles utilizing vision processing for obstacle detection. The basic idea is to push the obstacle to the edge of the camera field-of-view causing the vehicle to maneuver around the obstacle. Maintaining the obstacle at a predetermined angle prevents collision and allows time for a high level path planner to generate new waypoint paths. These results have been published in [58, 59].

The organization of the chapter is as follows. Section 3.2 introduces and analyzes the reactive obstacle avoidance of a single obstacle. Section 3.3 introduces and analyzes the reactive avoidance of multiple obstacles. We discuss methods of state estimation in Section 3.4. Simulation and flight tests are presented in Sections 3.5 and 3.6. Section 3.7 offers concluding remarks and possible future directions of research.

3.2 Single Obstacle Avoidance

This section derives and analyzes a reactive avoidance method for single obstacles.

The airframe is equipped with an onboard autopilot with inner loop control of roll, pitch, airspeed, and altitude. Assuming that airspeed and altitude are held constant, the kinematic model is given by

$$\dot{z}_n = V \cos\psi, \tag{3.1}$$

$$\dot{z}_e = V \sin\psi, \tag{3.2}$$

$$\dot{\psi} = \frac{g}{V} \tan\phi, \tag{3.3}$$

where $(z_n, z_e)^T$ is the North-East position of the aircraft, V is the airspeed, ϕ is the commanded roll angle, ψ is the heading, and g is the gravitational constant and where we have assumed zero ambient wind. The airframe is a fixed-wing micro air vehicle with a strapdown camera mounted parallel to the x-axis of the body frame. A forward looking camera allows objects to be viewed in the short reachable region of the MAV.

Maps of terrain and potential obstacles within those terrains are often available before launch and are used to plan a collision-free trajectory. However, those maps may not be accurate, requiring a sensor to detect pop-up obstacles. We assume a time-to-collision map is available from vision processing algorithm [9, 10, 34, 60]. Time-to-collision estimation is discussed in section 3.4.

We use geometry in the MAV body frame to derive the relative equations of motion between the MAV and the center of the obstacle, as shown in Figure 3.2. Let ρ be the range from the MAV to the center of the obstacle and let η be the bearing to the center of the obstacle. The equations of motion for the system are

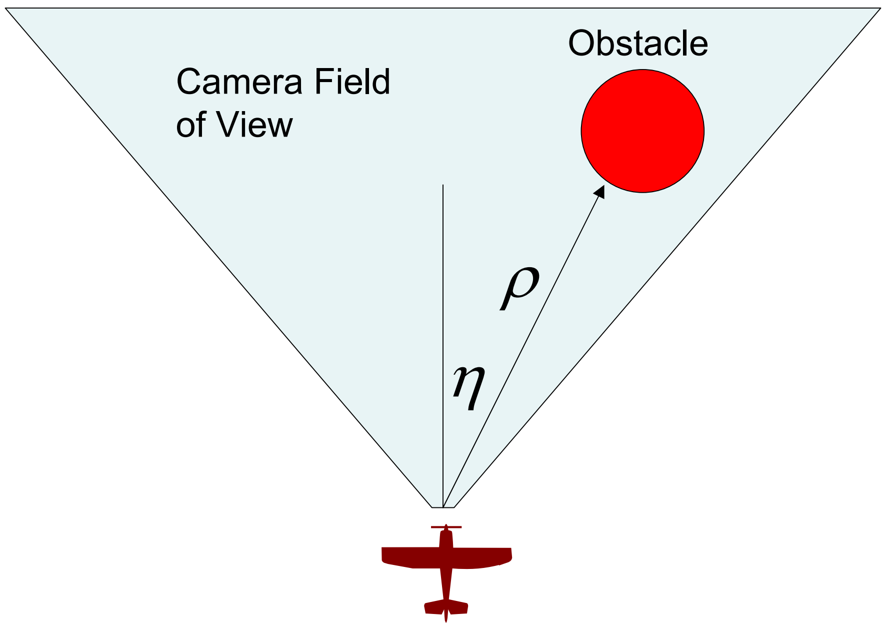


Figure 3.2: A conceptual view of the MAV approaching an obstacle. The guidance law moves the obstacle to the edge of the camera field of view to avoid collision.

$$\dot{\rho} = -V\cos\eta, \tag{3.4}$$

$$\dot{\eta} = \frac{V}{\rho}\sin\eta - \frac{g}{V}\tan\phi. \tag{3.5}$$

The objective is to develop a guidance strategy that effectively pushes the edge of the obstacle to a specified angle η_d in the image plane. The idea is that if the obstacle is pushed out of the field of view of the camera, then it can no longer be tracked by the guidance algorithm and a collision may not necessarily be avoided. We will assume that the obstacle is a cylinder of radius R, and pose the guidance problem with respect to the range and bearing to the edge of the cylinder as shown in Figure 3.3.

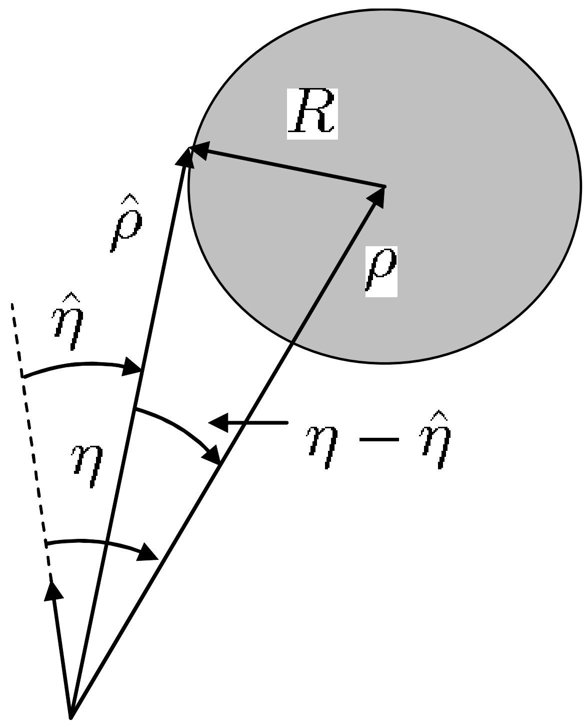


Figure 3.3: Change of variables from the range and bearing to the center of the obstacle (ρ, η), to the range and bearing to the edge of the obstacle $(\hat{\rho}, \hat{\eta})$.

From Figure 3.3 we see that

$$R = \rho \sin(\eta - \hat{\eta}), \tag{3.6}$$

$$\hat{\rho} = \rho \cos(\eta - \hat{\eta}). \tag{3.7}$$

Differentiating (3.6) with respect to time and using Equations (3.4) and (3.5) we get

$$\begin{aligned}
&\dot{R} = 0 = \dot{\rho} \sin(\eta - \hat{\eta}) + (\dot{\eta} - \dot{\hat{\eta}})\rho \cos(\eta - \hat{\eta}) \\
\implies & -V \cos\eta \sin(\eta - \hat{\eta}) + V \sin\eta \cos(\eta - \hat{\eta}) \\
& -\rho \cos(\eta - \hat{\eta})(\frac{g}{V} \tan\phi + \dot{\hat{\eta}}) = 0 \\
\implies & V \sin\hat{\eta} - \rho \cos(\eta - \hat{\eta})(\frac{g}{V} \tan\phi + \dot{\hat{\eta}}) = 0 \\
\implies & \dot{\hat{\eta}} = -\frac{g}{V} \tan\phi + \frac{V}{\rho} \frac{\sin\hat{\eta}}{\cos(\eta - \hat{\eta})}.
\end{aligned}$$

Substituting from Equation (3.7) gives

$$\dot{\hat{\eta}} = -\frac{g}{V}\tan\phi + \frac{V}{\hat{\rho}}\sin\hat{\eta}. \tag{3.8}$$

Similarly, by differentiating Equation (3.7) and simplifying we get

$$\dot{\hat{\rho}} = -V\left[\cos\hat{\eta} - \frac{R}{\hat{\rho}}\sin\hat{\eta}\right]. \tag{3.9}$$

If $\bar{\eta}$ is the field of view of the camera, then we desire that $|\eta| \leq \bar{\eta}$. Let $|\eta_d| < \bar{\eta}$ be the desired position of the edge of the obstacle in the image plane. Consider the Lyapunov function candidate

$$W = \frac{1}{2}(\hat{\eta} - \eta_d)^2,$$

and differentiate to obtain

$$\begin{aligned}\dot{W} &= (\hat{\eta} - \eta_d)\dot{\hat{\eta}} \\ &= (\hat{\eta} - \eta_d)\left(-\frac{g}{V}\tan\phi + \frac{V}{\hat{\rho}}\sin\hat{\eta}\right).\end{aligned}$$

Therefore, selecting the guidance law as

$$\phi = \tan^{-1}\left(\frac{V^2}{g\hat{\rho}}\sin\hat{\eta} + \frac{Vk}{g}(\hat{\eta} - \eta_d)\right), \tag{3.10}$$

gives

$$\dot{W}_1 = -k(\hat{\eta} - \eta_d)^2,$$

which implies that $\hat{\eta}(t) \rightarrow \eta_d$.

Theorem 3.2.1 *By keeping the object at angle η_d in the camera field of view using the guidance strategy*

$$\phi = \tan^{-1}\left(\frac{V^2}{g\hat{\rho}}\sin\hat{\eta} + \frac{Vk}{g}(\hat{\eta} - \eta_d)\right), \tag{3.11}$$

the MAV converges to a limit cycle of radius $\rho(t) = R\sqrt{1+\tan^2\eta_d}$ *and* $\hat{\rho}(t) \to R\tan\eta_d$.

Proof: As $\hat{\eta} \to \eta_d$, from Equation (3.9) we get that

$$\dot{\hat{\rho}} \to -V\left[\cos\eta_d - \frac{R}{\hat{\rho}(t)}\sin\eta_d\right].$$

We argue that $\dot{\hat{\rho}} \to 0$. Suppose that as $\hat{\eta} \to \eta_d$, that $\dot{\hat{\rho}} > 0$, then

$$\begin{aligned} &-V\left[\cos\eta_d - \frac{R}{\hat{\rho}(t)}\sin\eta_d\right] > 0 \\ &\Leftrightarrow \cos\eta_d < \frac{R}{\hat{\rho}(t)}\sin\eta_d \\ &\Leftrightarrow \hat{\rho}(t) < R\tan\eta_d. \end{aligned}$$

Since $\dot{\hat{\rho}} > 0$, $\hat{\rho}$ is monotonically increasing, bounded above, and will converge, which implies that $\dot{\hat{\rho}} \to 0$. On the other hand, suppose that $\dot{\hat{\rho}} < 0$, then

$$\begin{aligned} &-V\left[\cos\eta_d - \frac{R}{\hat{\rho}(t)}\sin\eta_d\right] < 0 \\ &\Leftrightarrow \cos\eta_d > \frac{R}{\hat{\rho}(t)}\sin\eta_d \\ &\Leftrightarrow \hat{\rho}(t) > R\tan\eta_d. \end{aligned}$$

Since $\dot{\hat{\rho}} < 0$, $\hat{\rho}$ is monotonically decreasing, bounded below, and will converge, which implies that $\dot{\hat{\rho}} \to 0$. Therefore, as $\hat{\eta}(t) \to \eta_d$, $\dot{\hat{\rho}} \to 0$, which implies that $\hat{\rho}(t) \to R\tan\eta_d$.

By holding the edge of a cylindrical object at angle η_d in the camera field of view, that the MAV will converge to the limit cycle around the object shown in Figure 3.4. From geometry we have that

$$\rho^2 = \hat{\rho}^2 + R^2 = R^2(1+\tan^2\eta_d).$$

Therefore

$$\rho(t) \to R\sqrt{1 + \tan^2 \eta_d}.$$

■

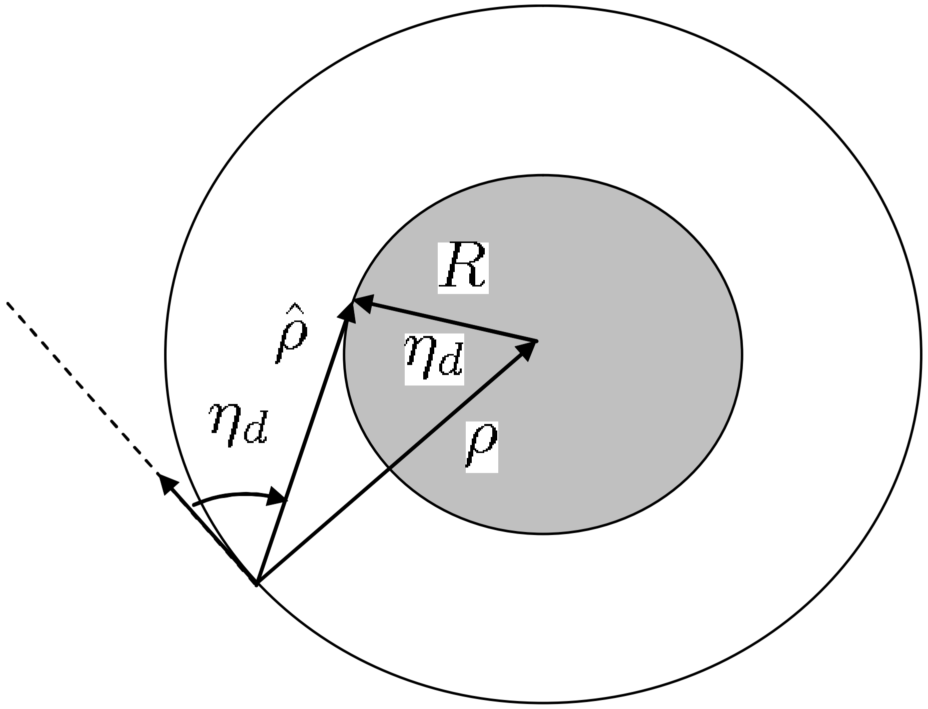


Figure 3.4: By keeping the object at angle η_d in the camera field of view, the MAV converges to a limit cycle of radius $R\sqrt{1 + \tan^2 \eta_d}$.

If the MAV is following a waypoint path and a cylindrical obstacle pops-up in the image, the collision avoidance strategy is then to push the edge of the object to an angle of η_d in the image plane using guidance strategy (3.11), until the MAV moves well past the obstacle and can resume tracking its original path.

3.3 Multiple Obstacles

The purpose of this section is to derive and analyze a reactive avoidance method for multiple obstacles.

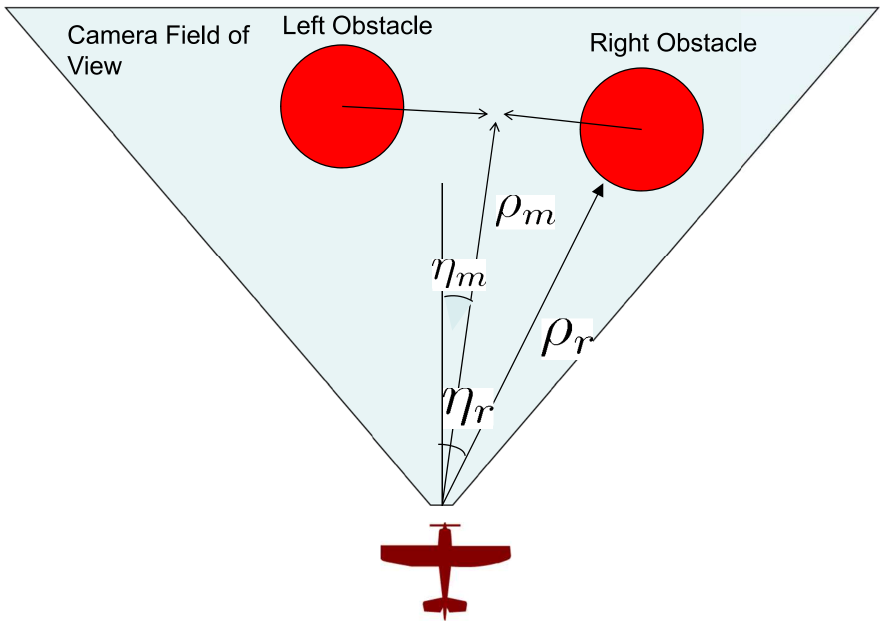


Figure 3.5: A conceptual view of the MAV approaching 2 obstacles is shown. The objective of the control is to fly between the obstacles to avoid collision.

To avoid a large number of obstacles at a constant altitude, the MAV will have to maneuver between two obstacles, to the left of a single obstacle, or to the right of a single obstacle, regardless of the total number of obstacles in the current path. We will simplify multiple obstacle avoidance into the case of avoiding two obstacles at a time, a left obstacle and a right obstacle. In section 3.3.1, we present an algorithm to decide the best set of obstacles to maneuver between.

Let ρ_r and ρ_l denote the range to the right and left obstacles in the camera field-of-view and let η_r and η_l be the bearing to the right and left obstacles. An illustration of the obstacles in relation to the MAV body frame and camera field-of-view is shown in Figure 3.5.

Let ρ_m and η_m be the range and bearing to the midpoint between the two obstacles where

$$\rho_m = \frac{\rho_r + \rho_l}{2},$$
$$\eta_m = \frac{\eta_r + \eta_l}{2}.$$

The midpoint is a static point in the inertial frame and therefore moves in the body frame. The dynamics of the obstacle in the MAV body frame are generalized from (3.4) and (3.5) as

$$\dot{\rho}_* = - V_a \cos \eta_*, \tag{3.12}$$
$$\dot{\eta}_* = \frac{V_a}{\rho_*} \sin \eta_* - \frac{g}{V_a} \tan \phi, \tag{3.13}$$

where $*$ may denote right obstacle r, left obstacle l, or midpoint m.

The objective is to fly between the obstacles. Our approach to this problem is to balance the obstacles on either side of the image plane. In the absence of wind, the course angle of the MAV is the angle of the optical axis. Balancing the obstacles on opposite sides of the camera field-of-view effectively steers the MAV on a collision-free path.

Moving the obstacles to the edges of the image plane will not ensure the MAV does not come in close proximity of an obstacle. If the MAV approaches the obstacles at a steep angle, as shown in Figure 3.6, it will come in close proximity to one of them, and possibly collide. To avoid this problem, we also regulate the difference in range-to-obstacles $\rho_d = \rho_r - \rho_l$ to zero. If the guidance loop maintains $\rho_r = \rho_l$, then in the scenario in Figure 3.6 the MAV will roll away from the obstacles to force ρ_d to zero while simultaneously regulating η_m to zero.

We use backstepping to derive the guidance strategy to regulate η_m and ρ_d. Consider the following Lyapunov function candidate

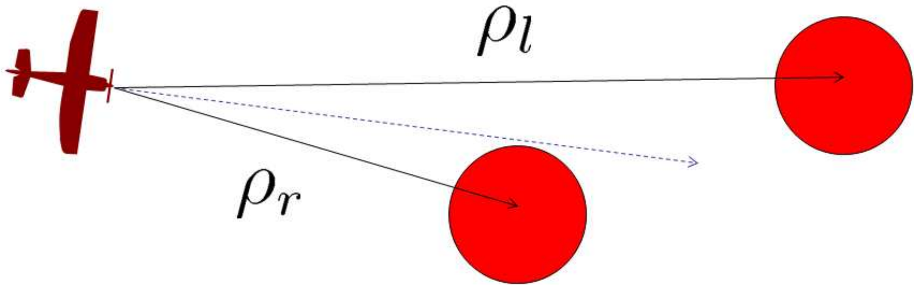


Figure 3.6: If the MAV avoids the obstacles by only moving the obstacles to the edges of the image plane, the MAV may come in close proximity to the obstacle. To avoid this problem, force $\rho_r - \rho_l$ to zero as well.

$$W_1 = \frac{k_1}{2}\rho_d^2,$$

and differentiate to obtain

$$\begin{aligned} \dot{W}_1 &= \rho_d \left(-V \cos \eta_r + V \cos \eta_l\right) \\ \dot{W}_1 &= \rho_d \left(-V \cos \left(2\eta_m - \eta_l\right) + V \cos \eta_l\right). \end{aligned} \tag{3.14}$$

Assume for the moment that η_m is a controllable input and let $\beta = \eta_m$ be the input, then

$$\dot{W}_1 = \rho_d \left(-V \cos \left(2\beta - \eta_l\right) + V \cos \eta_l\right).$$

Letting

$$\beta = \frac{1}{2}\left(\eta_l + \cos^{-1}\left(\cos \eta_l + \frac{k_1}{V}\rho_d\right)\right), \tag{3.15}$$

we get

$$\dot{W}_1 = -\, k_1^2 \rho_d^2.$$

Let $\zeta = \eta_m - \beta$ and differentiate to obtain

$$\begin{aligned}\dot{\zeta} =& \dot{\eta}_m - \dot{\beta} \\ =& \frac{V}{\rho_m} \sin \eta_m - \frac{g}{V} \tan \phi - \dot{\beta},\end{aligned} \tag{3.16}$$

where

$$\begin{aligned}\dot{\beta} =& \frac{1}{2} \left[\frac{V}{\rho_l} \sin \eta_l - \dot{\psi} - \frac{1}{\sqrt{1 - cos\eta_l - \frac{k_1}{V}\rho_d}} \right. \\ & \left. \left(- \sin \eta_l \left(\frac{V}{\rho_l} \sin \eta_l - \dot{\psi} \right) + k_1 \left(-cos\eta_r + cos\eta_l \right) \right) \right].\end{aligned} \tag{3.17}$$

Let W_2 be defined as

$$W_2 = \frac{k_1}{2} \rho_d^2 + \frac{1}{2} \zeta^2,$$

and differentiate to obtain

$$\begin{aligned}\dot{W}_2 =& k_1\rho_d\dot{\rho_d} + \zeta\dot{\zeta} \\ =& k_1\rho_d\left[-V\cos\left(2\eta_m - 2\beta - \eta_l + 2\beta\right) + V\cos\eta_l\right] + \zeta\dot{\zeta} \\ =& k_1\rho_d\left[-V\cos\left(2\zeta\right)\cos\left(2\beta - \eta_l\right) + V\sin\left(2\zeta\right)\sin\left(2\beta - \eta_l\right) + V\cos\eta_l\right] + \zeta\dot{\zeta} \\ =& k_1\rho_d\left[-V\cos\left(2\zeta\right)\left(\cos\eta_l + \frac{k_1}{V}\rho_d\right) + V\sin\left(2\zeta\right)\sqrt{1-\left(\cos\eta_l + \frac{k_1}{V}\rho_d\right)^2}\right. \\ & \left. + V\cos\eta_l\right] + \zeta\dot{\zeta}.\end{aligned}$$

Using an infinite series expansion for $\cos(2\zeta)$ and $\sin(2\zeta)$ we obtain

$$\begin{aligned}\dot{W}_2 =& k_1\rho_d\left[-V\left(1 - \frac{(2\zeta)^2}{2!} + ...\right)\left(\cos\eta_l + \frac{k_1}{V}\rho_d\right)\right. \\ & + V\left(2\zeta - \frac{(2\zeta)^3}{3!} + ...\right)\sqrt{1-\left(\cos\eta_l + \frac{k_1}{V}\rho_d\right)^2} \\ & \left. + V\cos\eta_l\right] + \zeta\dot{\zeta}.\end{aligned}$$

By extracting ζ and simplifying, $\dot{W}_2$ becomes

$$\begin{aligned}\dot{W}_2 =& k_1\rho_d\left[-V\left(\cos\eta_l + \frac{k_1}{V}\rho_d\right) + V\cos\eta_l\right] \\ & + \zeta\left[k_1\rho_d\left(-V\left(-\frac{2(2\zeta)}{2!} + \frac{2(2\zeta)^3}{4!} + ...\right)\left(\cos\eta_l + \frac{k_1}{V}\rho_d\right)\right.\right. \\ & \left.\left. + V\left(2 - \frac{2(2\zeta)^2}{3!} + ...\right)\sqrt{1-\left(\cos\eta_l + \frac{k_1}{V}\rho_d\right)^2}\right) + \dot{\zeta}\right]\end{aligned}$$

$$\dot{W}_2 = -k_1^2\rho_d^2 + \zeta\left[2k_1\rho_d V\left(\sum_{i=0}^{\infty}\frac{(2\zeta)^{2i+1}(-1)^i}{(2i+2)!}\left(\cos\eta_l + \frac{k_1}{V}\rho_d\right)\right.\right.$$
$$\left.\left.+\sum_{i=0}^{\infty}\frac{(2\zeta)^{2i}(-1)^i}{(2i+1)!}\sqrt{1-\left(\cos\eta_l + \frac{k_1}{V}\rho_d\right)^2}\right) + \dot{\zeta}\right]. \tag{3.18}$$

Substituting Equation (3.16) for $\dot{\zeta}$ into equation (3.18) gives

$$\dot{W}_2 = -k_1^2\rho_d^2 + \zeta\left[2k_1\rho_d V\left(\sum_{i=0}^{\infty}\frac{(2\zeta)^{2i+1}(-1)^i}{(2i+2)!}\left(\cos\eta_l + \frac{k_1}{V}\rho_d\right)\right.\right.$$
$$\left.\left.+\sum_{i=0}^{\infty}\frac{(2\zeta)^{2i}(-1)^i}{(2i+1)!}\sqrt{1-\left(\cos\eta_l + \frac{k_1}{V}\rho_d\right)^2}\right) + \frac{V}{\rho_m}\sin\eta_m - \frac{g}{V}\tan\phi - \dot{\beta}\right]. \tag{3.19}$$

The control signal in Equation (3.19) is the roll angle ϕ. To stabilize the system, let

$$\phi = \tan^{-1}\left[\frac{V}{g}\left(\frac{V}{\rho_m}\sin\eta_m - \dot{\beta} + k_2\zeta + 2k_1\rho_d V\left(\sum_{i=0}^{\infty}\frac{(2\zeta)^{2i+1}(-1)^i}{(2i+2)!}\left(\cos\eta_l + \frac{k_1}{V}\rho_d\right)\right.\right.\right.$$
$$\left.\left.\left.+\sum_{i=0}^{\infty}\frac{(2\zeta)^{2i}(-1)^i}{(2i+1)!}\sqrt{1-\left(\cos\eta_l + \frac{k_1}{V}\rho_d\right)^2}\right)\right)\right], \tag{3.20}$$

which results in the negative definite Lyapunov function

$$\dot{W}_2 = -k_1^2\rho_d^2 - k_2\zeta^2. \tag{3.21}$$

The following theorem shows that the guidance law in Equation (3.20) causes the MAV to converge to the perpendicular bisector of the two obstacles.

Theorem 3.3.1 *Using the guidance strategy in Equation* (3.20), *the MAV trajectory asymptotically approaches the line* $\rho_r = \rho_l$.

Proof: From (3.21) we have that $\rho_d \to 0$ and $\eta_m \to 0$ simultaneously. If $\rho_d = 0$, then $\rho_r = \rho_l$ implying the MAV is on the line $\rho_r = \rho_l$. Equation (3.21) is negative definite, implying $\rho_d \to 0$, hence the MAV approaches the line $\rho_r = \rho_l$. ■

Notice that $\cos^{-1}$ in (3.17) adds the constraint $\cos \eta_l + \frac{k_1}{V}\rho_d < 1$ to the system. We can simplify the constraint by letting $\eta_l = \bar{\eta}$ and $\rho_d = \rho_{max}$ where ρ_{dmax} is the maximum distance possible between any two obstacles (ρ_{dmax} depends on your camera field of view and sensor range). Simplifying yields

$$\cos \bar{\eta} + \frac{k_1}{V}\rho_{dmax} < 1. \tag{3.22}$$

Letting $\bar{\eta} = 40°$ and $\rho_{dmax} = 50$, typical values discussed in section 3.5.2, sets the left side of (3.22) to 0.8, making this a reasonable constraint.

3.3.1 High-level Decision Making

In a general urban environment, any number of obstacles may clutter the path of the MAV, not just two at a time. In addition, there may not be any need to maneuver between obstacles if they are not in the current path of the MAV. Hence, we require a higher level decision strategy to decide if avoidance is required and which obstacles to maneuver between.

To decide if obstacle avoidance is necessary, we need to parse obstacles from the time-to-collision map provided by vision processing [10]. We assume all obstacles are static in the inertial frame, therefore we can calculate a depth map from the time-to-collision map. Since the MAV maintains constant altitude, we only need to consider obstacles at the current altitude of the MAV. To find obstacles at the altitude of the MAV, we scan a line of the depth map that is rotated to remove the effects of roll and translated to remove the effects of pitch. Let $x \in \left[-\frac{S_x}{2}, \frac{S_x}{2}\right]$ and $y \in \left[-\frac{S_y}{2}, \frac{S_y}{2}\right]$ be the horizontal and vertical coordinates of a pixel in the camera field-of-view, where S_y and S_x are the vertical and horizontal number of pixels of the camera. The equation of the scan line in the depth map removing roll and pitch is,

$$y = \tan\phi \left(x - \frac{S_x}{2} \right) + \frac{S_y}{2} - \frac{\theta S_y}{\varphi_V} \cos\phi, \tag{3.23}$$

where φ_V is the vertical field-of-view of the camera. The scan line reduces computation by examining only the obstacles corresponding to same altitude as the MAV. The steps to choose which obstacles to avoid are outlined in Algorithm 1.

Algorithm 1: High level decision making for constant altitude obstacle avoidance

```
   Data: Minimum horizontal obstacle separation $\delta d_{hmin}$, minimum safe distance
         from obstacles $d_{min}$
   Input: Inertial location, euler angles, camera depth map, waypoints
   Output: Commanded roll angle
 1 Initialization;
 2 Calculate scan line, removing roll and pitch;
 3 Input scan line from depth map;
 4 Find obstacles within minimum safe distance;
 5 Determine if obstacles are on current path;
   if Obstacles are in path then
 6     Insert pseudo obstacles to left of and depth of leftmost obstacle, and right
       of and depth of rightmost obstacle;
 7     $\eta_m = 2\pi$;
       for Number of Obstacles do
 8         $d_i$ = distance between obstacles;
 9         $m_i$ = midpoint between edges of obstacles;
10         $\eta_{m_i}$ = horizontal angle from optical axis to $m_i$;
           if $\eta_{m_i} < \eta_m$ and $d_i > \delta d_{hmin}$ then
11             $m = m_i$;
           end
       end
12     Use $m$ in guidance law;
13     Return roll from guidance law;
   else
14     Return roll to continue on waypoint path;
   end
```

The first step is to calculate and input the scan line from the depth map in lines $1-3$. By iterating through the scan line, we can find obstacles that are within a predefined minimum safe distance. If any of those obstacles are in the path of the MAV, then obstacle avoidance is required. The guidance law maneuvers between two obstacles, and the MAV may need to maneuver to the left of the leftmost obstacle or to the right of the rightmost obstacle. Therefore, we insert pseudo obstacles to the left of the leftmost obstacle and the right of the rightmost obstacle in line 6. To minimize heading change, find the midpoint m between obstacles that requires the smallest change in heading of the MAV and that also passes the minimum distance from the obstacles in lines $7-11$. Use midpoint m for the guidance law to return a roll angle in lines $12-13$. In the case that no obstacles block the path of the MAV, the algorithm returns the roll angle to continue along the waypoint path in line 14.

Ideally, we want the guidance strategy to avoid obstacles in three dimensions. Algorithm 1 operates in two dimensions, but can be modified to operate in three dimensions by varying altitude. Toward that end, we derive a guidance law for altitude control based on the MAV pitch angle. The dynamics of vertical obstacle movement in the body frame are similar to horizontal movement and characterized as

$$\dot{\eta}_\theta = -\dot{\theta} + \frac{V}{\rho_m}\sin\eta_\theta. \tag{3.24}$$

where $\dot{\theta}$ is the pitch rate and η_θ is the vertical angle from the optical axis to the midpoint m. We assume that the angle of attack is close to zero. Consider the Lyapunov function candidate where

$$\begin{aligned} W &= \frac{1}{2}\eta_\theta^2, \\ \dot{W} &= \frac{1}{2}\eta_\theta\dot{\eta}_\theta \\ &= \frac{1}{2}\eta_\theta\left(-\dot{\theta} + \frac{V}{\rho_m}\sin\eta_\theta\right). \end{aligned} \tag{3.25}$$

Let

$$\dot{\theta} = \frac{V}{\rho_m} \sin \eta_\theta + k_3 \eta_\theta, \tag{3.26}$$

resulting in regulating η_θ and consequently change altitude to the level of the midpoint m. The Lyapunov function candidate simplifies to

$$\dot{W} = k_3 \eta_\theta^2. \tag{3.27}$$

Theorem 3.3.2 *Using the guidance law described by* (3.26)*, the angle* η_θ *is asymptotically stable.*

Proof: Substituting the guidance law (3.26) into the Lyapunov function (3.25) results in the negative definite function (3.27). Therefore, the angle η_θ is asymptotically stable. ■

Since we assume the angle of attack is close to zero, a thrust change may be necessary to minimize the angle of attack.

Using (3.26) as the altitude control, we can modify Algorithm 1 to Algorithm 2 to add altitude variation. The basic idea is to scan for obstacles at multiple altitudes to determine if a small change in altitude would reduce the heading change required to avoid obstacles. While the code outline is much larger, the main difference is the loops for multiple scan lines at different altitudes and a cost function to determine the best two obstacles to maneuver between. The cost function incorporates the required pitch change and the required heading change to maneuver between two obstacles. Since altitude change has slower response time than heading change, the cost of a pitch variation is calculated as twice the cost of a heading variation. Hence the cost of a midpoint between two obstacles is $2\theta_{m_i} + \eta_{m_i}$ where θ_{m_i} is the pitch required to change altitude and η_{m_i} is the horizontal angle to the midpoint from the optical axis. The result is a midpoint between two obstacles with minimal cost.

Algorithm 2: Variable altitude obstacle avoidance

```
1  for Number of vertical scan lines do
2      Calculate scan line, removing roll and pitch;
3      Input scan line from depth map;
4      Find obstacles within minimum safe distance;
5      if Obstacles are in current path then
6          Collision flag = 1;
       end
   end
   if Collision flag = 1 then
       for Number of vertical scan lines do
7          Insert pseudo obstacles to left of leftmost obstacle, and right of
           rightmost obstacle;
           for number of obstacles on scan line-1 do
8              $d_i$ = distance between obstacles;
9              $m_i$ = midpoint between obstacle and next obstacle;
10             $\theta_{m_i}$ = pitch required to change to that altitude;
11             $\eta_{m_i}$ = horizontal angle to $m_i$;
12             cost = $2 * \theta_{m_i} + \eta_{m_i}$;
               if cost is smallest found and $d_i$ greater than minimum horizontal
               obstacle separation and $\theta_{m_i} < \theta_{max}$ then
13                 min cost = cost;
14                 $m = m_i$;
               end
           end
       end
       if No $m$ found within constraints then
15         return maximum roll and maximum climb angle;
       else
16         Calculate roll $\phi$ from guidance law using $m$;
17         Calculate pitch $\theta$ from guidance law using $m$;
18         return roll and pitch;
       end
   else
       Return roll and pitch to continue on waypoint path;
   end
```

3.4 State Estimation

We assume the autopilot provides the Euler angles of the MAV. Time-to-collision and bearing-to-obstacle estimates are provided by vision processing, described in [10]. Range-to-obstacle is estimated from time-to-collision because the obstacles are static. Here we will discuss an alternate method of estimating the range-to-obstacle ρ.

Range and Bearing

Bearing-to-obstacle η will be measured in the vehicle frame. This angle cannot be measured directly from the image plane because the roll angle can affect the obstacle location in the image plane. However, the roll angle can be removed from the image plane by a simple rotation. Let $(\mathbf{u}, \mathbf{v})$ be the location of the obstacle in the image plane. The roll is removed by the transformation

$$\begin{pmatrix} \hat{\mathbf{u}} \\ \hat{\mathbf{v}} \end{pmatrix} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}. \tag{3.28}$$

After the transformation, η is given by

$$\eta = \tan^{-1}\left(\frac{\hat{\mathbf{u}}}{f}\right), \tag{3.29}$$

where f is the focal length of the camera in units of pixels.

A range estimate can be derived from the equations of motion. Equation (3.5) is a function of range-to-obstacle. If we measure $\dot{\eta}$ from movement in the image plane, then (3.5) can be used as a measurement in an Extended Kalman Filter (EKF) to estimate the range-to-obstacle. Incorporating the range and bearing as states, the equations of motion are

$$f(\mathbf{x}) = \begin{pmatrix} \dot{\rho}_* \\ \dot{\eta}_* \end{pmatrix} = \begin{pmatrix} \frac{V}{\rho_*}\cos\eta_* \\ \frac{V}{\rho_*}\sin\eta_* - \frac{g}{V}\tan\phi \end{pmatrix}, \tag{3.30}$$

$$h(\mathbf{x}) = \begin{pmatrix} \dot{\eta}_* \\ \eta_* \end{pmatrix} = \begin{pmatrix} \frac{V}{\rho_*}\sin\eta_* - \frac{g}{V}\tan\phi \\ \eta_* \end{pmatrix}. \tag{3.31}$$

The Jacobians of the equations of motion and output are:

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} 0 & V\sin\eta \\ -\frac{V}{\rho^2}\sin\eta & \frac{V}{\rho}\cos\eta \end{pmatrix}, \tag{3.32}$$

$$\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} -\frac{V}{\rho^2}\sin\eta & \frac{V}{\rho}\cos\eta \\ 0 & 1 \end{pmatrix}. \tag{3.33}$$

The above equations are sufficient to implement an EKF to estimate range and bearing.

3.4.1 Time Delay

The proposed system utilizes computer vision to track obstacles and provide range and bearing estimates to those obstacles. As a result, the time delay introduced by sending the video to the ground, processing, and returning the results, may be significant enough to adversely affect the guidance strategy. We address this problem using an Out-Of-Sequence Measurement Extended Kalman Filter (OOSM-EKF) [61–64].

The approach in this section differs from [61–64] in that we do not assume a linear process or a linear measurement. The downside is that our approach will require additional memory and processing power. The estimate of the state at sample time k is denoted by $\hat{x}_k$, and the associated error covariance is denoted by P_k. Camera data received at time k is denoted by $\mathcal{C}_k$, and inertial sensor data (rate gyros, accelerometers, airspeed, altitude) measured at time k is denoted by $\mathcal{S}_k$. If inertial data is used to drive the prediction phase of the EKF, and camera data is used for the correction step, and both inertial data and camera data is received at every time step, then the standard EKF algorithm is shown in Algorithm 3.

Algorithm 3: Standard EKF Algorithm

```
Input: Inertial data $\mathcal{S}_k$, camera data $\mathcal{C}_k$, state estimate $\hat{x}_{k-1}$, error covariance
       $P_{k-1}$
Output: State estimate $\hat{x}_k$, error covariance $P_k$

    $[\hat{\xi}_k, \Pi_k] = \text{ekf\_predict}(\hat{x}_{k-1}, P_{k-1}, \mathcal{S}_k)$
    $[\hat{x}_k, P_k] = \text{ekf\_correct}(\hat{\xi}_k, \Pi_k, \mathcal{C}_k)$
```

For simplicity of exposition we will make the following assumptions which can be relaxed at the expense of additional bookkeeping:

- The camera data is received in order, i.e., $\mathcal{C}_k$ is received after $\mathcal{C}_{k-1}$.
- The delay in the camera data d is a constant integer number of samples, i.e., while $\mathcal{C}_k$ is received at sample time k, it is valid (image was captured) at sample time $k - d$.

The basic idea of our OOSM-EKF is to retain in memory the state estimate and error covariance over d samples. The revised algorithms is listed in Algorithm 4. When a camera measurement is received at sample k, it is d samples old. Therefore, to properly incorporate the data into the estimator, it must be fused with the estimate for sample $k-d$. This is accomplished in Algorithm 4, Line 1. The old state estimates between samples $k-d$ and $k-1$, which have been used for real-time feedback and were produced via Algorithm 4, Line 4, are ignored, and the inertial data S_j, $j \in \{k-d, k\}$ which has been stored in memory, is used to propagate the state estimate forward to sample k using Algorithm 4, Lines 2 and 3.

The major advantage of Algorithm 4 is that the delayed camera data is incorporated into the estimation process in the same way that it would be if the information

```
Algorithm 4: EKF algorithm with delayed camera data
  Input: Current inertial data $\mathcal{S}_k$ (valid at sample $k$), stored inertial data $\mathcal{S}_j$
         (valid at sample $j$), $j \in \{k-d, \dots, k\}$, camera data $\mathcal{C}_k$ (valid at sample
         $k-d$), state estimate $\hat{x}_{k-d}$ and $\hat{x}_{k-1}$, error covariance $P_{k-d}$ and $P_{k-1}$.
  Output: State estimate $\hat{x}_k$, error covariance $P_k$
  if Camera measurement received at sample k then
1     $\left[\hat{\xi}_{k-d}, \Pi_{k-d}\right] = \text{ekf\_correct}(\hat{x}_{k-d}, P_{k-d}, \mathcal{C}_k)$
      for $j = k-d+1$ to $k$ do
2         $\left[\hat{\xi}_j, \Pi_j\right] = \text{ekf\_predict}(\hat{\xi}_{j-1}, \Pi_{j-1}, \mathcal{S}_j)$
      end
3     $[\hat{x}_k, P_k] = \left[\hat{\xi}_k, \Pi_k\right]$
  else
4     $[\hat{x}_k, P_k] = \text{ekf\_predict}(\hat{x}_{k-1}, P_{k-1}, \mathcal{S}_k)$
  end
```

were not delayed. In other words, when camera information is received at sample k, the resulting estimate $\hat{x}_k$ and error covariance P_k are identical to the estimate and error covariance that would have resulted if the camera information had been received at sample $k-d$. The disadvantages of Algorithm 4 are the increased computational and memory requirements. The inertial data S_j and the old state estimate $\hat{x}_{k-d}$ and error covariance P_{k-d} must be retained in memory. In addition, the prediction step involving the inertial data S_j is performed twice (Lines 2 and 4) for every inertial measurement, and d prediction steps must be performed at sample k.

3.5 Simulation Results

Simulations were conducted in Simulink with a six degree-of-freedom model and full flight dynamics. The optical axis of the simulated camera was parallel to the longitudinal axis of the MAV body frame. The locations of objects in the camera field-of-view were calculated for use in the guidance strategy which uses the locations of objects in the image frame to estimate range-to-obstacle and bearing-to-obstacle as discussed in section 3.4.

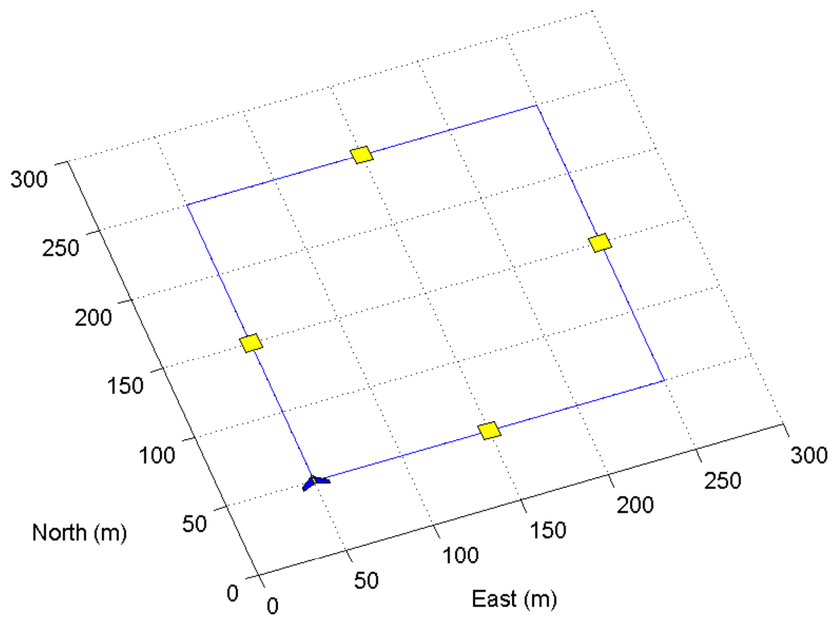


Figure 3.7: The simulation setup is shown. The solid lines are waypoint paths. The squares are obstacles. The MAV starts on the upper right waypoint heading south. The control law must avoid the obstacles and continue along the waypoint path afterward.

3.5.1 Single Obstacles

The obstacle locations for single obstacle avoidance are shown in Figure 3.7. The obstacles are represented as squares. The autopilot uses waypoint path tracking for guidance [38]. The waypoint paths are shown as solid lines in Figure 3.8. The upper right waypoint is the starting location of the MAV. The MAV tracks the waypoints in a clockwise fashion. An obstacle is placed on each waypoint path such that the reactive planner must maneuver around the obstacle or a collision will occur.

The obstacle avoidance strategy overrides the waypoint path tracker when an obstacle is in the camera field of view and within approximately 70 meters of the MAV. The waypoint path tracker resumes control after the obstacle has left the camera field of view. The results of the simulation are shown in Figure 3.8. The path of the MAV is shown by the dashed line. The obstacles are successfully avoided.

The estimates of range-to-obstacle and bearing-to-obstacle without time delay are shown in Figures 3.9-3.10. The range always stays within 0.3 meters of the actual

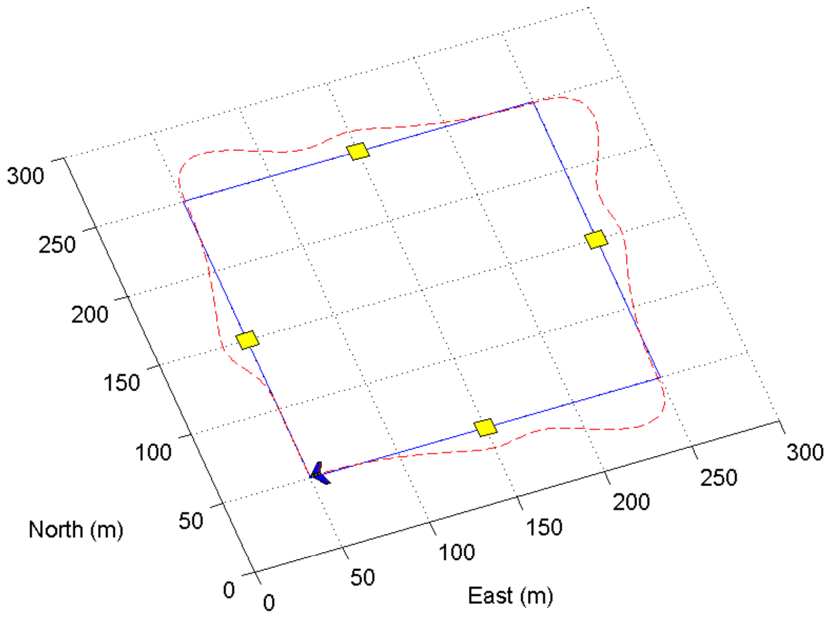


Figure 3.8: The results of a simulation are shown. The dashed line shows the path of the MAV in tracking the waypoint path and avoiding obstacles. The MAV successfully avoids the obstacles.

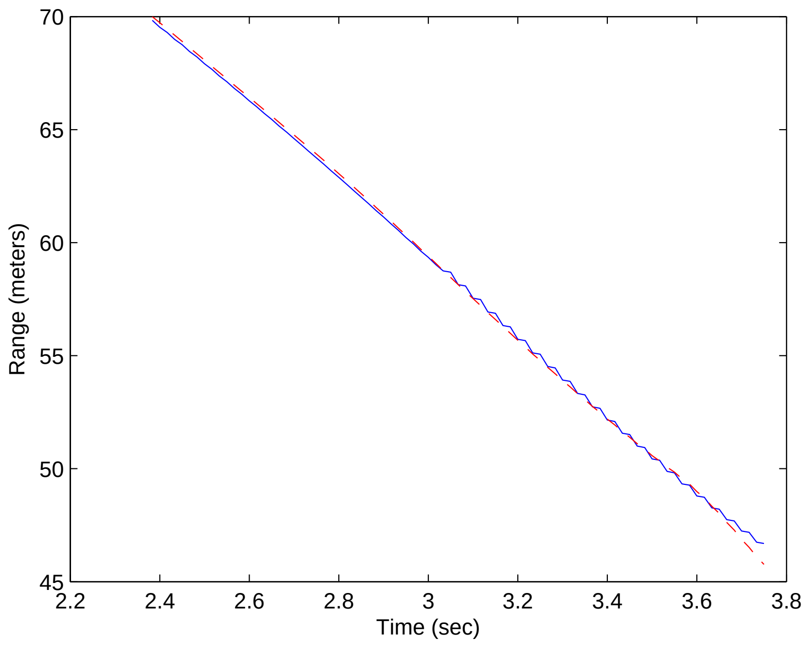


Figure 3.9: The plots for actual range-to-obstacle (solid line) and its estimate (dashed line) while approaching an obstacle are shown.

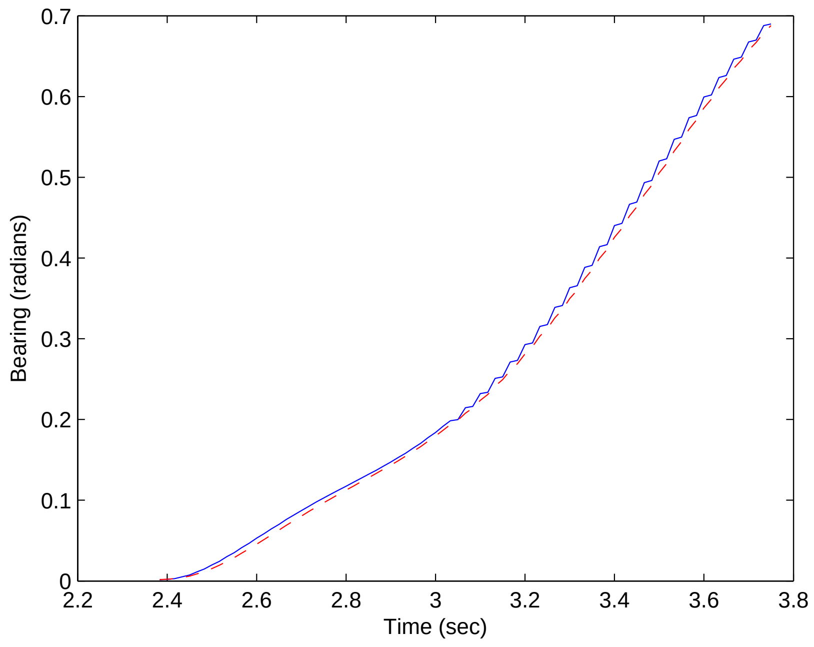


Figure 3.10: The plots for actual bearing-to-obstacle (solid line) and its estimate (dashed line) while approaching an obstacle are shown.

value. The bearing-to-obstacle always stays within 0.02 radians of the actual value. These values are adequate to avoid obstacles in flight.

The range-to-obstacle and bearing-to-obstacle estimates under the effects of a 0.33 second time delay, but without the time delay EKF, are shown in Figures 3.11-3.12. The purpose of these figures is to show how time delay can affect range and bearing estimates. The error in range increases to 1 meter and the bearing to 0.3 radians.

The results of the time delay EKF are shown in Figures 3.13-3.14. The time delay is set to 0.33 seconds. The range-to-obstacle shows a decrease in error down to 0.4 meters as a result of the time delay EKF. The bearing-to-obstacle error decreases to 0.1 radians.

3.5.2 Multiple Obstacles

The multiple obstacle guidance strategy uses two gains, k_2 to balance the obstacles to the edges of the image plane, and k_1 to drive $\rho_d \rightarrow 0$. The gain k_1 is

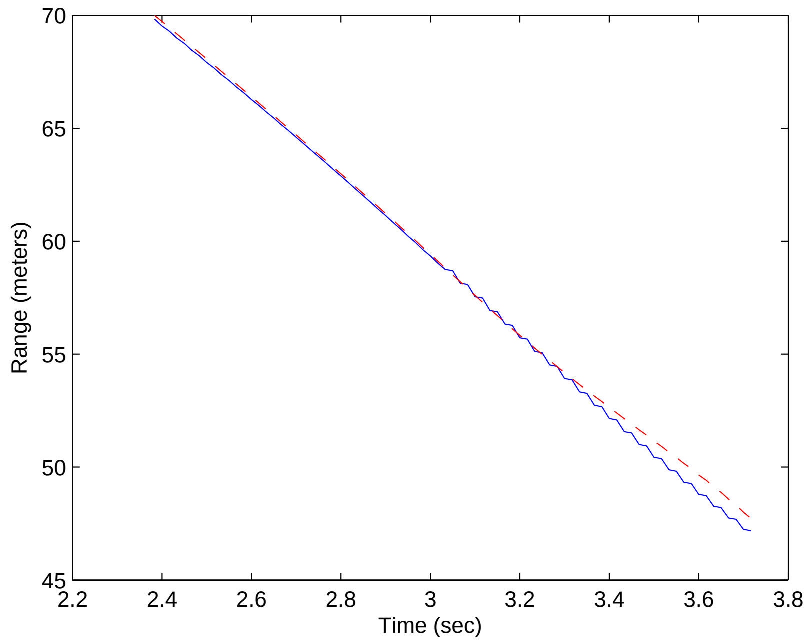


Figure 3.11: The plots for actual range-to-obstacle (solid line) and its estimate (dashed line) with time delay, but without the time delay EKF, while approaching an obstacle are shown.

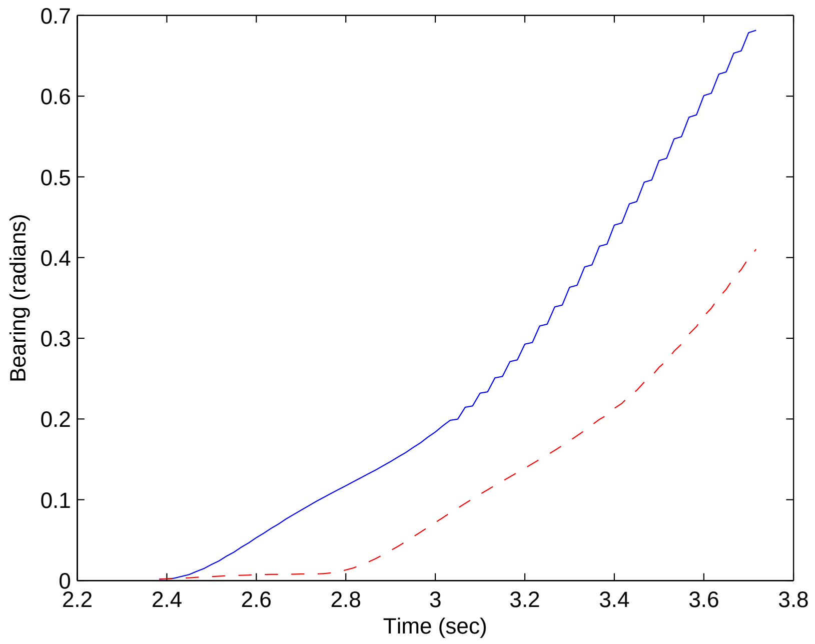


Figure 3.12: The plots for actual bearing-to-obstacle (solid line) and its estimate (dashed line) with time delay, but without the time delay EKF, while approaching an obstacle are shown.

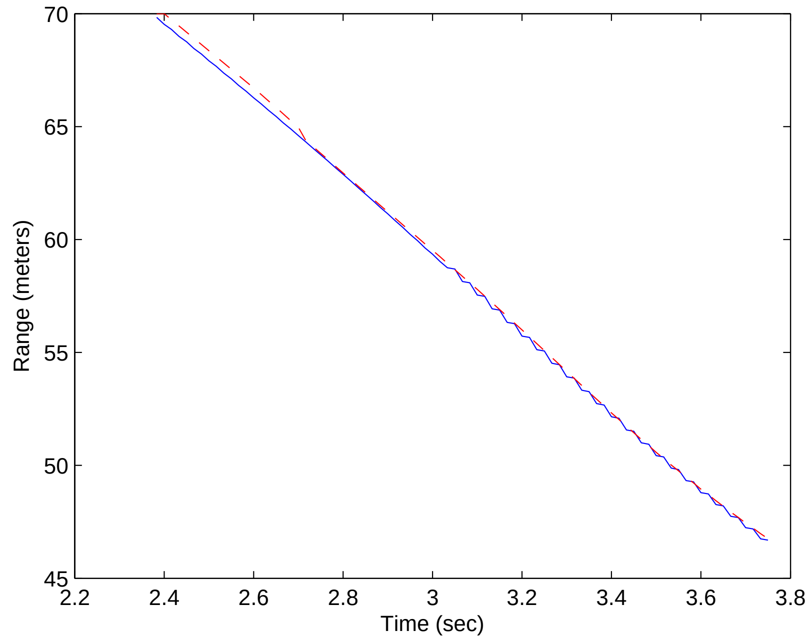


Figure 3.13: The time delay EKF estimates for range-to-obstacle (dashed line) are compared to the actual range values (solid line).

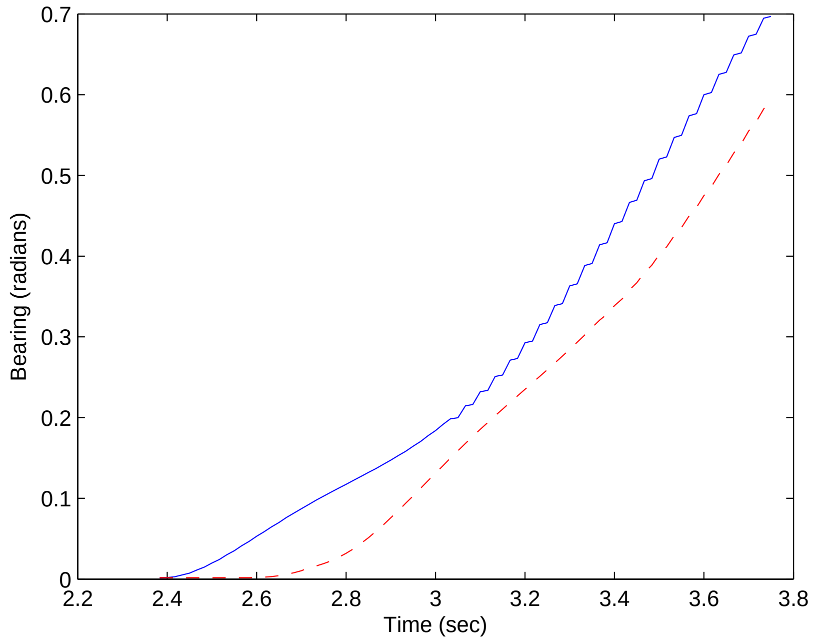


Figure 3.14: The time delay EKF estimates for bearing-to-obstacle (dashed line) are compared to the actual range values (solid line).

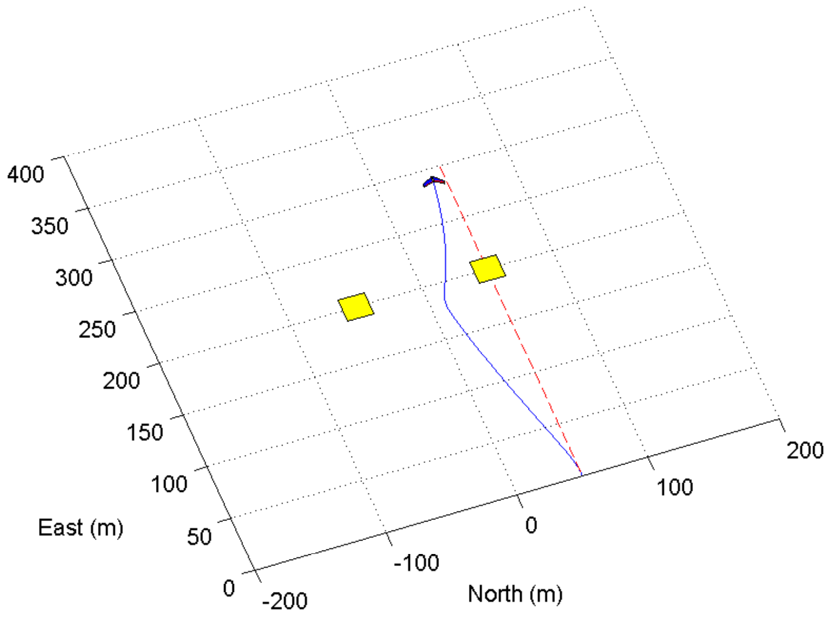


Figure 3.15: The guidance strategy is flown in simulation with $k_1 = .001$. The MAV does not approach the perpendicular bisector strongly enough.

tuned to prevent overshoot while still converging in a reasonable time. The gain k_2 balances driving $\eta_m \to 0$ and $\rho_d \to 0$. If either gain is too high, obstacles will go out of the camera field-of-view as the MAV maneuvers, and the result will be an oscillation as the obstacles move in and out of the camera field of view. The gains need balance to allow the guidance law to maneuver away from the obstacles if $|\rho_d| \gg 0$.

Simulations for $k_1 = 0.001$, $k_1 = 0.01$, and $k_1 = 0.02$ are shown in figures 3.15, 3.16, and 3.17 respectively. The waypoint path is shown by a dotted line while the resulting trajectory is shown by a solid line. In Figure 3.15 the trajectory only slightly approaches the perpendicular bisector of the obstacles implying ρ_d is not driven to zero. Figure 3.16 shows a tendency toward the perpendicular bisector of the obstacles. Figure 3.17 shows a strong tendency toward the perpendicular bisector. The gain $k_2 = 0.01$ was chosen for flight tests.

The purpose of driving $\rho_d \to 0$ is to maneuver the MAV away from obstacles when the approach to the obstacles is at an acute angle. This was tested in simulation and is shown in Figure 3.18. If the guidance strategy were to only drive $\eta_m \to 0$, the

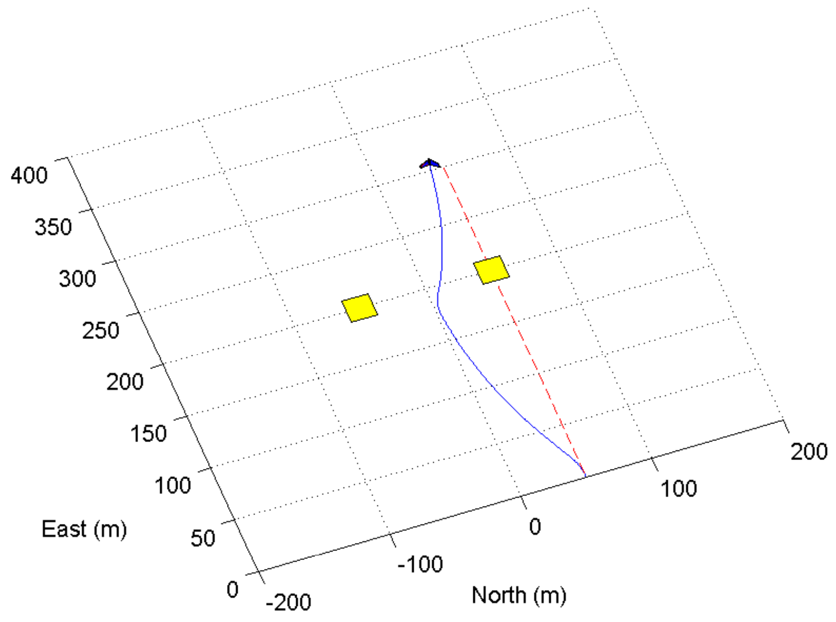


Figure 3.16: The guidance strategy is flown in simulation with $k_1 = .01$. The MAV approaches the perpendicular bisector slightly.

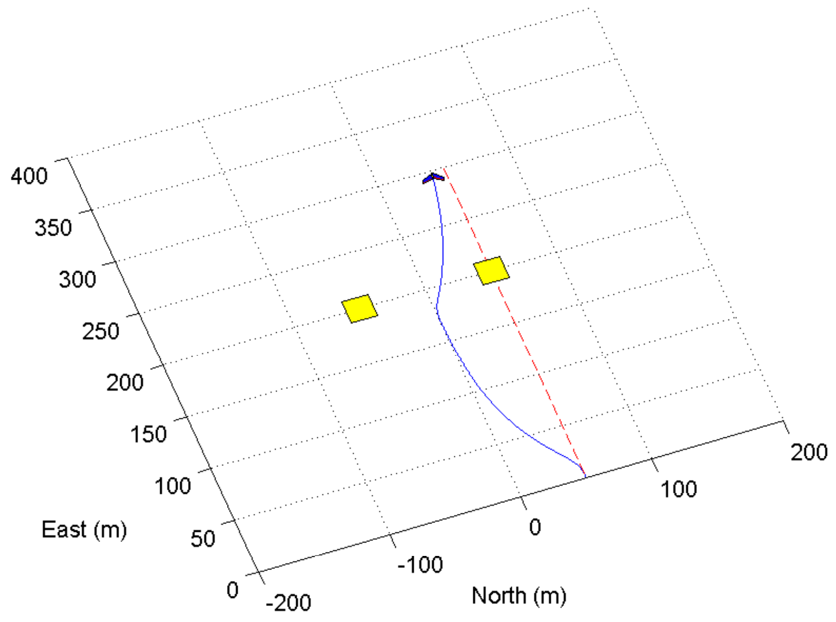


Figure 3.17: The guidance strategy is flown in simulation with $k_1 = .02$. The MAV strongly approaches the perpendicular bisector.

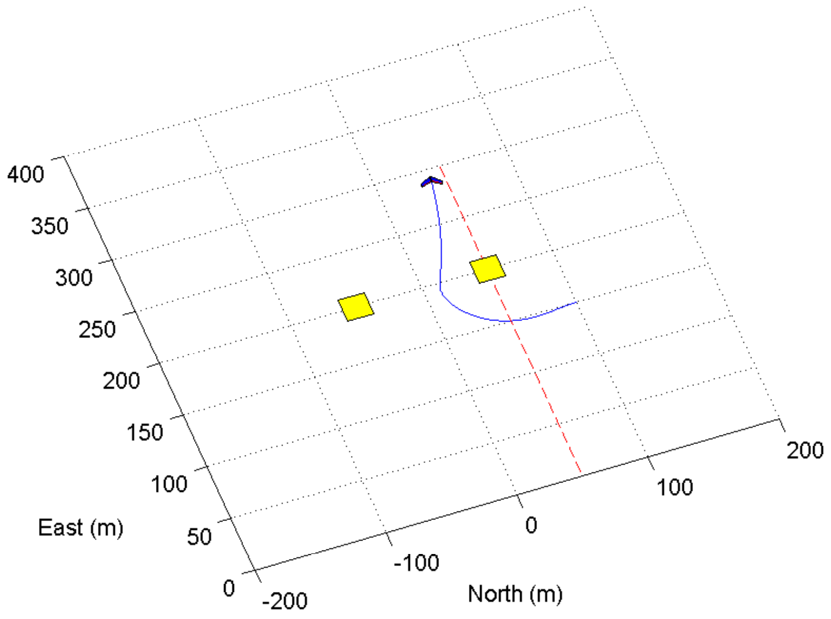


Figure 3.18: The guidance strategy is flown in simulation with $k_1 = .01$ with an approach to obstacles that must drive $\rho_r - \rho_l \rightarrow 0$. The MAV approaches the perpendicular bisector and returns to the waypoint path after passing the obstacles.

MAV would collide with the edges of the obstacle. Alternatively when both $\eta_m \rightarrow 0$ and $\rho_d \rightarrow 0$ simultaneously, the MAV maneuvers toward the bisector of the obstacles. The simulation demonstrates the effectiveness of this approach.

To test the effectiveness of the 3D algorithm, we simulated a city with obstacles where changing altitude to avoid obstacles is preferable to changing heading. The initial conditions of the simulation were set such that the waypoint path went through an overpass. To avoid collision, the UAV would have to change altitude below the underpass and avoid the trees and pillars before continuing on the waypoint path. The results of the simulation are shown in Figure 3.19. The spheres represent the trajectory of the MAV and the cubes represent the waypoint path. The MAV successfully navigates underneath the underpass, around the trees and pillars, and emerges on the other side without collision to continue along the waypoint path.

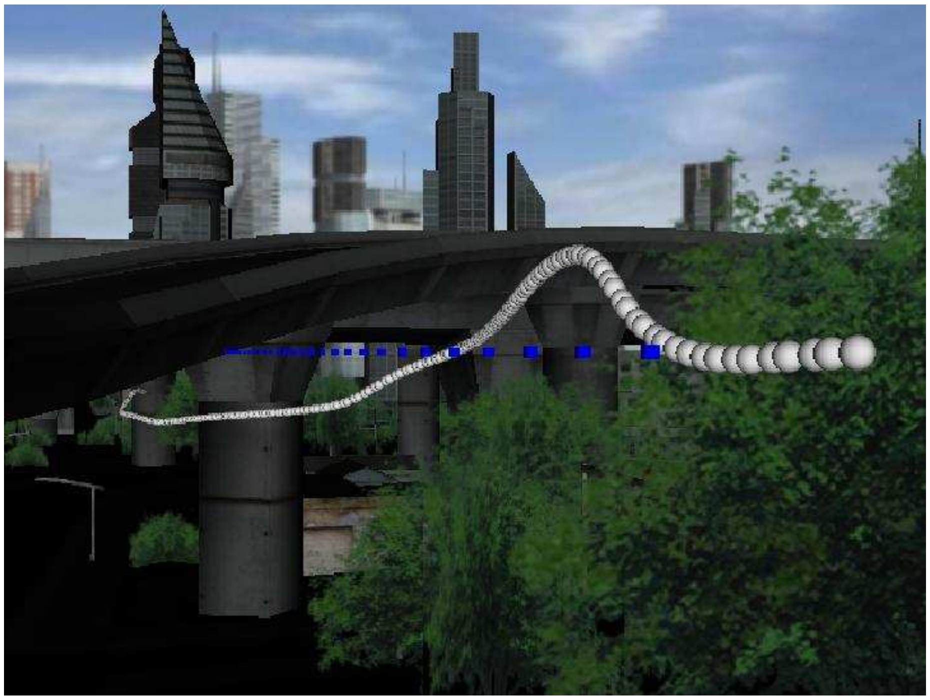


Figure 3.19: The 3D avoidance algorithm guides the UAV below an underpass and around trees to avoid collision in an urban environment. The cubes represent the waypoint path and the spheres represent the trajectory of the MAV.

3.6 Flight Tests

Flight tests were conducted using an air vehicle with a wing span of 48 inches and two elevon control surfaces, as shown in Figure 3.21. The Kestrel autopilot from Procerus Technologies [39], shown in Figure 3.22, navigated the MAV. The guidance strategy in implemented in MATLAB on the ground station and roll commands are transmitted to the autopilot. The camera is mounted parallel to the longitudinal axis of the MAV. For obstacles, we used red cloth on a PVC pipe frame as shown in Figure 3.20. Bearing and time-to-collision of the obstacles is estimated using color segmentation.

3.6.1 Single Obstacles

The path of the MAV in flight avoiding a single obstacle is shown in Figure 3.24. The waypoint path is represented by the solid line, the obstacle by the square, and

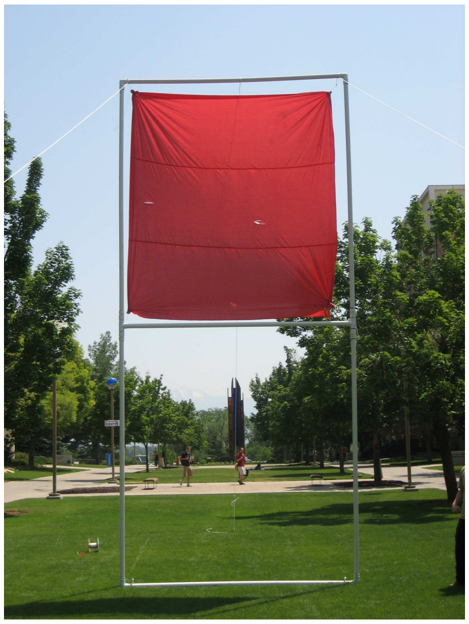


Figure 3.20: The obstacle is an elevated red banner. Color segmentation finds the obstacle in the image plane.

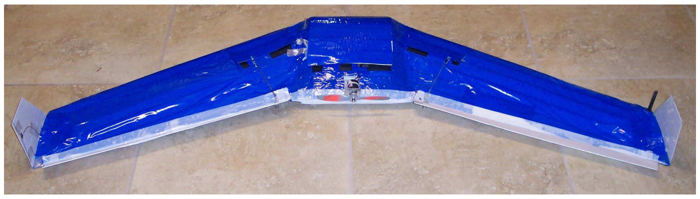


Figure 3.21: The MAV has a wing span of approximately 48 inches and uses the Kestrel autopilot.

Figure 3.22: The Kestrel autopilot onboard the MAV is compared to the size of a quarter.

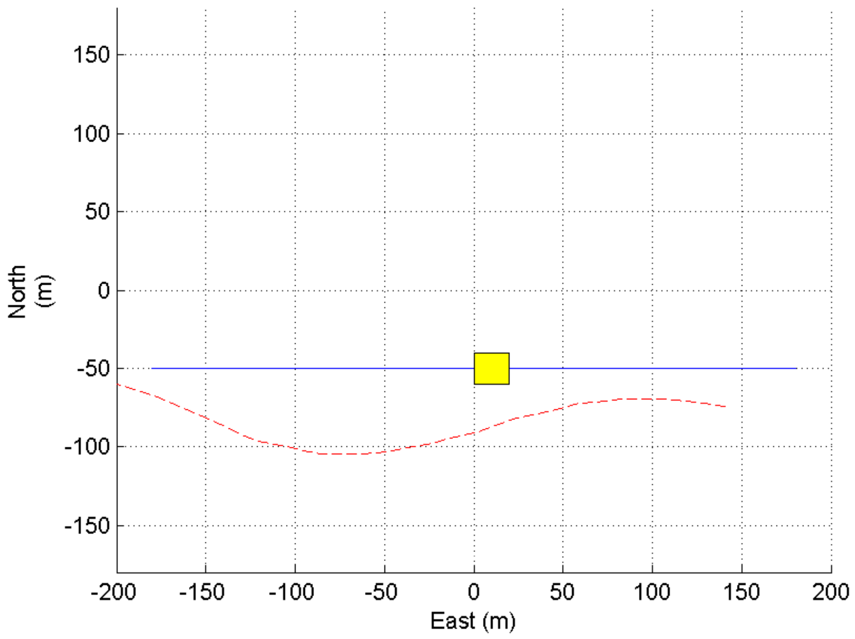


Figure 3.23: The path of the MAV in flight tests. The solid line is the waypoint path and the dashed line is the path of the MAV in flight. The yellow box is the obstacle. The MAV avoids collision with the obstacle and continues along the waypoint path.

the flight path by the dashed line. The MAV avoided the obstacle by approximately 25 meters and continued along the waypoint path.

The range-to-obstacle and bearing-to-obstacle estimates compared with GPS truth data are shown in Figure 3.24-3.25. The time delay is approximated at 0.1 seconds. The range estimate error is within 5 meters of the true value and the bearing estimate error is within 0.01 radians after the filter converges.

3.6.2 Multiple Obstacles

A collision avoidance scenario using two obstacles is shown in figure 3.26. The commanded waypoint path of the MAV is represented by the dashed line and intersects one of the obstacles. The squares represent the obstacles. To avoid collision, the MAV maneuvers away from the waypoint path. The trajectory of the MAV is represented by the solid line. The trajectory deviates from the waypoint path and

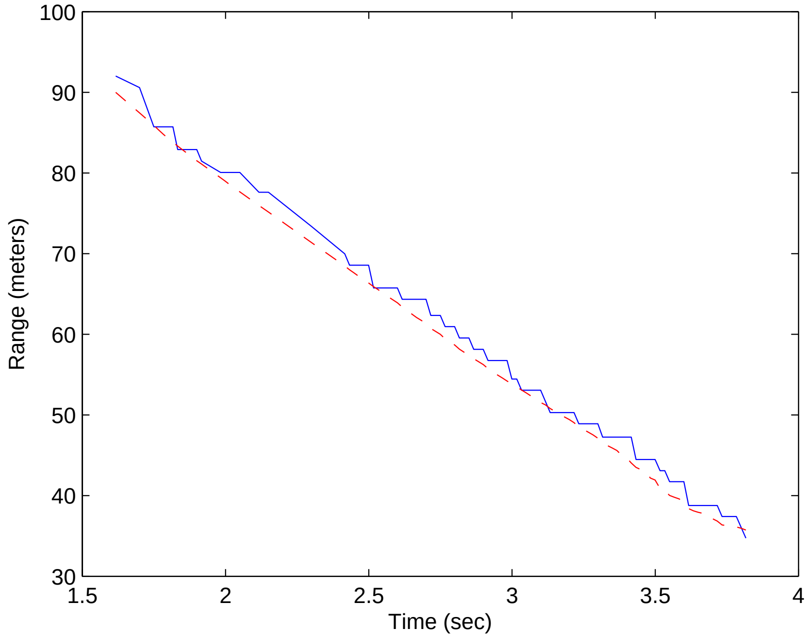


Figure 3.24: Range-to-obstacle estimates are compared with GPS truth data. The dashed line represents the truth data and the solid represents the estimate.

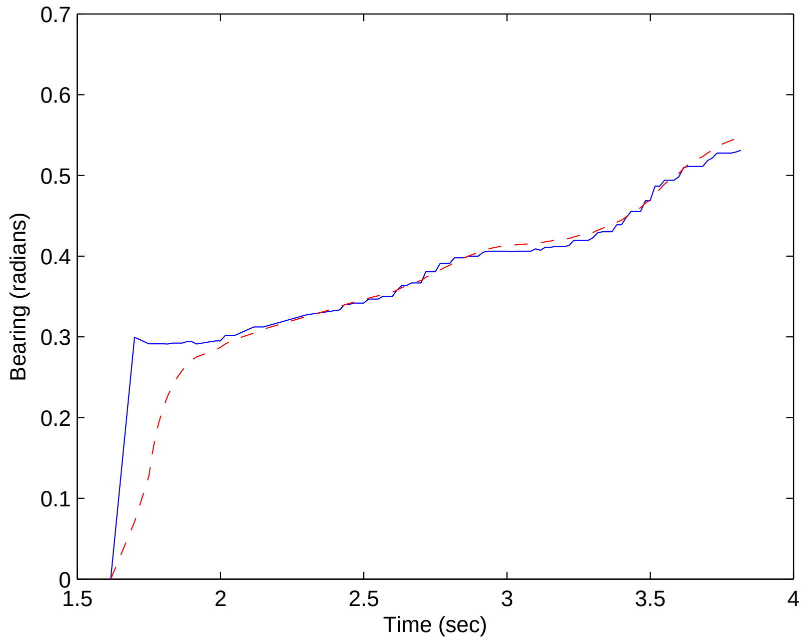


Figure 3.25: Bearing-to-obstacle estimates are compared with GPS truth data. The dashed line represents the truth data and the solid represents the estimate.

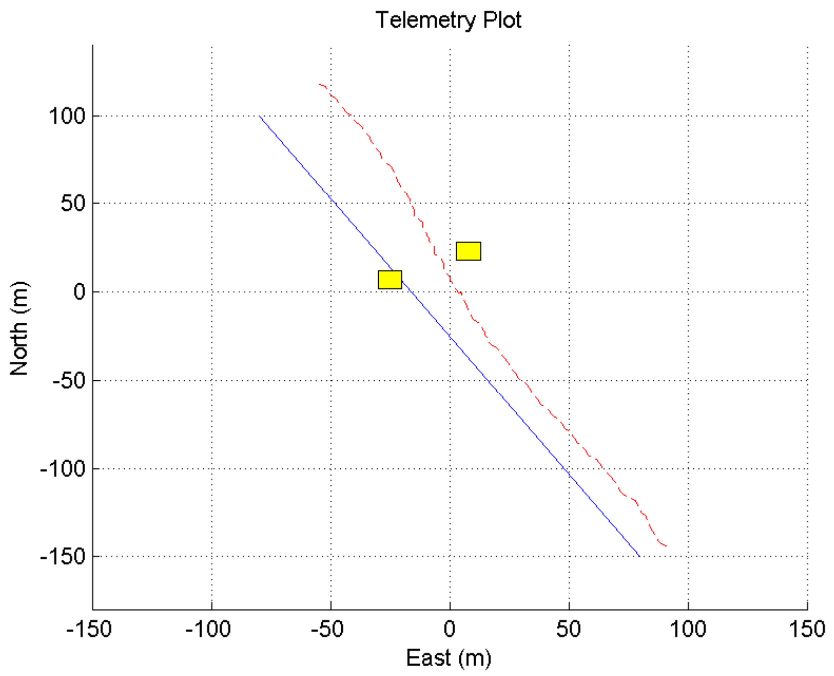


Figure 3.26: The flight path of the MAV avoiding two obstacles is shown. The solid line is the waypoint path and the dashed line is the trajectory of the MAV.

does not intersect either obstacle. The MAV successfully avoids the obstacles in the two passes shown in Figure 3.26.

3.7 Conclusion

In this chapter we have presented two methods for reactive obstacle avoidance by using guidance strategies based on observations made in the image plane. The first guidance strategy pushes obstacles to the edge of the image frame to avoid collision with single obstacles. The second guidance strategy balances obstacles on either side of the image plane, effectively flying between them. When multiple obstacles are present, 2D and 3D algorithms determine between which two obstacles to maneuver. State estimation utilizes obstacle location in the image plane to provide estimates of range-to-obstacle and bearing-to-obstacle. The time delay EKF provides filtered measurements of range-to-obstacles and bearing-to-obstacle. Both strategies are demonstrated in simulation and flight tests.

These are reactionary methods, implying that they override a trajectory tracker in reaction to unknown obstacles. As unknown obstacles are discovered and mapped, a higher level path planner may have time to plan a path around the obstacle as the reactionary planner prevents direct collision. Future work includes integrating the higher level tiers, the path planners, of the proposed obstacle avoidance system described in Section 3.1.

Chapter 4

Visual Tracking in Wind with Field of View Constraints

4.1 Introduction

Current autopilots are as small as two inches square, weigh less than an ounce, and may be installed on aircraft with wingspans of less than twelve inches [40]. Small aircraft using onboard cameras can avoid detection because they have a weak audio signature and because the passive camera does not emit detectable energy. This chapter focuses on autonomous vision-based *target tracking*, defined here as maintaining a target in the field-of-view of an onboard camera with sufficient resolution for visual detection.

Target tracking is an enabling technology for a wide range of potential military and civilian uses. For example, enemy military movements can be automatically tracked by a MAV. The Coast Guard can send a MAV to inspect and track unidentified watercraft. Police can follow suspects without the need for ground vehicles.

Vision-based target tracking is difficult for several reasons. Processing visual data often requires computational resources that are not available on small MAVs, thus requiring a nearby ground station to acquire and process video. Transmitted video is often noisy and intermittent because of transmission interference as well as camera jitter resulting in dropped frames and thus in noise in the tracked position of the target in the image frame. Therefore the tracking method must be robust to target position error introduced by the camera. Another difficulty is that the target may move outside the camera field-of-view, either because of camera motion caused by gusts of wind, or because the target moves. This implies that the tracking method must respond quickly to motion in the image plane.

Previous work has addressed some of the problems described in the previous paragraph. For example, Thomasson derived a solution to target tracking in wind [17] by assuming an elevation controlled camera and constant wind. He found that the air vehicle must fly an ellipse with the target located at one of its foci and with the major axis of the ellipse aligned perpendicular to the direction of the wind. The calculation of the dimensions of the ellipse is based on prior knowledge of the magnitude and direction of the wind velocity. Given the kinematic constraints of the MAV, the wind velocity, and the target velocity, an elliptical trajectory is generated for the MAV to track. If the wind velocity and target velocity remain constant, the MAV will track the target indefinitely, but since the method is open-loop, it will not be robust to changes in wind, gusts, or target motion. This chapter improves on [17] by using a feedback method that still results in elliptical orbits around the target.

Successful path planning feedback solutions to target tracking problems have been demonstrated in [11–16,34,65]. The general approach is to generate paths based on current knowledge. As information is acquired, whether about the wind, target, or MAV, the path to track the target is regenerated. Reference [65] generates paths to a landing position based on vision of a runway or other landmark. Moving targets are tracked using dynamic planning in [11–16] and use gimballed cameras to help track targets. Unfortunately, gimbals are not always feasible on small MAVs. Removing the gimbal introduces additional kinematic constraints and path planning becomes more difficult. Additionally, a gust of wind or a sudden change in the course of the target may push it outside the field-of-view before a response is generated, possibly causing the MAV to lose the target altogether. Fast response is necessary to prevent the target from leaving the image plane. This chapter differs from previous work by using visual feedback to maneuver the MAV so that the target remains in the camera field-of-view.

A feedback control law is developed in [66] for target tracking from kinematic laws similar to those developed in this chapter. The authors assume known target velocity, zero wind, and a gimballed camera. The flight paths in [66] were circular trajectories centered around the target. Circular, rather than elliptical, trajectories are possible because of the degrees of freedom provided by the gimbal. However, the gimbal adds weight and may not be feasible on small MAVs. We build on these results by developing a feedback law around the image plane and by removing the gimbal.

An adaptive feedback approach to target tracking using computer vision is developed in [67, 68]. A fixed angle camera is used to track a target and an adaptive scheme estimates the target velocity and other parameters. This approach had good success in simulation results. Adaptive control however often suffers from transients that affect initial flight performance in hardware tests.

A common application of target tracking is *target geo-location*, which refers to the process of inferring the inertial location of a target from the pose of the sensor and the location of the target in the sensor frame [69, 70]. While target geo-location provides a frame of reference for the guidance algorithm, target location error is introduced from error in the estimated pose of the sensor and error in the estimated target location in the sensor frame. If the guidance strategy is based on target movement in the sensor frame rather than the inertial frame, the resulting system will be more robust.

In this chapter a non-linear control law is developed using range-to-target and bearing-to-target information obtained from target motion in the image plane. This approach maintains the target in the image plane by providing quick reaction to target movement or gusts of wind. As the target moves in the image plane, the control law "pushes" the target back to the center of the camera field-of-view. An extended Kalman filter (EKF) reduces noise in the estimates. The resulting trajectory in wind as a result of the guidance law is an elliptical orbit. The goal is to maintain the target in the camera field-of-view in the presence of wind and target motion. The results of this chapter have been published in [71].

4.2 Problem Description and System Dynamics

4.2.1 Inertial Dynamics

The MAV is assumed to have an autopilot with inner control loops to command roll angle, pitch angle, airspeed, and altitude. The constant altitude kinematic model is therefore given by

$$\dot{n} = V_a \cos\psi + V_w \cos\chi_w, \tag{4.1}$$

$$\dot{e} = V_a \sin\psi + V_w \sin\chi_w, \tag{4.2}$$

$$\dot{\psi} = \frac{g}{V_a} \tan\phi, \tag{4.3}$$

$$\dot{V}_a = \alpha_V (V_a^c - V_a), \tag{4.4}$$

$$\dot{\phi} = \alpha_\phi (\phi^c - \phi), \tag{4.5}$$

where $(n, e)^T$ is the North-East position of the aircraft, V_w, V_a, V_a^c, ϕ, and ψ are the wind speed, airspeed, commanded airspeed, roll, and heading angles, χ_w is the wind angle, g is the gravitational constant, and α_ϕ and α_V are positive autopilot parameters. Assuming that the controlled roll dynamics are significantly faster than the heading dynamics (a valid assumption for the MAVs that we use), we let

$$\phi^c = \arctan\left(\frac{\dot{\psi} V_a}{g}\right), \tag{4.6}$$

and think of $\dot{\psi}$ as the commanded input.

The MAV is equipped with a camera pointed out of the right wing allowing it to persistently orbit a target in the camera field-of-view. The camera is not gimballed and does not move during flight. We assume that vision processing software is available to track the location of the target in the image plane.

Tracking moving targets in wind is difficult because the crab angle caused by wind may move the target out of the camera field-of-view. Currently available sensors on MAVs do not allow accurate measurement of the crab angle. GPS sensors allow for measurement of the course angle. The idea pursued in this chapter is to use the location of the target in the image plane to track the target. If motion in the image plane is translated to motion in the aircraft body frame, then the target can be maintained in the center of the image plane, thus removing the requirement to measure the crab angle and removing possible error from the transformation to the world frame.

4.2.2 Relative Dynamics

The relative dynamics are best described by polar coordinates in the MAV body frame as shown in Figure 4.1. Let ρ be the range-to-target and let η be the

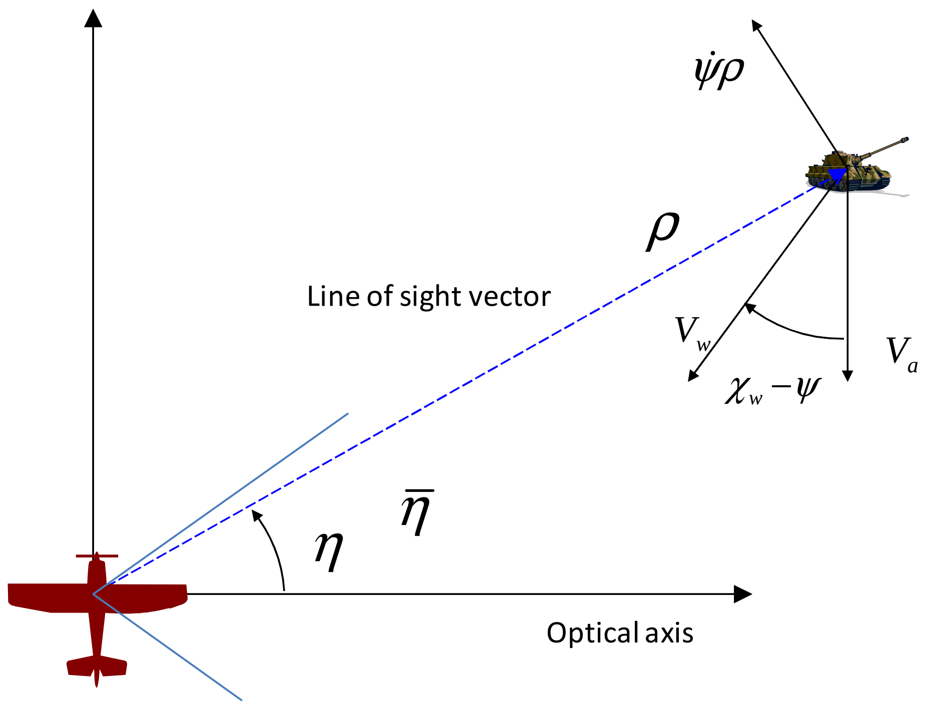


Figure 4.1: Dynamics of the target are viewed from the MAV body frame.

bearing-to-target as measured from the optical axis. The equations of motion are

$$\dot{\rho} = -V_a \sin\eta - V_w \sin(\eta + \chi_w - \psi), \tag{4.7}$$

$$\dot{\eta} = \dot{\psi} - \frac{V_a}{\rho}\cos\eta - \frac{V_w}{\rho}\cos(\eta + \chi_w - \psi), \tag{4.8}$$

where $\dot{\psi}$ is the input.

4.2.3 Camera Geometry

The geometry for the lateral field-of-view of the camera is shown in Figure 4.1. Since the camera is pointed out the right wing, the azimuth angle of the target in the image is given by the state variable η. Let $\bar{\eta}$ be the limit, due to the field-of-view of the camera, on the azimuth angle. Therefore, to maintain the target in the camera field-of-view, we require that

$$|\eta| \leq \bar{\eta}.$$

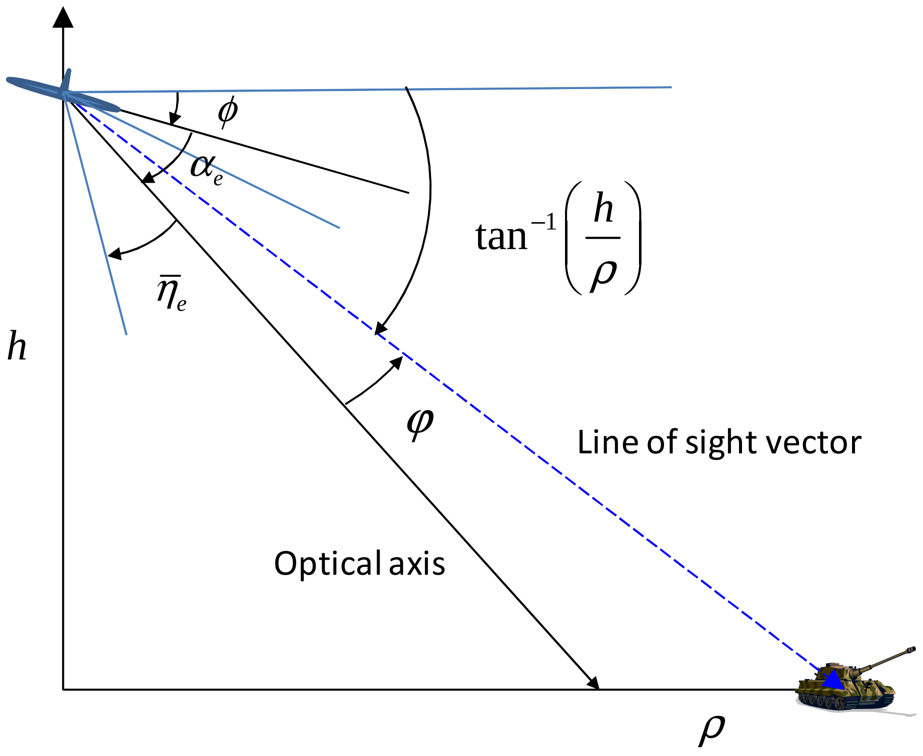


Figure 4.2: Longitudinal view of camera geometry.

The camera geometry in the longitudinal direction is shown in Figure 4.2, where ϕ denotes the roll angle of the MAV, α_e is the (constant) elevation angle of the optical axis relative to the body frame, and $\bar{\eta}_e$ is the field-of-view limit on the elevation angle. The angular deviation of the line of sight vector from the optical axis is given by

$$\varphi = \tan^{-1}\left(\frac{h}{\rho}\right) - \phi - \alpha_e. \tag{4.9}$$

To ensure that the target remains in the camera field-of-view, we require that

$$|\varphi| \leq \bar{\eta}_e.$$

4.2.4 Control Objective

The control objective is to minimize the distance to the target while simultaneously maintaining the target in the camera field-of-view, and to do this within specified roll angle limits. Formally, the control objectives are:

1. If $|\eta(0)| \leq \bar{\eta}$, ensure that $|\eta(t)| \leq \bar{\eta}$ for all $t \geq 0$, where $\bar{\eta}$ is the lateral field-of-view of the camera.
2. If $|\varphi(0)| \leq \bar{\eta}_e$, ensure that $|\varphi(t)| \leq \bar{\eta}_e$ for all $t \geq 0$, where $\bar{\eta}_e$ is the longitudinal field-of-view of the camera.
3. Ensure that $|\phi(t)| \leq \bar{\phi}$ where $\bar{\phi}$ is the maximum allowable roll angle.
4. Minimize the stand-off distance to the target $\rho(t)$ to maximize the resolution of the image in the camera frame.

4.3 Target Motion in the Image Plane

4.3.1 Maintaining the Target in the Lateral Field-of-View

To derive a strategy to maintain the target in the camera field-of-view, consider the Lyapunov function candidate

$$W_1 = \frac{1}{2}\eta^2.$$

Differentiating W_1 along solutions of (4.8) gives

$$\dot{W}_1 = \eta\left(\dot{\psi} - \frac{V_a}{\rho}\cos\eta - \frac{V_w}{\rho}\cos(\eta + \chi_w - \psi)\right).$$

Selecting the heading rate as

$$\dot{\psi} = \frac{V_a}{\rho}\cos\eta + \frac{V_w}{\rho}\cos(\eta + \chi_w - \psi) - k_1\eta + \nu, \tag{4.10}$$

results in

$$\begin{aligned} \dot{W}_1 &= -k_1\eta^2 + \eta\nu \\ &\leq -k_1\eta^2 + |\eta|\,|\nu| \\ &= -(1-\theta)k_1\eta^2 + \left[|\eta|\,|\nu| - \theta k_1\eta^2\right], \end{aligned}$$

where $\theta \in (0,1)$ is a selectable gain. Therefore, $\dot{W}_1$ is negative definite if $|\nu| < k_1\theta\,|\eta|$, resulting in the following theorem.

Theorem 4.3.1 *If $\dot{\psi}$ is given by Equation* (4.10)*, where ν satisfies*

$$|\nu| < k_1\theta\,|\eta|, \tag{4.11}$$

for some $\theta \in (0,1)$, then $|\eta(0)| \leq \bar{\eta}$ implies that $|\eta(t)| \leq \bar{\eta}$.

Proof: Since $\dot{W}_1(\bar{\eta})$ and $\dot{W}_1(-\bar{\eta})$ are both negative, the set $\{-\bar{\eta} \leq \eta \leq \bar{\eta}\}$ is positively invariant. ■

The signal ν will be used later in this section to minimize $\rho(t)$.

4.3.2 Minimizing the Range-to-Target

The signal ν can be used to minimize the average value of ρ subject to constraint (4.11). In this section we will derive two possible strategies for selecting ν. The first strategy is based on a continuous time derivation. Toward that end, let

$$W_2 = \frac{1}{2}(\rho + \lambda\dot{\rho})^2,$$

where $\lambda > 0$. Minimizing W_2 will minimize the deviation of ρ from the origin. Differentiating W_2 gives

$$\dot{W}_2(\nu) = (\rho + \lambda\dot{\rho})(\dot{\rho} + \lambda\ddot{\rho}(\nu)),$$

where

$$\ddot{\rho}(\nu) = (-V_a \cos\eta)\nu + \left[k_1 V_a \eta \cos\eta + \frac{V_w V_a}{\rho}\cos\eta\cos(\eta + \chi_w - \psi) \right. \tag{4.12}$$
$$\left. + \frac{V_w^2}{\rho}\cos^2(\eta + \chi_w - \psi)\right], \tag{4.13}$$

is obtained by differentiating Equation (4.7) and substituting Equation (4.8). Therefore ν can be selected as

$$\nu^* = \arg\min_{0 \le |\nu| \le k_1\theta\eta} \dot{W}_2(\nu), \tag{4.14}$$

where $\theta \in (0, 1)$.

The second strategy for picking ν is based on a discrete time version of the dynamics. Substituting Equation (4.10) into Equation (4.8) gives

$$\dot{\eta} = -k_1\eta + \nu. \tag{4.15}$$

Given a sample rate T, the sampled-data version of (4.15) is

$$\eta_{k+1} = e^{-k_1 T}\eta_k + (1 - e^{-k_1 T})\nu_k.$$

Using an Euler approximation, the sampled-data equivalent of (4.12) is

$$\rho_{k+2} = \rho_{k+1} + T\left[-V_a \sin\eta_{k+1} - V_m \sin(\eta_{k+1} + \chi_w - \psi_{k+1})\right],$$

where

$$
\begin{aligned}
\rho_{k+1} &= \rho_k + T\left[-V_a \sin\eta_k - V_m \sin(\eta_k + \chi_w - \psi_k)\right], \\
\psi_{k+1} &= \psi_k + T\left[\frac{V_a}{\rho_k}\cos\eta_k + \right. \\
&\qquad \left. \frac{V_w}{\rho_k}\cos(\eta_k + \chi_w - \psi_k) - k_1\eta_k + \nu_k\right].
\end{aligned}
$$

Therefore, ν_k can be selected as

$$\nu_k^* = \arg\min_{|\nu_k| \leq k_1\theta\eta}\rho_{k+2}.$$

4.3.3 Maintaining the Target in the Longitudinal Field-of-View

When the MAV changes roll angle, the target moves in the longitudinal camera field-of-view. While some movement is expected and unavoidable, we need to find a set of sufficient conditions which guarantee that the target will not leave the camera field-of-view. The conditions for the longitudinal field-of-view are hard to find because the target may leave the camera field-of-view if the initial conditions are extreme, such as when the initial conditions place the target in the corner of the camera field-of-view. One possible method is to solve the differential equations for all initial conditions and to determine which initial conditions do not result in the target leaving the camera field-of-view. However, the differential equations are difficult to solve analytically. An alternative is to simulate a reasonable range of initial conditions to determine a sufficient gain for each set of initial conditions. If the worst-case conditions are simulated, then a sufficient set of gains can be found to maintain the target in the camera field-of-view. In this section, we simulate a set of worst-case conditions to find bounds on the gains to maintain the target in the longitudinal field-of-view.

To simulate a set of worst case conditions, we set $\eta = \bar{\eta} = 40°$, $\bar{\eta}_e = 40°$, $\alpha_e = 30°$, $\bar{\phi} = 30°$, and a constant wind V_w blowing from the MAV to the target as shown in Figure 4.3. The initial distance to target is selected so that the target is on the edge of the longitudinal field-of-view and would leave the field-of-view if the

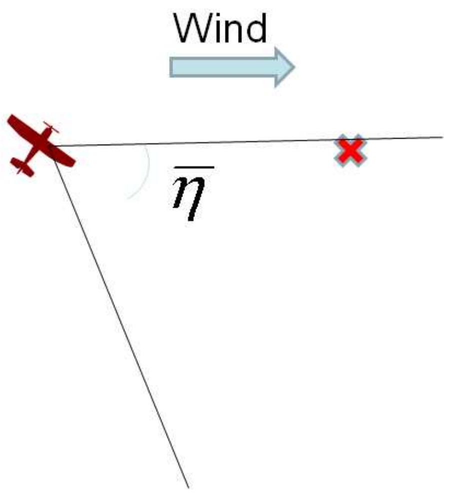


Figure 4.3: The simulation finds the minimum initial distance to the target by using a set of worst case scenarios. The MAV is initialized with $\eta = \bar{\eta}$ and such that the wind direction pushes the MAV towards the target. This scenario is used to find the minimum distance so that the target does not leave the camera field-of-view.

MAV banked to turn away from the target. Each instance of the simulation adds one meter to the initial distance to target until the MAV can successfully converge to an orbit around the target without the target leaving the camera field-of-view. The gain k and wind V_w are varied to find a wide range of conditions. The results are shown in Figure 4.4. To apply the results to a wide range of vehicles, the units are nondimensionalized by dividing by V_a or h as required.

Since the results are close to linear, we can estimate a linear upper bound using a plane. The plane shown in Figure 4.5 was found using least squares estimation and biased above the data points. It estimates a sufficient set of initial conditions to maintain the target in the camera field-of-view. The plane is described as a function of k and V_w by

$$\rho_{0min} > (3.9037)\frac{kh^2}{V_a} - (0.0291)\frac{V_w h}{V_a} + (0.4911). \tag{4.16}$$

Equivalently, the plane can also be described as a function of ρ_{0min} and V_w by

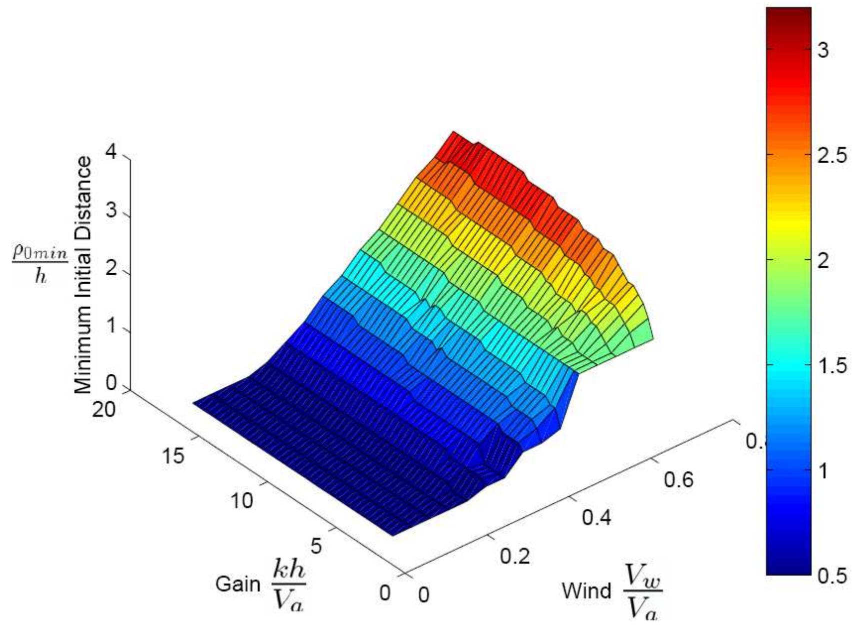


Figure 4.4: Results of simulation experimentation show the gain k given a minimum initial distance from target (ρ_{0min}) and constant wind V_w.

$$0.5 \leq k \leq (0.2562)\frac{\rho_{0min}V_a}{h^2} + (0.0075)\frac{V_w}{h} - (0.1258)\frac{V_a}{h^2}. \tag{4.17}$$

The wind is a constant and must be chosen based on expected environmental conditions. Given the wind speed, the minimum initial distance from the target can be chosen to determine the required gain k.

The same set of simulations determine the smallest distance from target ρ_{min} resulting from the gain k and wind V_w when starting from the initial distance ρ_{0min}. If the MAV starts with an initial distance of ρ_{0min}, then the distance to target will not be less then ρ_{min} during the flight. The results are shown in Figure 4.6. The gain and wind chosen in Equation (4.16) determine the smallest distance to target ρ_{min} in Figure 4.6.

Given the value ρ_{min} determined in simulation, The following theorem guarantees that the target will stay in the longitudinal field-of-view.

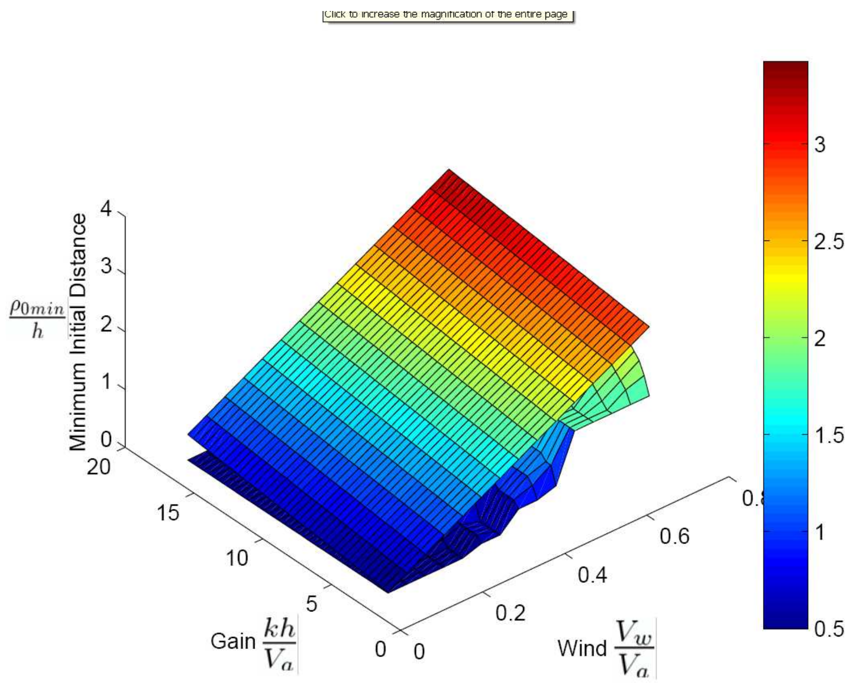


Figure 4.5: A plane estimates an upper bound to the experimental distances. The plane provides a simple mechanism of finding a sufficient k.

Theorem 4.3.2 *If the design constraints*

$$\bar{\phi} \geq \tan^{-1}\left(\frac{h}{\rho_{min}}\right) - \bar{\eta}_e - \alpha_e, \tag{4.18}$$

$$\bar{\phi} \geq -\bar{\eta}_e + \alpha_e, \tag{4.19}$$

$$\frac{g}{V_a}\tan\left(-\bar{\phi}\right) \leq -\frac{V_a\cos\bar{\eta} + V_w}{\rho_{min}} - k_1\left(1-\theta\right)\bar{\eta}, \tag{4.20}$$

$$\frac{g}{V_a}\tan\bar{\phi} \geq \frac{V_a\cos\bar{\eta} + V_w}{\rho_{min}}, \tag{4.21}$$

$$\frac{g}{V_a}\tan\left(\bar{\eta}_e - \alpha_e\right) \geq \frac{V_a\cos\bar{\eta} + V_w}{\rho_{min}}, \tag{4.22}$$

$$\frac{g}{V_a}\tan\left(\tan^{-1}\left(\frac{h}{\rho_{min}}\right) - \bar{\eta}_e - \alpha_e\right) \leq -\frac{V_a\cos\bar{\eta} + V_w}{\rho_{min}} - k_1\left(1-\theta\right)\bar{\eta}, \tag{4.23}$$

are satisfied and

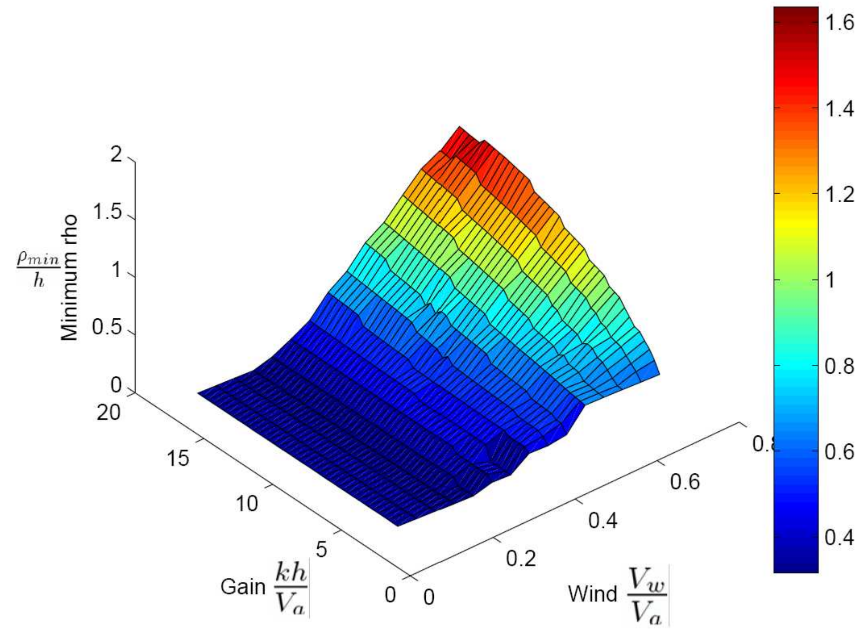


Figure 4.6: Results of simulation experimentation show the smallest distance from target (ρ_{min}) given a gain k and constant wind V_w.

$$\dot{\psi} = \frac{V_a}{\rho}\cos\eta + \frac{V_w}{\rho}\cos(\eta + \chi_w - \psi) - k_1\eta + \nu, \tag{4.24}$$

and Equation (4.16) *is satisfied, then the target will stay in the longitudinal field-of-view.*

Proof:

The upper and lower bounds of the roll angle to maintain the target in the longitudinal field-of-view are found geometrically from Figure 4.2 as

$$\phi_{vup} = \tan^{-1}\left(\frac{h}{\rho}\right) + \bar{\eta}_e - \alpha_e, \tag{4.25}$$

$$\phi_{vlo} = \tan^{-1}\left(\frac{h}{\rho}\right) - \bar{\eta}_e - \alpha_e. \tag{4.26}$$

The camera field-of-view is not the only constraint on the roll angle. The physical roll angle limits also constrain the roll. The physical roll constraints are added to the roll bounds as

$$\phi_{up} = \min\left(\tan^{-1}\left(\frac{h}{\rho}\right) + \bar{\eta}_e - \alpha_e, \bar{\phi}\right), \tag{4.27}$$

$$\phi_{lo} = \max\left(\tan^{-1}\left(\frac{h}{\rho}\right) - \bar{\eta}_e - \alpha_e, -\bar{\phi}\right). \tag{4.28}$$

We first need to show that these constraints are consistent, $\phi_{lo} \leq \phi_{up}$, by separating the limits into four inequalities. The first inequality

$$-\bar{\phi} \leq \bar{\phi}, \tag{4.29}$$

is always true. The second

$$\begin{aligned} \tan^{-1}\left(\frac{h}{\rho}\right) + \bar{\eta}_e - \alpha_e &\geq \tan^{-1}\left(\frac{h}{\rho}\right) - \bar{\eta}_e - \alpha_e \\ \Leftrightarrow \bar{\eta}_e &\geq -\bar{\eta}_e, \end{aligned} \tag{4.30}$$

is also always true. The third inequality

$$\begin{aligned} \bar{\phi} &\geq \tan^{-1}\left(\frac{h}{\rho}\right) - \bar{\eta}_e - \alpha_e \\ \Leftarrow \bar{\phi} &\geq \tan^{-1}\left(\frac{h}{\rho_{min}}\right) - \bar{\eta}_e - \alpha_e, \end{aligned} \tag{4.31}$$

and fourth inequality

$$\begin{aligned} -\bar{\phi} &\leq \tan^{-1}\left(\frac{h}{\rho}\right) + \bar{\eta}_e - \alpha_e \\ \Leftarrow \bar{\phi} &\geq -\bar{\eta}_e + \alpha_e, \end{aligned} \tag{4.32}$$

are design constraints, given by Equations (4.18) and (4.19).

The next step is to constrain the guidance strategy to maintain $\phi_{lo}(t) \leq \phi(t) \leq \phi_{up}(t)$. First find the upper and lower bounds of the control

$$\dot{\psi} = \frac{V_a}{\rho}\cos\eta + \frac{V_w}{\rho}\cos\left(\eta + \chi_w - \psi\right) - k_1\eta + \nu, \tag{4.33}$$

as

$$-\frac{V_a\cos\bar{\eta} + V_w}{\rho_{min}} - k_1\eta + \nu \leq \dot{\psi} \leq \frac{V_a\cos\bar{\eta} + V_w}{\rho_{min}} - k_1\eta + \nu. \tag{4.34}$$

Using the coordinated turn $\dot{\psi} = \frac{g}{V_a}\tan\phi$ assumption to relate the heading rate to the roll angle gives

$$\begin{aligned} -\frac{V_a\cos\bar{\eta} + V_w}{\rho_{min}} - k_1\eta + \nu &\leq \frac{g}{V_a}\tan\phi \\ &\leq \frac{V_a\cos\bar{\eta} + V_w}{\rho_{min}} - k_1\eta + \nu. \end{aligned} \tag{4.35}$$

If $\frac{g}{V_a}\tan\phi_{lo}$ is less than the lower bound, then the target will not move below the bottom of the longitudinal field-of-view or violate the lower physical roll bound. Therefore, we constrain $\frac{g}{V_a}\tan\phi_{lo}$ to be less than the lower bound in (4.35) by setting

$$\frac{g}{V_a}\tan\phi_{lo} \leq -\frac{V_a\cos\bar{\eta} + V_w}{\rho_{min}} - k_1\eta + \nu. \tag{4.36}$$

Using the guidance strategy in Section 4.3.2, substitute $\nu = k_1\theta\eta$ into (4.36) and simplify yielding

$$\frac{g}{V_a}\tan\phi_{lo} \leq -\frac{V_a\cos\bar{\eta} + V_w}{\rho_{min}} - k_1\left(1-\theta\right)\bar{\eta}. \tag{4.37}$$

There are two possible bounds for ϕ_{lo} described in (4.28). Equation (4.37) must satisfy both. To satisfy those bounds, substitute the two limits for ϕ_{lo} from Equation (4.28) into (4.37) which yields the two design constraints (4.19) and (4.23).

Similarly, if $\frac{g}{V_a}\tan\phi_{up}$ is always greater than the upper bound of (4.35), then the target will not move above the top of the longitudinal field-of-view or violate the upper physical roll bound. Constrain $\frac{g}{V_a}\tan\phi_{up}$ to be greater than the upper bound by setting

$$\frac{g}{V_a}\tan\phi_{up} \geq \frac{V_a\cos\bar{\eta} + V_w}{\rho_{min}} - \nu. \tag{4.38}$$

Again, minimize the distance to target by setting $\nu = k_1\theta\eta$ to obtain

$$\frac{g}{V_a}\tan\phi_{up} \geq \frac{V_a\cos\bar{\eta} + V_w}{\rho_{min}} - k_1\eta + k_1\theta\eta. \tag{4.39}$$

The bound can be made tighter by removing the last two terms to get

$$\frac{g}{V_a}\tan\phi_{up} \geq \frac{V_a\cos\bar{\eta} + V_w}{\rho_{min}}. \tag{4.40}$$

There are two possible bounds for ϕ_{up} described in (4.27). Equation (4.40) must satisfy both. To satisfy those bounds, substitute the two limits for ϕ_{up} from Equation (4.27) into (4.40). This results in two more design constraints given in Equations (4.21) and (4.22).

If Equations (4.31), (4.32), (4.20), (4.21), (4.22), and (4.23) are satisfied, then the target will be maintained in the longitudinal field-of-view, i.e. $\phi_{lo}(t) < \phi(t) < \phi_{up}(t)$.

■

4.3.4 Resulting Flight Paths

Ref [17] shows that the path of an air vehicle tracking a target in wind with a roll only camera is an elliptical orbit if $\eta = 0$. To show that our approach produces a similar result, divide (4.7) by (4.10), letting $\eta = \nu = 0$ to get

$$\frac{d\rho}{d\psi} = \rho \frac{-V_w \sin(\chi_w - \psi)}{V_a + V_w \cos(\chi_w - \psi)},$$

which, as pointed out in [14], is an elliptical orbit with eccentricity $\epsilon = \frac{V_w}{V_a}$. One of the advantages of our approach is that rather than forcing the target to be located along the optical axis, the target is allowed (through the selection of ν) to move in the image plane to facilitate more circular orbits in wind. An interesting question is whether circular orbits, where the target remains in the camera field-of-view, are possible in wind. The following theorem provides explicit conditions.

Theorem 4.3.3 *Circular orbits that maintain the target in the field-of-view are possible if*

$$\tan \bar{\eta} \geq \max_{\psi \in [0, 2\pi]} \left| \frac{\frac{V_w}{V_a} \sin(\chi_w + \psi)}{1 + \frac{V_w}{V_a} \cos(\chi_w + \psi)} \right|.$$

Proof: Dividing (4.7) by (4.10) gives

$$\frac{d\rho}{d\psi} = \rho \frac{-V_a \sin \eta - V_w \sin(\eta + \chi_w - \psi)}{V_a \cos \eta + V_w \cos(\eta + \chi_w - \psi) - k_1 \eta + \nu}.$$

When the orbit is circular, $\frac{d\rho}{d\psi} = 0$, or in other words,

$$-V_a \sin \eta - V_w \sin(\eta + \chi_w - \psi) = 0.$$

Solving for $\tan\eta$ gives

$$\tan\eta = -\frac{\frac{V_w}{V_a}\sin(\chi_w + \psi)}{1 + \frac{V_w}{V_a}\cos(\chi_w + \psi)}.$$

Maximizing the right hand side over all possible values of ψ gives the desired result.

■

4.4 Time-Delayed State Estimation

The proposed system utilizes computer vision to track obstacles and provide range estimates to those obstacles. As shown in Figure 4.7, vision processing is implemented on the ground station. As a result, the time delay introduced by sending the video to the ground, processing, and returning the results, may be significant enough to adversely affect the guidance strategy. We address this problem using the time delay EKF previously described in Section 3.4.1.

4.5 Simulation Results

Simulations were conducted in Simulink using a six degree-of-freedom model. We used an emulated Kestrel autopilot [39] with airspeed set to $V_a = 13$ m/s. The location and size of the target in the camera image plane were calculated and delayed to emulate vision processing. The location of the target in the image plane was filtered by the delayed EKF and used by the guidance strategy described in Section 4.3. The system architecture is shown in Figure 4.7.

The results of a stationary target in zero wind are shown in Figure 4.8 (a). Without wind, the expected shape of the orbit is circular. Figure 4.8 (a) demonstrates the flight path of the MAV converging to a circular orbit as predicted. The circular orbit is the result of a guidance loop without a priori calculation of a flight path.

The gain k_1 changes the rate of convergence to the orbit. To examine the effect of k_1 on the convergence rates, the gains were set to $k_1 = 1.5$ in Figure 4.8 (a), $k_1 = 3.0$ in Figure 4.9 (a), and $k_1 = 5.0$ in Figure 4.10 (a). While a larger gain results in faster convergence, it also induces more target movement in the camera frame. The corresponding target motion in the camera frame are shown in Figures 4.8 (b),

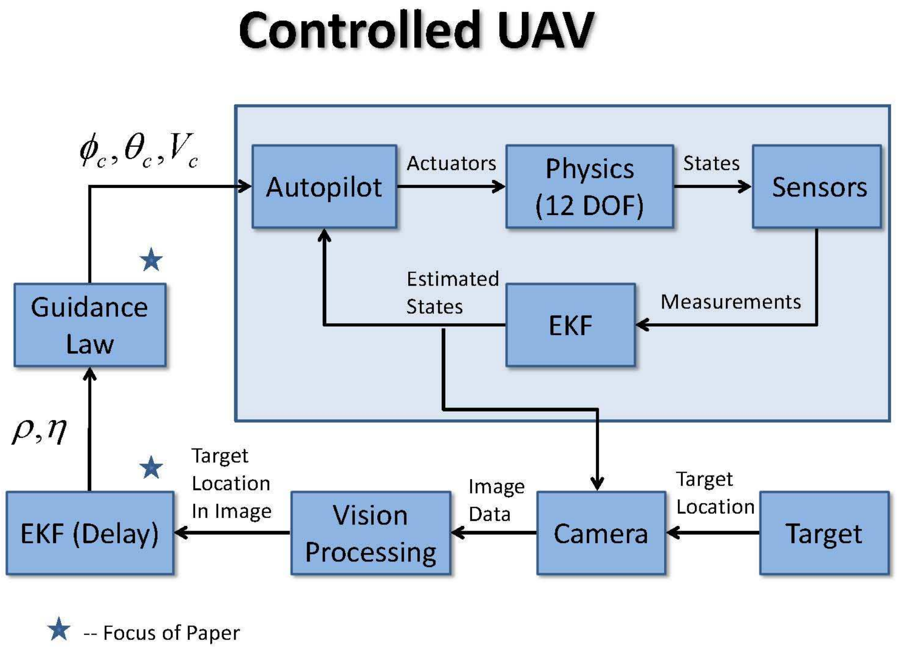


Figure 4.7: A block diagram of the system architecture. The elements in gray are implemented on-board the aircraft, the vision processing, delayed EKF, and guidance law are implemented on the ground station.

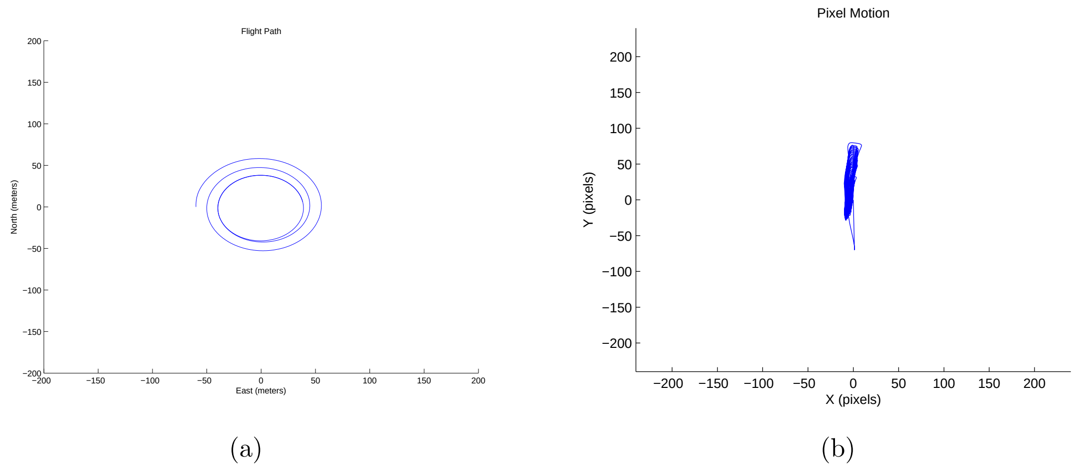
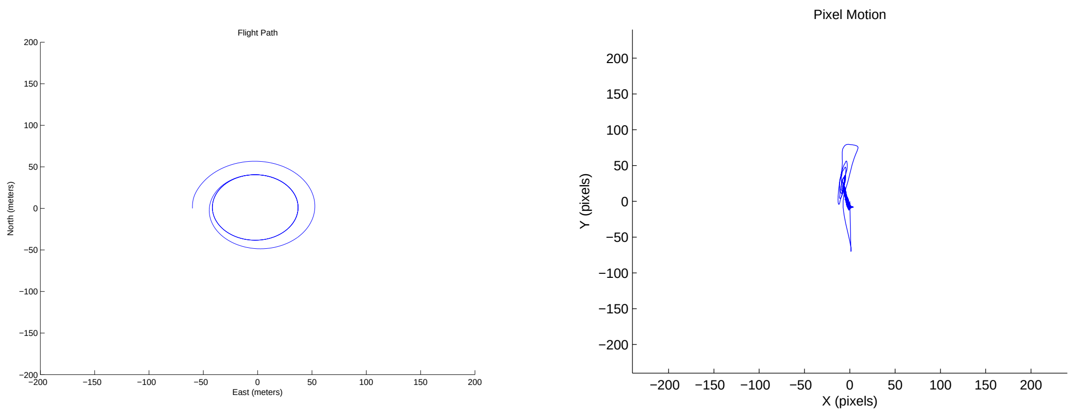
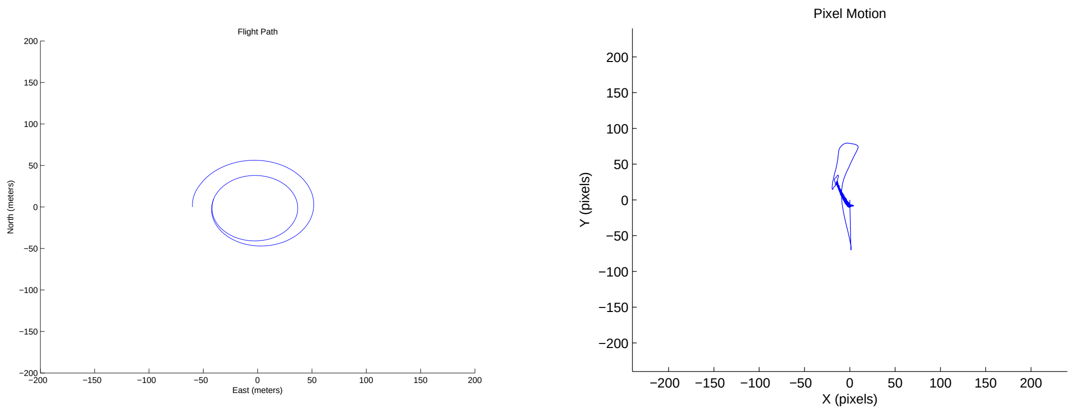


Figure 4.8: (a) Simulated flight path using the guidance strategy in Section 4.3 in zero wind. The MAV's initial configuration allows the camera to capture the target. The MAV navigates onto a stable orbit around the target. The gain is set to $k_1 = 1.5$. (b) The corresponding motion of the target in the image plane.



(a) (b)

Figure 4.9: (a) Simulated flight path in zero wind with gain $k_1 = 3.0$. (b) The corresponding motion of the target in the image plane.



(a) (b)

Figure 4.10: (a) Simulated flight path in zero wind with gain $k_1 = 5.0$. (b) The corresponding motion of the target in the image plane.

4.9 (b), and 4.10 (b). To balance convergence rate with target movement, we chose a gain of $k_1 = 3.0$ for flight tests.

The results of a stationary target in a constant wind of $V_w = 5$ m/s to the North are shown in Figure 4.11 (a). Notice that the flight path converges to an ellipse as was predicted in [17], without explicit path planning. The corresponding motion of the target in the image plane is shown in Figure 4.11 (b).

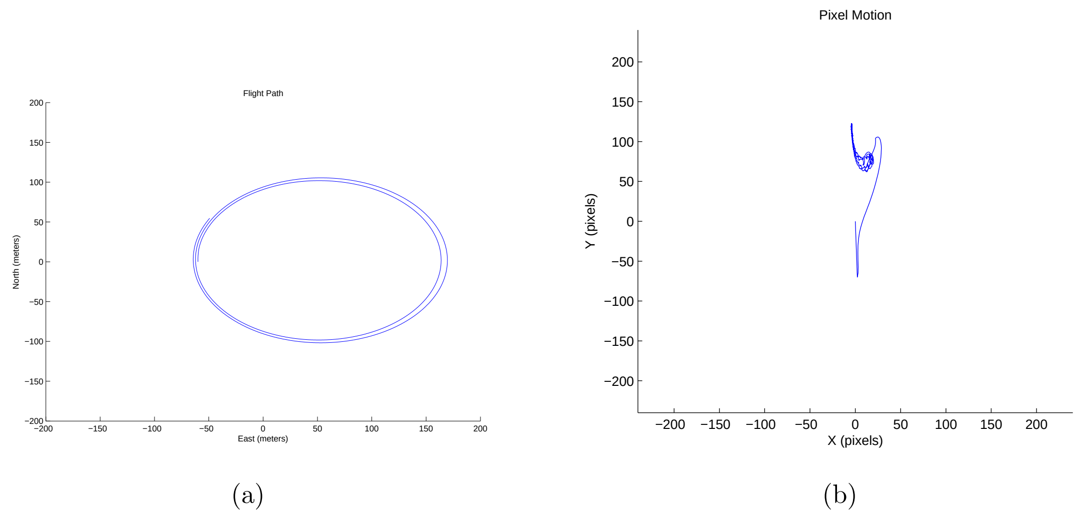


Figure 4.11: (a) Simulated flight path result with wind of 5m/s from the North, and parameters set to $k_1 = 5.0$, $\bar{\eta} = 0$, $\bar{\phi} = 20°$. (b) The corresponding motion of the target in the image plane.

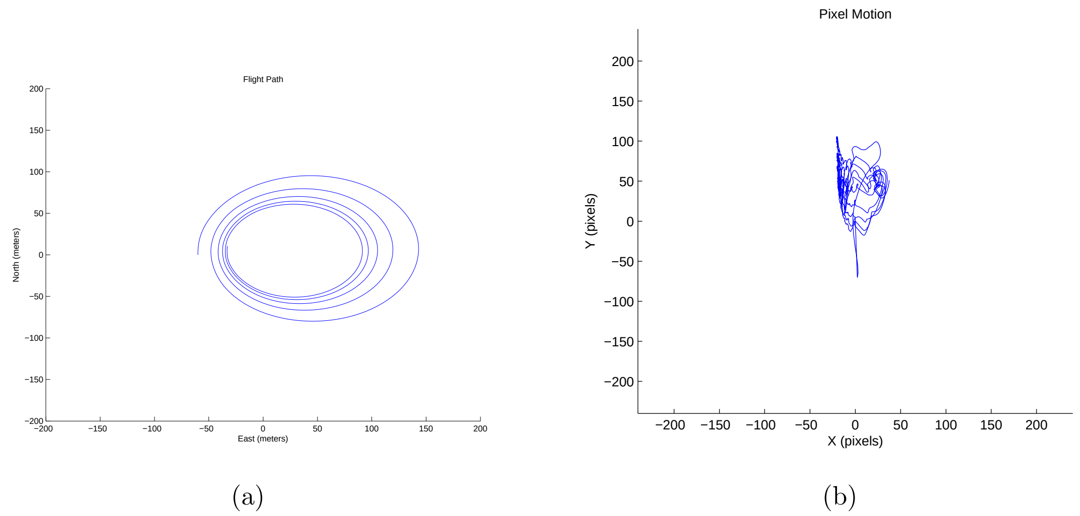


Figure 4.12: (a) Simulated flight path result with wind of 5m/s from the North, and parameters set to $k_1 = 5.0$, $\bar{\eta} = 30$, $\bar{\phi} = 20°$. (b) The corresponding motion of the target in the image plane.

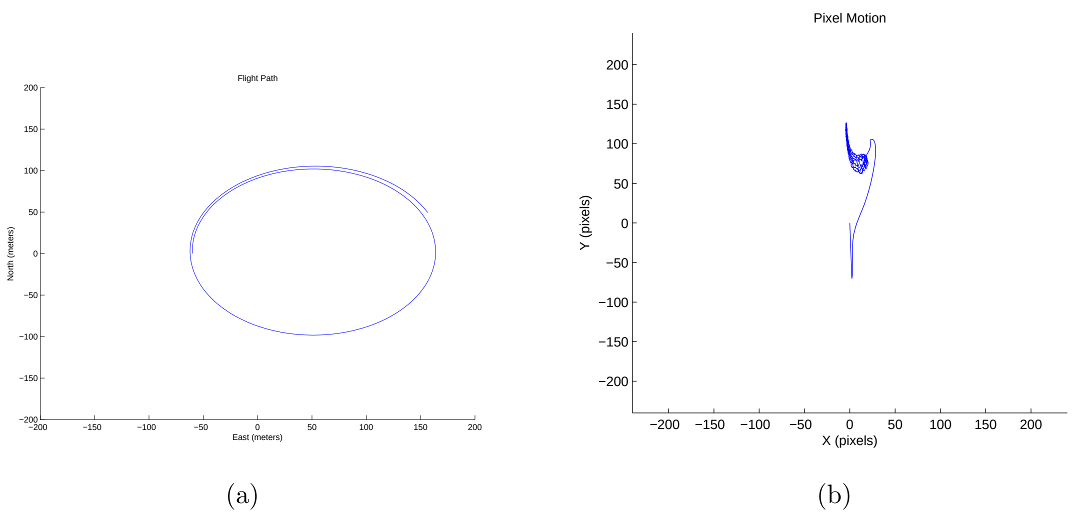
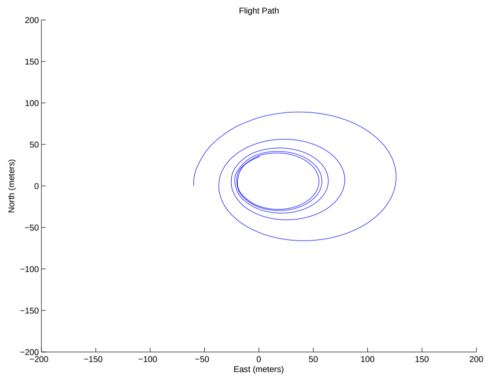


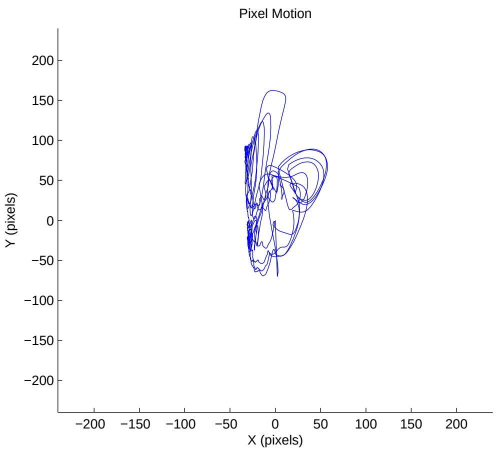
Figure 4.13: (a) Simulated flight path with wind of 5m/s from the North, and parameters set to $k_1 = 5.0$, $\bar{\eta} = 0$, $\bar{\phi} = 30°$. (b) The corresponding motion of the target in the image plane.

The constraint $\bar{\eta}$ bounds the movement of the target in the camera field-of-view. If the bound is relaxed, the target can move more in the camera field-of-view. The result is a more circular orbit, or an ellipse with a lower eccentricity. Figures 4.11 (a) and 4.12 (a) show the flight path for $\bar{\eta} = 0°$ and $\bar{\eta} = 30°$, respectively. The eccentricity of the ellipse decreases from 0.42 for $\bar{\eta} = 0°$ to 0.33 for $\bar{\eta} = 30°$. In addition, the target motion in the camera field-of-view increases as shown in Figures 4.11 (b) and 4.12 (b).

The constraint $\bar{\phi}$ bounds the roll angle of the MAV. As $\bar{\phi}$ increases, the MAV can fly a smaller orbit and converge more quickly to orbits because of the increased turning rate limit. The effect of $\bar{\phi}$ is shown in Figure 4.13 (a) with $\bar{\phi} = 30°$, $\bar{\eta} = 0°$ and 4.14 (a) with $\bar{\phi} = 30°$, $\bar{\eta} = 30°$. In Figure 4.13 (a), the constraint $\bar{\eta} = 0°$ prevents the MAV from increasing the roll angle to $\bar{\phi}$. An increase in the roll angle would also affect η, therefore $\bar{\eta}$ is the active constraint. However, when $\bar{\eta} = 30°$, the roll angle becomes the active constraint. The orbit, shown in Figure 4.14 (a), becomes smaller as expected. In addition, the target motion in the camera field-of-view increases as shown in Figure 4.14 (b).



(a)



(b)

Figure 4.14: (a) Simulated flight path with wind of 5m/s from the North, and parameters set to $k_1 = 5.0$, $\bar{\eta} = 30$, $\bar{\phi} = 30°$. (b) The corresponding motion of the target in the image plane.

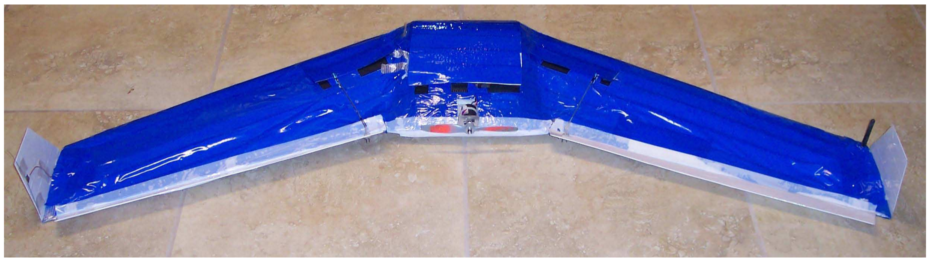


Figure 4.15: The MAV has a wing span of approximately 48 inches and uses the Kestrel autopilot from Procerus Technologies.

4.6 Flight Results

Flight tests were conducted using a MAV with a 48 inch wing span with two elevon control surfaces as shown in Figure 4.15, and the Kestrel autopilot from Procerus Technologies [39]. A camera was mounted on the MAV at an azimuth angle of 90° (out the right wing) and an elevation angle of 30°, as shown in Figure 4.16. The wind speed during the flight tests was approximately 2 meters per second. The parameters were $k_1 = 3$, $\bar{\eta} = 10°$, and $\bar{\phi} = 15°$.

The video was transmitted to the ground station via a 2.4 GHz analog transmitter, where a color segmentation algorithm was used to locate the target in the

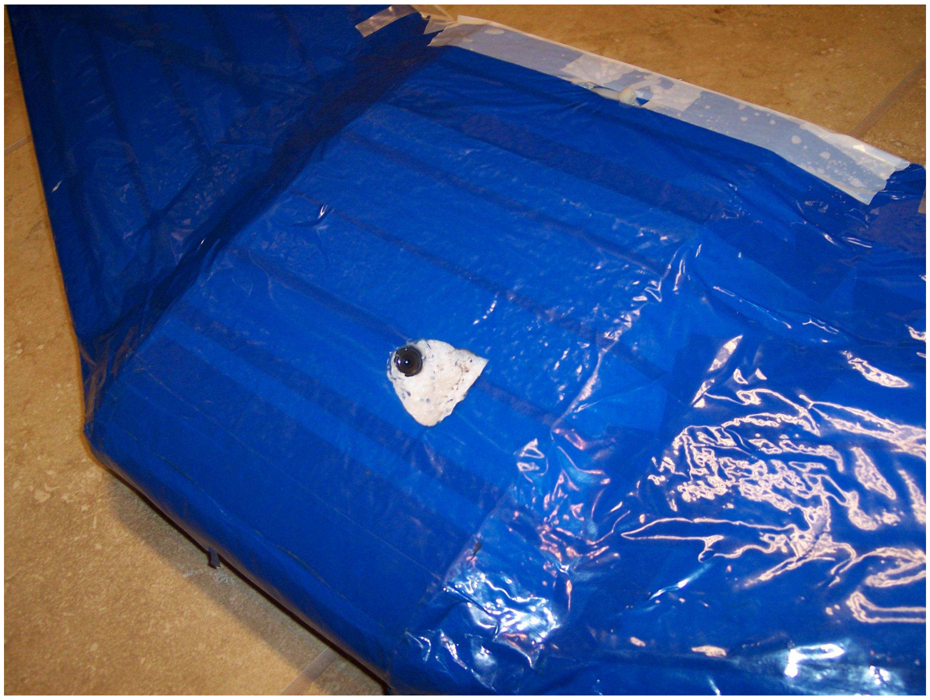


Figure 4.16: The camera is mounted under the MAV at an elevation angle of 30°.

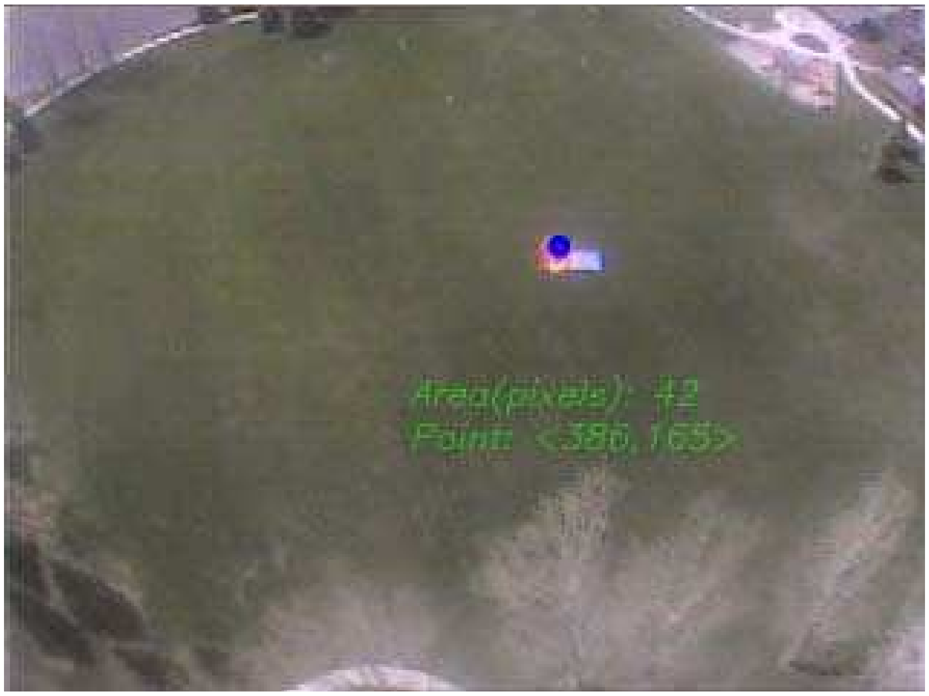


Figure 4.17: The ground station processes video by color segmentation to find the location of the red cloth.

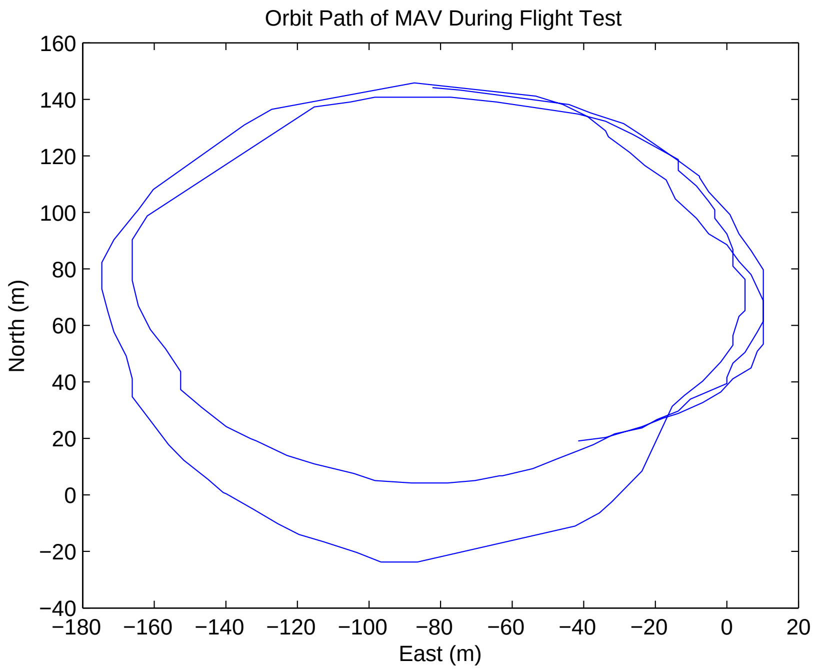


Figure 4.18: Flight results using the algorithm described in Section 4.3 in constant ambient wind of approximately $V_w = 2m/s$.

image frame. The target was a red cloth on the ground. The color segmentation algorithm used thresholds to remove all colors other than red, and returned the location of the largest patch of red. A representative image from the video sequence is shown in Figure 4.17.

The resulting flight path of the MAV is shown in Figure 4.18 and the target motion in the camera field-of-view is shown in Figure 4.19. The ambient wind resulted in an elliptical orbit with eccentricity $\epsilon = 0.17$. The theoretical prediction for a wind of $V_w = 2$ meters per second is $\epsilon = \frac{V_w}{V_a} = 0.15$. The guidance algorithm presented in this chapter successfully maintained the target in the field-of-view of the camera throughout the flight test.

4.7 Conclusion

In this chapter we have derived equations of motion for a target in a MAV body frame of reference and the image plane. In addition we have shown a feasible non-linear guidance law to track the target using image plane target movement using

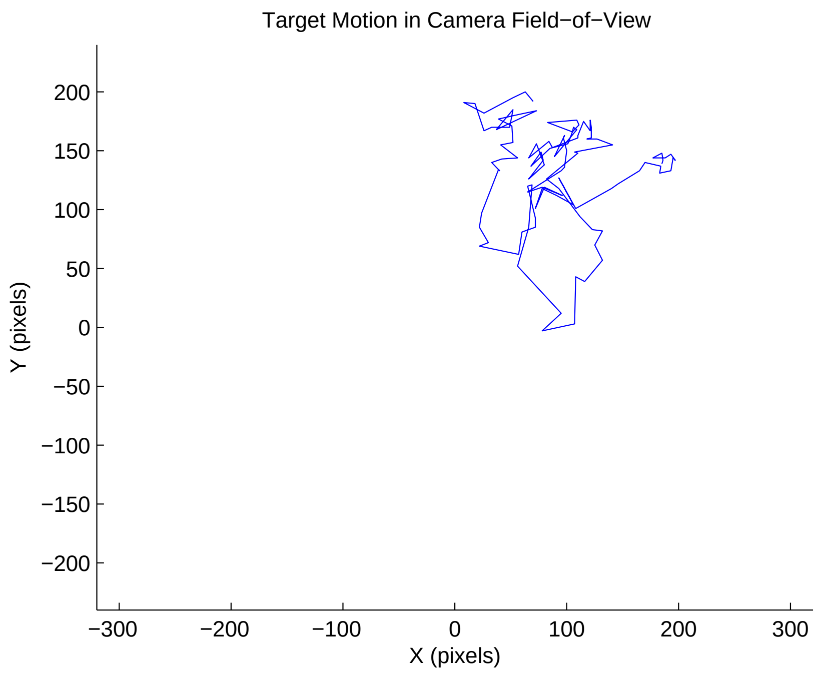


Figure 4.19: The motion of the target in the image during a flight test is shown.

a side-mounted camera. Movement in the image is used to "push" the target in the image plane to maintain it in the camera field-of-view. Simulation and flight results verify the effectiveness of the approach.

Chapter 5

A Priori Collision Avoidance Path Planning Using Branch and Bound Tree Search

5.1 Introduction

Micro air vehicles (MAVs) have received increasing amounts of attention in the last few years. Low cost and ease of use give them application for military and civilian purposes alike. Small onboard autopilots allow for accurate autonomous navigation removing the need to place pilots in danger. The next logical step is deployment of teams of MAVs to cooperatively and synergistically accomplish a series of tasks.

A cooperative team of MAVs has the potential to produce greater and more efficient results than a team of independent MAVs. They can adapt to changing situations, replan task assignments when a MAV is lost, communicate over large areas, and quickly complete scenarios. We would like to use as much of that potential as possible. The goal of this paper is to produce a method of task assignment such that we can produce a flight plan a priori and have the ability to replan mid-flight to adapt to changing situations.

Many military intelligence and reconnaissance mission scenarios involving teams of MAVs have been introduced in the past few years. Each scenario involves varying degrees of coupling between the goals of team coordination, task priority, and planning feasible paths to accomplish tasks. Finding a general optimal solution is difficult. Many methods result in combinatorial problems that are difficult to search. Algorithms in [54] and [72] use mixed integer linear programming to solve the combinatorial problem. Both methods involve a high amount of computation and cannot integrate collision avoidance and other coupled factors without adding to computation time. The capacitated transhipment network solver and iterated capacitated transhipment network solver in [73] and [74] respectively decouple the problem to produce suboptimal solutions.

The desired solution must have the ability to couple all constraining aspects of the problem, including path planning and collision avoidance, and be expandable to include other desired constraints. The branch and bound tree search described in [75] is expandable to include additional constraints. We propose to improve the algorithm to include additional costs to complement the distance cost already introduced in [75]. We will show that the combined additional costs produce near optimal results while successfully adding needed constraints.

5.2 Problem Description

Consider the proposed COUNTER solution for intelligence and reconnaissance missions. The COUNTER research program proposes to have one operator control one SAV (small air vehicle) and four MAVs as they perform the mission. The mission is to find and investigate objects in the city in order to locate targets. The task for the SAV is to orbit above the city providing video to the operator and acting as a relay for the MAVs. The MAVs are tasked to visit selected objects in the city in order to convict or acquit those objects as being targets. Since there are many UAVs involved and only one operator, the algorithms developed for COUNTER were designed to utilize the MAVs to maximize the number of objects acquitted or convicted.

COUNTER is a cooperative control environment in which MAVs must cooperate together to quickly observe objects. The cooperative environment involves many coupled problems that may not be so obvious at first glance including minimizing distance travelled, collision avoidance, staggered target arrival (to optimize operator efficiency), and revisiting poorly observed targets. This paper attempts to solve the coupling of minimizing distance travelled, collision avoidance, and staggered target arrival. The other coupled parts of the problem may also be solvable by the same methods, but are not explored here. The results of this chapter have been published in [76].

5.3 Branch and Bound Algorithm

The Branch and Bound tree search is described fully in [75] and is presented here for completeness.

Combinatorial problems can be completely represented by a decision tree. A decision tree is defined by a set of nodes $T = (N, E)$ where N is the set of all nodes and E is the set of all edges. Each edge is directed, meaning it can only be traversed in one direction. Every node can have parent nodes from which an edge is traversed, or children node to which an edge is traversed. The root node is the single node without a parent node from which all nodes derive. Leaf nodes are nodes without children nodes. The root node begins the tree and the leaf nodes terminate the tree. The leaf nodes represent possible solutions.

Each node in the tree is given a cost based on the scenario it represents. The cost function determines which solutions are optimal. The leaf node with the lowest cost is considered the optimal solution.

Branch and bound assumes the cost of nodes monotonically increases, i.e. all children of a node have a higher cost than the node itself. Using a best first search, a feasible solution can quickly be found. Any nodes with a larger cost than the initial solution can be pruned from the tree significantly reducing the number of nodes to expand. Large parts of the tree can be pruned. If a new leaf node with a lower cost is found, it becomes the current optimal solution with which to compare other nodes. The search can continue till either the tree is completely searched and the optimal solution is found, or some time limit is reached at which the current best solution is used.

In the COUNTER scenario, the root node represents all the vehicles and targets without any assignments. Each edge traversal adds one target assignment to a vehicle. The leaf nodes represent complete sets of target assignments to vehicles. In [75], the cost of a node is the maximum distance flown by any one MAV. The optimal solution minimizes the maximum distance flown by any one MAV.

5.4 Cost Function Algorithm

Our goal is to find a set of target assignments that minimize the distances traveled by any of the vehicles while maintaining a large distance between MAVs and staggering the MAVs arrival at objects. The J_2 (min-max) cost function of [75]

$$J_2 = \underset{i \in U}{max} R_i > 0, \tag{5.1}$$

where R_i is the distance traveled by MAV i, is used to minimize the distance traveled by any of the MAVs. In order to meet the other objectives of the goal, we will add one cost representing the proximity distance to other MAVs (collision cost) and another representing the minimum time between target arrivals (arrival time cost).

5.4.1 Collision Cost

We assume all paths flown by MAVs between targets are Dubbin's paths [77]. Each path consists of turns of radius R and straight lines. Lines and turns will be considered different waypoints to simplify calculations even though they represent a single Dubbin's path. The following descriptions estimate the minimum distance between MAVs flying combinations of straight line paths and turning paths.

Two Straight Line Paths

Straight line paths can be described perimetrically by:

$$\bar{\mathbf{m}}_i = \mathbf{w}_{i,j} - \mathbf{w}_{i,j-1},$$

$$\mathbf{m}_i = \frac{\bar{\mathbf{m}}_i}{\parallel \bar{\mathbf{m}}_i \parallel} = \frac{\mathbf{w}_{i,j} - \mathbf{w}_{i,j-1}}{\parallel \mathbf{w}_{i,j} - \mathbf{w}_{i,j-1} \parallel},$$

and

$$\mathbf{r}_i = V_i t \mathbf{m}_i + \mathbf{w}_{i,j-1}, \tag{5.2}$$

where j is the current waypoint and $t = 0$ when $r_i = w_{i,j-1}$. Equation 5.2 represents a MAV perfectly tracking a waypoint path starting at $\mathbf{w}_{i,j-1}$ when $t = 0$ and finishing at $\mathbf{w}_{i,j}$ when $t = \frac{\|\mathbf{w}_{i,j} - \mathbf{w}_{i,j-1}\|}{V_1}$.

We want to know the distance between two MAVs at any point along the path,

$$\| \mathbf{r}_1 - \mathbf{r}_2 \|^2 = d^2,$$

$$\| (V_1 t\mathbf{m}_1 + \mathbf{w}_{1,j-1} - V_2 t\mathbf{m}_2 - \mathbf{w}_{2,j-1}) \|^2 = d^2. \tag{5.3}$$

and take the minimum. The minimum can be found by taking the derivative and setting equal to zero. The result is:

$$t = \frac{V_1 \mathbf{m}_1^T (\mathbf{w}_{2,j-1} - \mathbf{w}_{1,j-1}) + V_2 \mathbf{m}_2^T (\mathbf{w}_{1,j-1} - \mathbf{w}_{2,j-1})}{V_1^2 \mathbf{m}_1^T \mathbf{m}_1 + V_2^2 \mathbf{m}_2^T \mathbf{m}_2 - 2V_1 V_2 \mathbf{m}_1^T \mathbf{m}_2}. \tag{5.4}$$

The minimum distance can be found by substituting t into

$$d = \| \mathbf{r}_1 - \mathbf{r}_2 \| . \tag{5.5}$$

Turn and Straight Line Path

The minimum distance between a turn and a straight path is somewhat more difficult algebraically. A gradient descent could be useful for accuracy, but introduces a large amount of computation. We can estimate the proximity of a turn path and a straight path to within an error of $2R$ by finding the geometric intersections of the paths and checking the times of those intersections.

Consider the equations of a circle and line:

$$(r_e - w_{1,j-1,e})^2 + (r_n - w_{1,j-1,n})^2 = (R + r_d)^2, \tag{5.6}$$

and

$$r_n = m(r_e - w_{2,j-1,e}) + w_{2,j-1,n}. \tag{5.7}$$

These can be simultaneously solved and the minimum can be found using the quadratic equation. Substitute

$$a = 1 + m^2, \tag{5.8}$$

$$b = 2(w_{2,j-1,n} - w_{1,j-1,n} - m w_{2,j-1,e}) - 2w_{1,j-1,e}, \tag{5.9}$$

and

$$c = (w_{2,j-1,n} - w_{1,j-1,n} - m w_{2,j-1,e})^2 + w_{1,j-1,e}^2 - (R + r_d)^2, \tag{5.10}$$

into

$$r_e = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{5.11}$$

r_n can be found by substituting r_e back into equation 5.7. If the result is imaginary, then the circle and line don't intersect. If the intersection isn't in the time frame of the waypoints, then they don't intersect. In these cases, the distance from the center of the circle to the MAV on the waypoint path in the correct time frame can estimate the distance.

Two Turns

For the distance between two turns, the distance between the circles is sufficient if they are in the same time frame. This introduces a maximum error of $4R$.

Cost Function

The cost function needs an adjustable parameter such that a desired minimum distance is imposed on the MAVs. An inverse square distance relationship allows the cost to greatly increase as distance decreases. Multiplication by β adds a tunable parameter to influence the distance at which the cost becomes significant. The collision cost function is,

$$J_{CC} = \frac{\beta}{\min_{i,j}\left(d_{i,j}\right)^2} > 0, \{i,j\} \in U, i \neq j. \tag{5.12}$$

5.4.2 Arrival Time Cost

The arrival time at each target is calculated using the length of the path travelled to the target and a constant velocity assumption. We find the minimum difference between any two arrival times and substitute that into an inverse square formula for the cost. Once again we multiply the result by a constant γ to tune the cost. The arrival time cost is,

$$J_{AT} = \frac{\gamma}{\min_{i,j}\left(t_{i,j}\right)^2}, \{i,j\} \in U, i \neq j, \tag{5.13}$$

where $t_{i,j}$ is the difference in arrival time between targets i and j.

5.4.3 Combined Arrival Time and Collision Cost

The combined cost function combines the min-max cost with the two new costs into one function. The constants β and γ in Equation 5.12 and Equation 5.13 allow for tuning. The function is,

$$J = J_2 + J_{CC} + J_{AT}. \tag{5.14}$$

5.5 Simulation

We ran 100 Monte Carlo simulations to demonstrate the effectiveness of the collision cost, arrival time cost, and combined cost. Each run consists of three vehicles and seven targets. The number of targets and vehicles is small to allow time to search the tree for the optimal solution in each run. The optimal solution could then be compared with the time limited branch and bound solution.

Each run used a uniform distribution to place targets and vehicle starting positions on square map. The map size is 3,000 feet by 3,000 feet for all calculations. However, simulations were also run on maps of up to 7,000 feet by 7,000 feet to show the path planner effectiveness at larger map sizes. Each MAV is assumed to have the same constant velocity.

The path planner was executed to the optimal solution four times for each run, once for min-max cost, collision cost, arrival time cost, and combined cost. With each cost function computed on the same data, they could be compared with more accuracy. We chose for parameters $\beta = 1000000$ and $\gamma = 15000$ to balance the costs based on simulation performance.

5.6 Results

5.6.1 Collision Cost

Our goal with the collision cost was to increase the proximity distance of the MAVs with minimal change in the min-max cost. The distance between MAVs was increased with an average of 90% compared to the min-max cost runs. The average increase in min-max cost was 0.8%.

Figure 5.1 shows the distance increase for each of the 100 runs with the percentage increase in figure 5.2. The runs are ordered with increasing distance. Most of the runs result in a significant increase in distance. The runs without change in distance already have a large distance between MAVs, with the smallest at 300 feet.

The difference in min-max cost between the two runs is displayed in figure 5.3. Little difference is made in min-max cost between cost functions. Notice the two

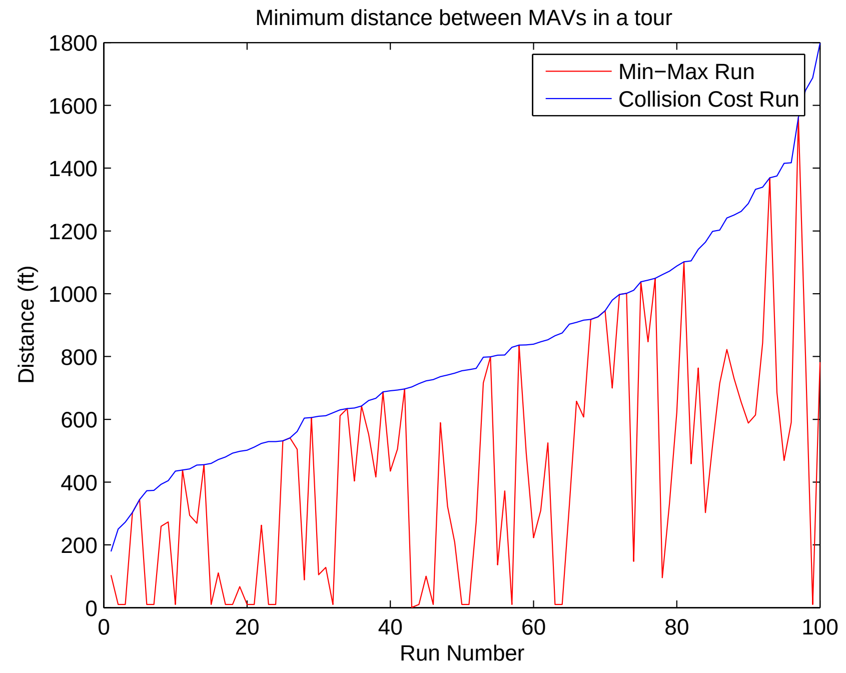


Figure 5.1: The minimum distance between MAVs over a 100 tours is shown.

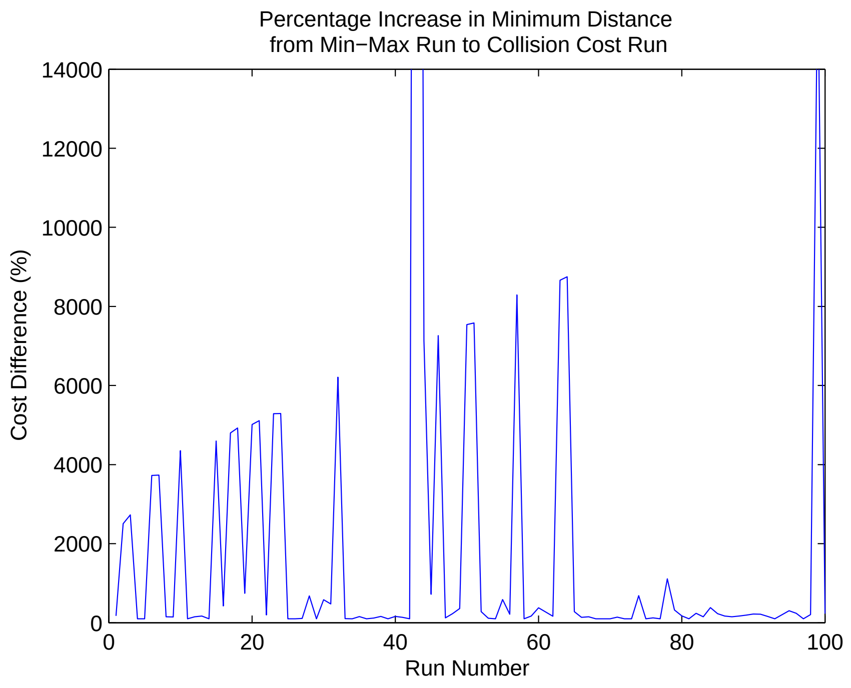


Figure 5.2: The percentage increase in the minimum distance from the min-max cost run to the collision cost run.

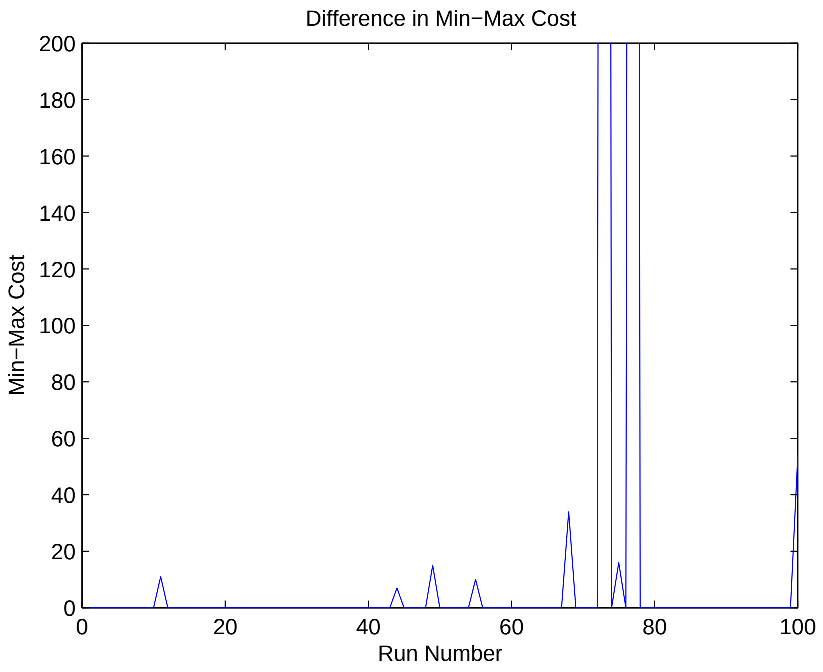


Figure 5.3: The difference in the min-max costs between the min-max run and the collision cost run.

spikes in the graph. This is the result of two MAVs with starting positions in close proximity. If the MAVs have similar starting positions, the planner will remove one of them from flight to prevent collision. This is expected, however undesirable.

All of the above calculations result from tree searches to find the optimal solution. The optimal solution can take large periods of time to find due to the combinatorial nature of the problem. However, the branch and bound search can find a near optimal solution in tenths of a second. Figure 5.4 shows the increase in average minimum distance across the runs as nodes are processed. A larger amount of nodes implies a larger amount of time to process. The minimum distance from the collision cost runs tends to increase as more nodes are processed. Notice the minimum distance in the min-max costs runs show little change over number of processed nodes.

5.6.2 Arrival Time Cost

The arrival time cost has similar results to the collision cost. Figure 5.5 shows the arrival time difference between the arrival time cost function and the min-max

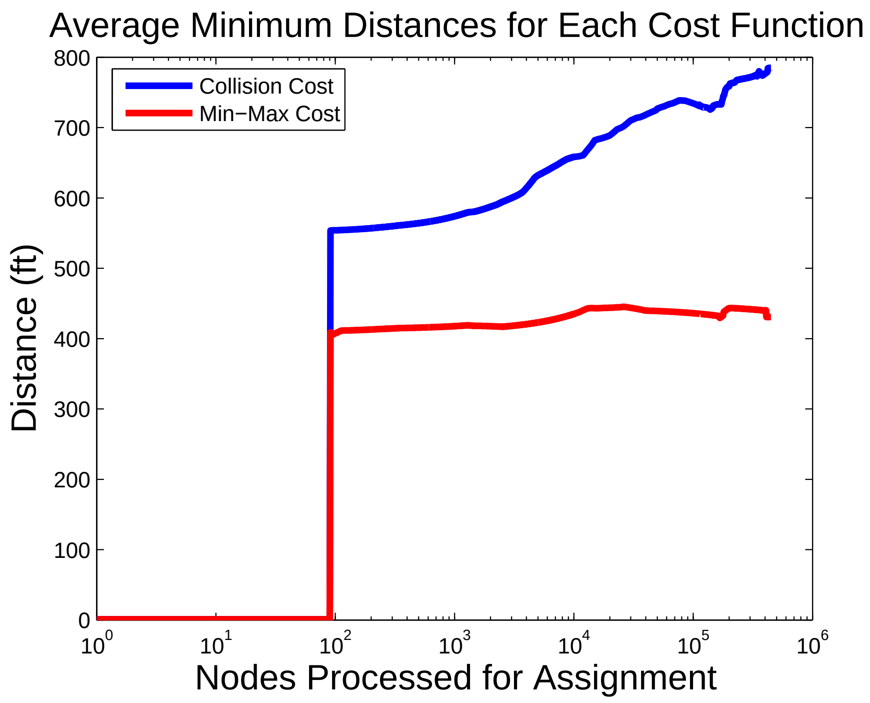


Figure 5.4: The average minimum distance between MAVs as nodes are processed is shown. The collision cost results in increased distance between MAVs.

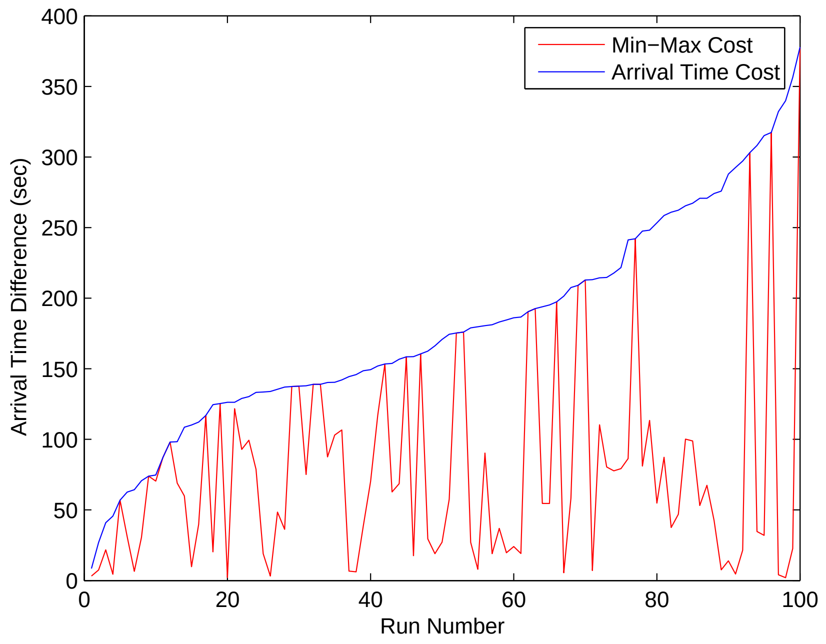


Figure 5.5: The minimum difference in arrival times between targets is plotted for each run. The arrival time cost function produces arrival time differences better than or equal to the min-max cost function.

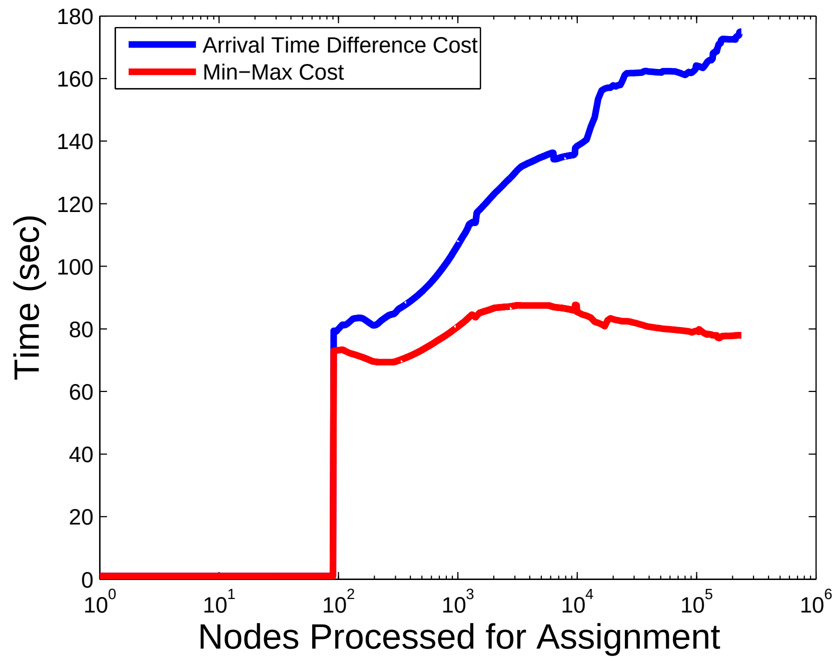


Figure 5.6: The minimum difference in arrival times between targets is plotted against number of nodes processed.

cost function. The arrival time cost function results in arrival time differences greater than or equal to that of the mix-max cost function. Arrival time differences are increased by 133%. The min-max cost is increased by 0.06%.

The arrival time difference is plotted with the nodes processed in figure 5.6. Once again, they are similar to the results of the collision cost. The initial solution has an initial increase in arrival time difference and the solution improves over time.

5.6.3 Combined Cost

The combined cost has the potential to improve proximity of the aircraft and stagger target arrival times while keeping the distance flown by the MAVs close to optimal. The performance of these goals can be measured by comparing the results of the combined cost function to that of the other cost functions.

Figure 5.7 compares the expected proximity distance of the MAVs as nodes are processed. The combined cost function results are similar to the results of the collision cost function. Adding another cost had very little effect on proximity results.

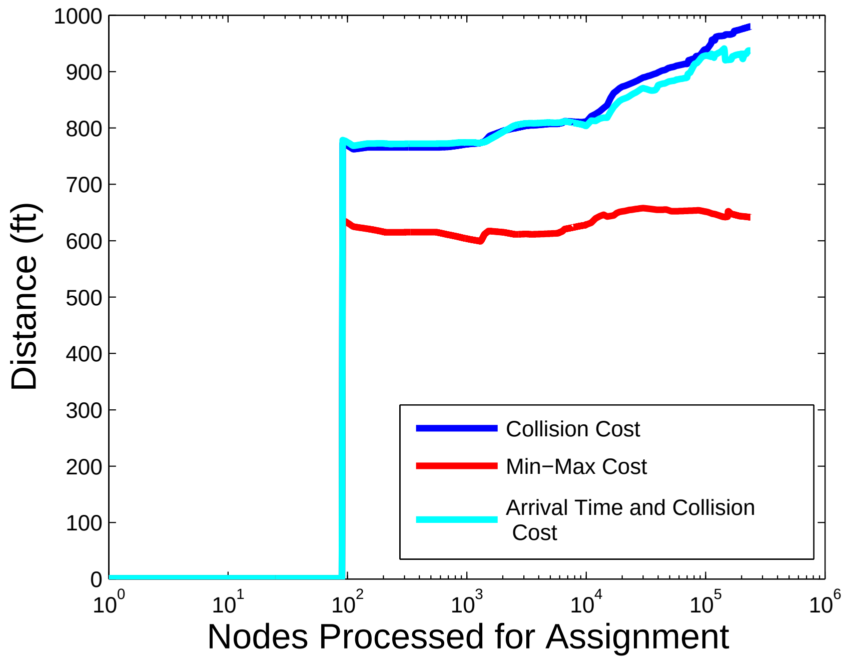


Figure 5.7: The minimum expected distance between MAVs is plotted against number of nodes processed. The results for the combined cost function are very similar to the results of the collision cost function.

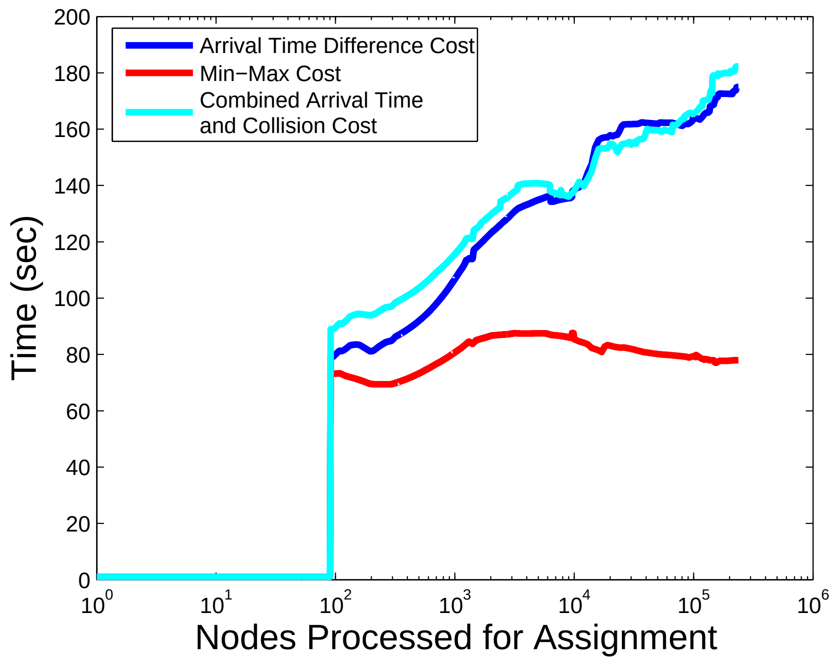


Figure 5.8: The expected minimum difference in arrival times is plotted against number of nodes processed. The arrival time and combined cost functions have similar results.

The differences in target arrival times are shown in figure 5.8. The combined cost function results are similar to the arrival time cost function. The combined cost has done little to negatively affect the target arrival time difference.

The min-max cost increased by 0.64% from the min-max cost run to the combined cost run. This is a small difference and worth the large increase in proximity distance and staggered target arrival.

5.7 Conclusion

In this paper we have demonstrated a method of assigning targets to multiple vehicles while balancing objectives of distance traveled, collision avoidance, and staggered target arrival. The results show significantly improved expected distance between MAVs and a large increase in staggered arrival time. Both of these features may be added simultaneously with little difference in results. Suboptimal solutions may be found quickly using a time limited branch and bound search.

The success of the additional costs suggests additional research into other desired constraints that may vary between different scenarios and objectives. The combinatorial nature of the problem allows for a large number of near optimal solutions that may satisfy a variety of constraints.

Chapter 6

Conclusion and Future Work

Obstacle avoidance with pop-up obstacles is a difficult problem. The constraints of fixed-wing air vehicles add complications that current avoidance strategies are unable to overcome. The minimum velocity of fixed-wing aircraft create time constraints on obstacle avoidance strategies. Finding optimal paths around obstacles is infeasible under time constraints because they require extensive computation using current methods. In this work, we contribute to current literature by presenting four strategies for obstacle avoidance that attempt to overcome these problems.

The first strategy, discussed in Chapter 2, uses laser rangers. The guidance generates sets of waypoints around obstacles. The waypoints maneuver the MAV around the obstacle, after which the MAV continues on the waypoint path. There is not always a guarantee of obstacle avoidance, but computation is minimized.

The second strategy uses a laser ranger to scan the area in the reachable region of the MAV by oscillating along a waypoint path. Obstacles in the path of the MAV are guaranteed to be detected and avoided. However, the oscillation is an undesirable feature.

For single obstacles, we developed a guidance strategy in Chapter 3 based on movement of the obstacle in the field-of-view of a camera. The guidance strategy estimates the range-to-obstacle and bearing-to-obstacle from obstacle movement in the camera field-of-view and the guidance strategy pushes the obstacles to the edge of the camera field-of-view. After maneuvering around the obstacle, the MAV continues along the waypoint path.

For multiple obstacles, we developed a guidance strategy to maneuver the MAV between obstacles based on motion in the camera field-of-view. A strategy is developed to determine between which obstacles to maneuver. This and the single obstacle strategy perform well in simulation and flight tests. However, they are intended for the zero to three second time frame while high level path planners generate paths around obstacles.

Chapter 4 addressed vision-based target tracking using a strap-down camera. Similar to the vision-based obstacle avoidance strategies, our method uses movement of the target in the camera field-of-view to guide the MAV and maintain the target in the field-of-view. The resulting paths are elliptical as predicted in literature.

In Chapter 5 we discussed a branch and bound algorithm to assign tasks to teams of UAVs. The problem is posed as a combinatorial optimization problem. A feasible solution is possible in less than a second, and better solutions are possible with increased computation time.

The research contributions of this work are as follows.

- developed a 2D obstacle avoidance strategy to generate waypoint paths around obstacles in real time,
- developed a 2D obstacle detection and avoidance strategy using a laser ranger and circular paths that guarantees obstacle detection and avoidance,
- developed a 2D visual obstacle detection and avoidance strategy for single obstacles,
- developed a 2D and 3D visual obstacle detection and avoidance strategy for multiple obstacles,
- derived a visual target tracking strategy for strap-down cameras,
- formulated a task allocation algorithm for teams of MAVs incorporating distance traveled, collision, and staggered arrival to task, and
- described a simulation environment designed for fast research and prototyping in an educational environment.

6.1 Future Work

There are number of prospective research topics that may extend from this research. Laser rangers are light-weight and potentially require little space. Multiple laser rangers on a single MAV may give enough information to guarantee obstacle detection without a need for obstacle avoidance. If the obstacles are detected outside the turning radius of the MAV, then a guidance strategy might be able to guarantee obstacle avoidance. This method would not require vision processing and could run on a low power system onboard the MAV.

Vision-based reactive obstacle avoidance has many areas for improvement. Satisficing control similar to [78] may be possible to guide the MAV around obstacles detected in the image plane as opposed to the deterministic method in this work. The guidance law would require little processing and possibly perform better. Additionally, probabilistic methods such as risk functions to assign a risk to each pixel in the image plane need to be explored. A guidance law based on risk functions would maneuver the MAV in the direction of the pixel with the lowest risk.

This work did not explore vision processing extensively. However, the state estimation methods developed in Chapter 3 performed well enough for flight tests with large red obstacles. Adding additional information from object looming, feature tracking, and other vision processing techniques may improve range and bearing estimation.

In Chapter 4, the guidance strategy minimized the distance to the target. However, this may not always be the goal of the operator. Similar goals may be to maintain a constant target size in the image, stay outside a minimum distance, maximize altitude, or minimize target movement in the image plane. Each of these could be based on a guidance law similar to the one explored in this dissertation.

Chapter 5 calculated collision using deterministic methods. A probabilistic technique may be more accurate and more favorable to differing vehicle models.

Appendix A

Matlab Simulation Environment

A.1 Introduction

Unmanned Air Vehicles (UAVs) take on tasks to replace piloted aircraft in situations where human intervention is dangerous, too expensive, or infeasible. In military tasks, they remove danger to human pilots by flying surveillance, reconnaissance, and attack missions without human involvement. Domestic missions include forest fire monitoring, search and rescue, and surveillance. Very often, small, relatively inexpensive UAVs accomplish these tasks. UAVs not only remove human risk, but lower the overall costs involved in mission tasks.

The high number of potential applications and the low cost of small UAVs have kindled research and development in small autopilots. The decrease in size of the required sensors and integrated circuits have allowed the development of light-weight autopilots capable of navigating airframes with wingspans as small as 6 inches [40,79]. The sensors available for large aircraft are not feasible on small UAVs because of payload limitations. They require small sensors that are typically less accurate. Small UAVs typically have fast dynamics, making accurate sensors even more important. While the technology of sensors improve, the theory for inner loop autopilot control continues to develop using alternative sensors such as computer vision [80]. The goal of this appendix is to create a simulation platform to aid in these new control methods.

The purpose of our research is to develop a simulation that expedites prototyping of research and development by providing mechanisms to test new control algorithms and sensors on various types of UAVs, including vision-based control. The base simulation is simple, but offers enough features to aid in development of complex control.

The University of Florida developed a simulation testbed for rapid vision processing [81]. Their goal was to accelerate the process of development of vision-based control inside simulation and smoothly transition the control to flight tests. One of our goals in this simulation is similar to theirs. We want to create a simulation where vision-based control can be rapidly developed in a simulation environment, tested with hardware in the loop, and transitioned smoothly to flight tests. We briefly discuss an application in which this was accomplished and the flight tests performed with that vision based control [58].

We have five goals for this simulator,

1. modularity to support different airframes and research,
2. fast research and development prototyping,
3. hardware-in-the-loop testing,
4. graphical interface with camera simulation for vision processing, and
5. research standardization to prevent code loss when employees leave.

A.2 Overview

We want the simulation to be modular. At Brigham Young University, we have a wide variety of airframes, including fixed wing aircraft, quadrotors, helicopters, and vertical take-off and land (VTOL) aircraft. We want to incorporate each of these different airframes into the simulation with minimal change when changing airframes, including physics and autopilot capabilities. To help with this, the simulation is encoded in Matlab's Simulink.

Simulink has many beneficial characteristics for a simulation using multiple airframes and autopilots that include,

1. a scripting interface for fast development,

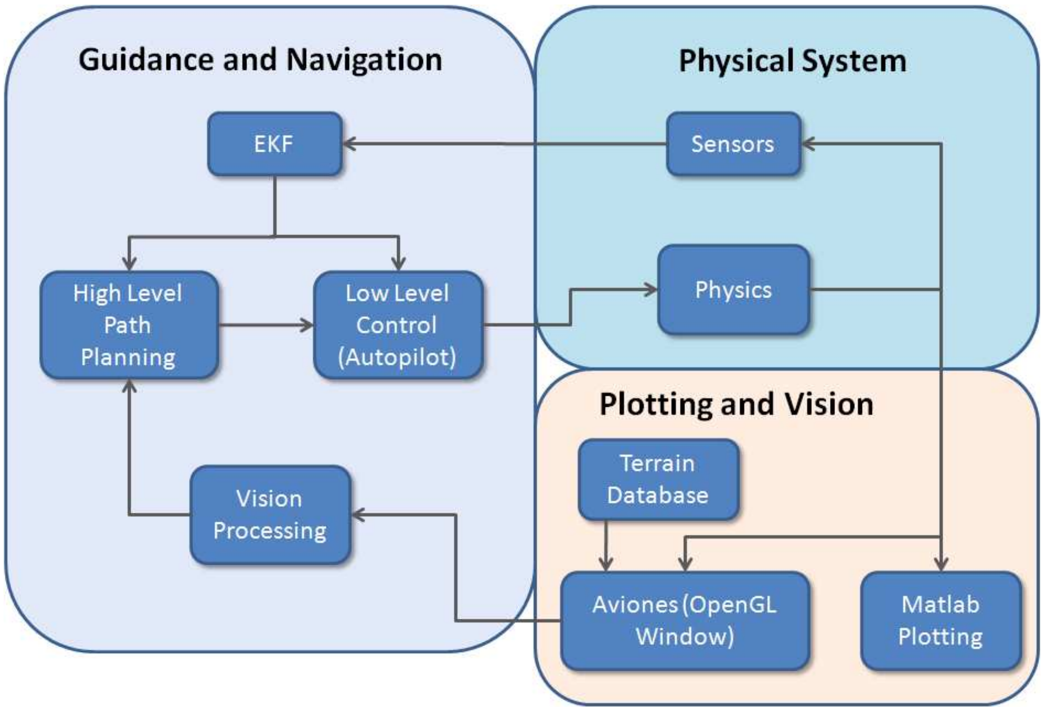


Figure A.1: The organization of the simulation is shown.

2. a large set of engineering script libraries including control and computer vision,
3. a customizable library for simulation blocks that can be used for different airframes,
4. simple data plotting functions, and
5. an interface for C++ code (mex functions).

The block diagram of the simulation structure is shown in Figure A.1. There are three subsystems. The first is the Guidance and Navigation System. This includes the Extended Kalman filter (EKF) for state estimation, the High Level Path Planning, the Autopilot, and any vision processing required. For most research development, this block is most important.

The second subsystem is the Physical System. This includes the sensors onboard the autopilot and the physics of the airframe. This system is important when changing airframes. A fixed-wing aircraft will have different flight characteristics than a quadrotor. To make this simulation universal, we store the physics engine of each airframe in the Simulink library. Each physics engine can be dragged from the Simulink library to the simulation.

The third subsystem is Plotting and Vision. There are two purposes of this section. The first is to display any graphs, plots, and data as required. The second is the graphical display called Aviones. Aviones displays the UAV in a terrain and simulates a camera for computer vision.

These three subsystems are simple, yet capable of fulfilling the goals of the simulation.

The end goal of research and development is not to produce a good implementation in simulation, but to produce flight results from the theory and code developed in the simulation. This simulation architecture provides one more step in that process to completion. Most embedded systems are coded using C and possibly C++. To implement an autopilot on a microprocessor, the Matlab script must be converted to C code. The conversion to C code also opens possibilities for bugs and problems that were not present in the script version. Fortunately, Matlab also has a C/C++ interface called mex functions. The script can be converted to C/C++ code and tested the simulation environment without changing any other details. This limits the programming bugs and errors present in the embedded system.

In an educational environment, the employment length of each student is relatively short before they graduate and move on. Students usually implement their research on the current autopilot platform, or in a specialized simulation, and leave. The code is rarely reusable and often poorly documented. This is another problem this simulation architecture will remedy. Each student provides a section of code with a simulation block that corresponds to their research, ideally using a script implementation and a mex function implementation. Some students work on low level design for new airframes, others high level control, and sometimes a new physics engine is required. Each of these are integrated into a Simulink block and added to a Simulink Library. After the student leaves, the research completed by the student may be required for another project. Rather than implement the research again, we can use the library block previously created. This standardized method of research for short term employees minimizes code loss.

The simulation has a standard method of transport that each researcher must adhere to. The system consists of 18 states. They are inertial position (p_n, p_n, h), body frame velocity (u, v, w), roll ϕ, pitch θ, yaw ψ, angular accerations (p, q, r), the azimuth α_a and elevation α_{el} of the camera, and the quaternion representation of the orientation (e_1, e_2, e_3, e_4) which is often required by VTOL aircraft. These states contain enough information for a physics engine of any airframe developed at BYU. By adhering to these standards, each block created will continue to be usable after its initial purpose has expired.

A.3 Physics

The physics of an airframe can be a difficult section to complete accurately. Matlab plays a key role in helping researchers write correct code for the physical system. In the case of a fixed-wing aircraft, there are typically 12 states which need differential equations. The force equations we use are

$$\begin{pmatrix} f_x \\ f_y \\ f_x \end{pmatrix} = mg \begin{pmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{pmatrix} + \frac{1}{2}\rho V_a^2 S \begin{pmatrix} C_X(\mathbf{x}, \delta) \\ C_Y(\mathbf{x}, \delta) \\ C_Z(\mathbf{x}, \delta) \end{pmatrix}, \tag{A.1}$$

$$\begin{pmatrix} l \\ m \\ n \end{pmatrix} = \frac{1}{2}\rho V_a^2 S \begin{pmatrix} \frac{b}{2}C_X(\mathbf{x}, \delta) \\ \bar{c}C_Y(\mathbf{x}, \delta) \\ \frac{b}{2}C_Z(\mathbf{x}, \delta) \end{pmatrix}, \tag{A.2}$$

where C_* are the coefficients of lift and drag, $\mathbf{x}$ is the states, and δ is the surface deflections. The differential equations for a fixed-wing aircraft using those forces are

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{pmatrix} = \begin{pmatrix} c\theta c\psi & s\phi s\theta s\psi - s\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\theta\psi & c\phi s\theta s\psi - s\phi c\psi \\ s\theta & -s\phi c\theta & -c\phi c\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \tag{A.3}$$

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \begin{pmatrix} -g\sin\theta \\ g\cos\theta\sin\phi \\ g\cos\theta\cos\phi \end{pmatrix} + \frac{\rho V_a^2 S}{2m} \begin{pmatrix} C_X(\mathbf{x},\delta) \\ C_Y(\mathbf{x},\delta) \\ C_Z(\mathbf{x},\delta) \end{pmatrix}, \tag{A.4}$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}, \tag{A.5}$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \Gamma_1 pq - \Gamma_2 2qr \\ \Gamma_5 pr - \Gamma_4(p^2 - r^2) \\ \Gamma_6 pq - \Gamma_1 qr \end{pmatrix} + \frac{\rho V_a^2 S}{2} \begin{pmatrix} \frac{b}{2}\left(\Gamma_3 C_l(\mathbf{x},\delta) + \Gamma_4 C_n(\mathbf{x},\delta)\right) \\ \frac{\bar{c}}{J_y} C_m(\mathbf{x},\delta) \\ \frac{b}{2}\left(\Gamma_4 C_l(\mathbf{x},\delta) + \Gamma_7 C_n(\mathbf{x},\delta)\right) \end{pmatrix}. \tag{A.6}$$

These sets of equations, and the equations for lift, drag, and force are complicated. Mistakes are easy to make and often occur. Matlab makes debugging easier with scripting, breakpoints, and plots. After a working copy of the script is ready, the script can be converted into a mex function because Matlab can sometimes consume a lot of processing time executing a script. We compiled the fixed-wing physics into a mex function to lower the processing time. This is more simple after the script implementation. Each of the physics Simulink blocks for the different airframes are integrated into the Simulink library so future researchers can drag and drop the desired airframe physics into the simulation. When a researcher needs to use an airframe in simulation, he or she only needs to drag and drop it from a library.

A.4 Guidance and Navigation

The guidance and navigation section of the simulation includes the low level control, high level control, vision processing, and sensor filters.

A.4.1 Autopilot

The autopilot is the low level inner loops to maintain altitude, heading, roll, pitch, heading rate, etc. Commercial autopilots commonly use a series of PID loops for low level control of an aircraft. For the fixed wing autopilot in this simulation, we use the PID loop architecture described in [40].

Simulink can implement an autopilot in script or in a C++ mex function. In this simulation, it is advantageous to use both methods. The scripting interface offers an excellent method of implementing and testing an autopilot system. Scripting requires no compilation, allows breakpoints, and provides complex mathematical operations with simple commands. After theoretical analysis is complete, the scripting interface provides a fast method to test the theory. After a script method demonstrates theoretical correctness, a C mex function can be written in simulation before porting it to an autopilot. The mex function provides a method of testing C code in simulation to minimize errors before moving it to an autopilot.

A.4.2 High Level Path Planning

The high level path planning can use any method required by the researcher. The most common method is manually created waypoint paths. The UAV flies a series of waypoints, and when the last waypoint is reached, the UAV returns to the first waypoint and starts over. The method we use of waypoint following is described in [38]. This is the default method of high level planning in this simulation. The high level planner commands a roll angle to the autopilot to stay on the waypoint path.

In the case automatic path planning is required, we often use rapidly-expanding random trees [26,36]. A series of random waypoints are generated and connected to a tree such that the kinematic constraints of the vehicle are not violated. This generates a waypoint path from a starting position to the goal.

Each researcher requires different methods of path planning. Each method is stored in a library Simulink block so code does not need to be rewritten when it is required in the future.

Figure A.2: The Procerus autopilot is used for hardware-in-the-loop simulations.

A.4.3 Vision Processing

Matlab has a wide variety of vision processing functions available through its scripting interface. The vision processing Simulink block can use the Matlab vision processing functions for fast development of required vision processing. The Aviones graphics block simulates a camera and sends the pixel data to the vision processing block. The pixel data from Aviones is treated as pixel data from an actual camera; even adding noise is possible. Vision processing on the data is different for each application, but the development process is simplified by script implementation. The Matlab functions offer a method to check theory validity without a lengthy program in C.

A vision processing implementation in C is usually required eventually, whether for the ground station or onboard the aircraft. To ease the process of converting from Matlab script, we use a mex function with OpenCV support. OpenCV is an open source vision processing library. OpenCV has many powerful vision functions. Using OpenCV in a mex function greatly reduces the time required in converting from script to C code. The mex function is used in the same way as a script file, allowing the C implementation to be tested in the simulation before using it in a separate program.

A.4.4 Hardware in the Loop

A common step in hardware development is hardware-in-the-loop testing. Before any flight, a control platform needs testing as close to actual flight as possible. Researchers often use the embedded autopilot with simulated physics to test the platform. In our case, the physics engine is located on the autopilot. An RF modem sends

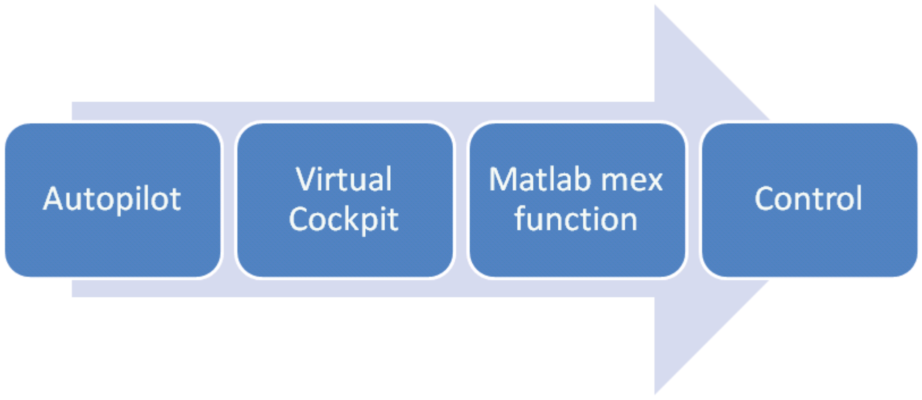


Figure A.3: Telemetry information flows from the autopilot to Virtual Cockpit via an RF modem, to Simuliank via a TCP/IP connection.

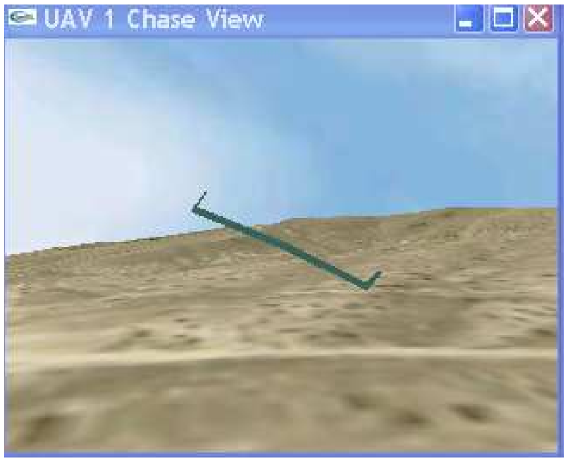
the telemetry data from the autopilot to the ground station. A computer program called Virtual Cockpit collects the data. A TCP/IP connection sends the telemetry data to other programs as required, including Matlab through a mex function. The simulation can use the telemetry information for any hardware-in-the-loop testing required, such as vision simulation with Aviones and vision processing with the script interface or OpenCV. Running hardware-in-the-loop using Matlab creates a vision simulation that normally isn't available for hardware-in-the-loop. This option speeds up the development cycle.

The hardware-in-the-loop mechanism is also available for flight tests. While another program can be made for monitoring and vision processing to separate simulation from hardware-in-the-loop and flight tests, the option of leaving control in Matlab speeds up development. Using Matlab for vision processing and control provides an option of using simulated vision during a flight test, increasing safety and allowing observance of flight characteristics during the control.

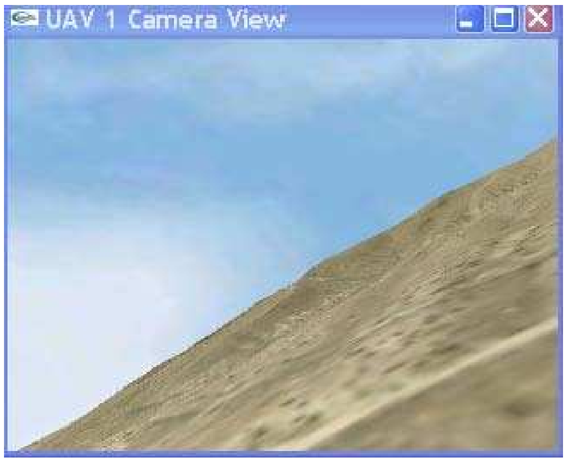
A.5 Graphical Interface

A.5.1 Matlab Plotting

Matlab has a large set of plotting functions useful for debugging, tuning, and evaluation. The Simulink plotting block is the place to store these plotting routines. Debugging and plotting routines are written by students as they continue research.



(a) UAV Chase View from the simulation is shown.



(b) UAV Camera view from the simulation is shown. The pixels from the camera view can be used in vision processing in the simulation.

Figure A.4:

Many of the routines are useful as and added to the Simulink library for use in other projects.

A.5.2 Aviones

We use another plotting interface called Aviones. Aviones is an OpenGL 3D environment that uses terrain elevation maps overlayed by satellite images of the terrain. The result is a virtual terrain matching any terrain for which elevation data is available. An example is shown in Fig A.4(a). This environment is available in Matlab through a mex function. The first call to the function initializes the window with the desired size, frame rate, terrain, etc. Another call to the function updates the state of the UAV. The result is smooth video of the UAV moving through the terrain.

The viewpoint in Aviones is adjustable. There are four viewpoints available: chase, camera, satellite, and groundtrack. The chase view is a viewpoint one meter behind the UAV with the same yaw angle as the UAV. Fig A.4(a) shows the chase view. The satellite view is simply a high altitude looking down on the terrain. The groundtrack views the UAV from the launch position. The most useful view for control is the camera view. The camera view simulates a camera at the elevation and azimuth angles given in the state vector. The pixel data from the camera view is sent to the video processing Simulink block for any required vision processing.

A.6 Flight Tests

We conduct flight tests on a regular basis. Some of these flight tests involve our obstacle avoidance algorithms. Obstacle avoidance is dangerous to test. If an obstacle is not detected properly, the airframe may crash into the obstacle, possibly damaging the airframe and the obstacle. In addition, most obstacles are close to the ground, requiring the UAV to fly at low altitudes and relying on accurate altitude measurements to prevent it from colliding with the ground. These dangers can be alleviated by first testing using hardware-in-the-loop vision provided by the simulation. The graphical interface section can generate camera data, process it, and send control commands to the autopilot in flight. This allows the UAV to fly at high altitudes, in an obstacle free area, while testing obstacle avoidance routines. If performance is adequate, then a flight test with real obstacles may be warranted.

Obstacle avoidance with hardware-in-the-loop vision has been conducted many times with this simulation architecture. An example is shown in Figure A.5. In this example, we use the obstacle avoidance routines described in Ref. [58]. As the vision processing detects the obstacles, it moves the obstacles to the edge of the camera field-of-view to avoid them. For two obstacles, each obstacle is moved to opposite sides, thus the UAV maneuvers between the obstacles. The flight was conducted safely at an altitude of 100 meters in an obstacle free area. This flight test allowed us to monitor flight performance, tune gains, and determine if a low altitude test with real obstacles is feasible. All control code was written in Matlab script, unchanged from the purely simulated tests.

A.7 Conclusion

In this chapter, we have described a simulator with the following properties,

1. modularity to support different airframes and research,

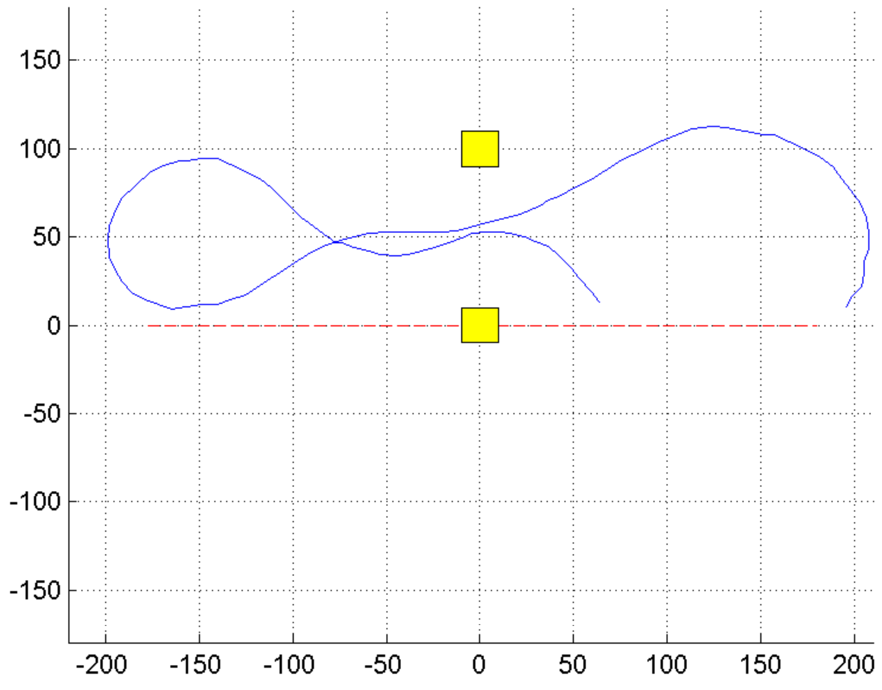


Figure A.5: The UAV avoids a virtual obstacle using the vision generated with the hardware-in-the-loop simulation. The solid line is the UAV path, the dashed line is the waypoint path, and the yellow blocks are obstacles.

2. fast research and development prototyping,
3. hardware-in-the-loop testing,
4. graphical interface with camera simulation for vision processing,
5. research standardization to prevent code loss when employees leave, and
6. support for flight testing with vision generation.

The simulator is a significant factor in fast development and prototyping of new research. It contributes to safe flight tests in situations where flight tests are dangerous by allowing testing at high altitudes and generated data. It also provides a standard research platform for employees.

Bibliography

[1] "Optics Planet," http://www.opticsplanet.com/.

[2] D. W. Casbeer, D. B. Kingston, R. W. Beard, T. W. McLain, S.-M. Li, and R. Mehra, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *International Journal of Systems Science*, vol. 37, no. 6, pp. 351–360, May 2006.

[3] H. Chang, C. Lee, Y. Lu, and Y. Hu, "P-slam: Simultaneous localization and mapping with environmental-structure prediction," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 281–293, April 2007.

[4] R. W. Beard and T. W. McLain, "Multiple UAV cooperative search under collision avoidance and limited range communication constraints," *IEEE Conference on Decision and Control*, pp. 25–30, December 2003.

[5] J.-H. Kim and S. Sukkarieh, "Airborne simultaneous localisation and map building," *Conference on Robotics and Automation*, pp. 406–411, 2003.

[6] J. Kim and S. Sukkarieh, "Robust multi-loop airborne SLAM in unknown wind environments," *Conference on Robotics and Automation*, pp. 1536–1541, 2006.

[7] P. Cheng, Z. Shen, and S. M. LaValle, "Using randomization to find and optimize trajectories for nonlinear systems," in *Proceedings of Annual Allerton Conference on Communications, Control, Computing*, 2000.

[8] J. Kim and J. P. Ostrowski, "Motion planning of aerial robot using rapidly-exploring random trees with dynamic constraints," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, September 2003, pp. 2200–2205.

[9] J. Byrne, M. Cosgrove, and R. Mehra, "Stereo based obstacle detection for an unmanned air vehicle," *International Conference on Robotics and Automation*, pp. 2830–2835, May 2006.

[10] J. Byrne and C. J. Taylor, "Expansion segmentation for visual collision detection and estimation," *International Conference on Robotics and Automation*, pp. 12–17, May 2009.

[11] R. Rezza, "Path following for air vehicles in coordinated flight," in *International Conference on Advanced Intelligent Mechatronics*, September 1999.

[12] J. Lee, R. Huang, A. Vaughn, X. Xiao, and J. K. Hedrick, "Strategies of path-planning for a uav to track a ground vehicle," in *Proceedings of the 2nd Anual Autonomous Intelligent Networks and Systems Conference*, 2003.

[13] S. Stolle and R. Rysdyk, "Flight path following guidance for unmanned air vehicles with pan-tilt camera for target observation," in *Digital Avionics Systems Conference*, 2003.

[14] R. Rysdyk, "Unmanned aerial vehicle path following for target observation in wind," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 5, pp. 1092–1100, September-October 2006.

[15] E. Frew, X. Xiao, S. Spry, T. McGee, Z. Kim, J. Tisdale, R. Sengupta, and J. K. Hendrick, "Flight demonstrations of self-directed collaborative navigation of small unmanned aircraft," in *AIAA 3rd Unmanned Unlimited Technical Conference*, September 2004.

[16] V. Stepanyan and N. Hovakimyan, "Visual tracking of a maneuvering target," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, August 2006.

[17] P. Thomasson, "Guidance of a roll-only camera for ground observation in wind," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 1, pp. 39–44, January-Feburary 1998.

[18] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," *Proceedings of the 2000 Congress on Evolutionary Computation*, vol. 1, pp. 256–263, July 2000.

[19] M. G. Park, J. H. Jeon, and M. C. Lee, "Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing," *IEEE International Symposium on Industrial Electronics*, vol. 3, pp. 1530–1535, June 2001.

[20] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transaction On Robotics and Automation*, vol. 12, no. 4, pp. 556–580, August 1996.

[21] R. Al-Hmouz, T. Gulrez, and A. Al-Jumaily, "Probabilistic road maps with obstacle avoidance in cluttered dynamic environment," *Intelligent Sensors, Sensor Networks and Information Processing Conference*, pp. 241–245, December 2004.

[22] J. Basch, L. Guibas, D. Hsu, and A. T. Nguyen, "Disconnection proofs for motion planning," *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1765–1772, 2001.

[23] R. Bohlin and L. Kavraki, "Path planning using lazy PRM," *IEEE International Conference on Robotics*, vol. 1, pp. 521–528, April 2000.

[24] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real time motion planning for agile autonomous vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 116–129, January-February 2002.

[25] M. Branicky, S. LaValle, K. Olson, and L. Yang, "Quasi-randomized path planning," *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1481–1487, 2001.

[26] S. LaValle and J. Kuffner, J.J., "Randomized kinodynamic planning," *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 473–479, May 1999.

[27] J. Kuffner, J.J. and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 995–1001, 2000.

[28] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. H. Reif, "Narrow passage sampling for probabilistic roadmap planning," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1105–1115, December 2005.

[29] T. Lozano-Perez, "Automation planning of manipulator transfer movements," *IEEE Transaction Systems, Man, and Cybernetics*, vol. 11, no. 10, pp. 681–698, 1981.

[30] D. Zhu and J. C. Latombe, "New heuristic algorithms for efficient hierarchical path planning," *IEEE Transaction On Robotics and Automation*, vol. 7, no. 1, pp. 9–20, August 1991.

[31] F. Lingelbach, "Path planning using probabilistic cell decomposition," *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 467–472, 2004.

[32] C. S. Sharp, O. Shakernia, and S. S. Sastry, "A vision system for landing an unmanned aerial vehicle," *Proceedings of the 2001 IEEE International Conference on Robotics; 8. Automation*, pp. 1720–1727, 2001.

[33] T. Kanade, O. Amidi, and Q. Ke, "Real-time and 3D vision for autonomous small and micro air vehicles," *43rd IEEE Conference on Decision and Control*, pp. 1655–1662, 2004.

[34] B. Sinopoli, M. Micheli, G. Donatot, and T. J. Koo, "Vision based navigation for an unmanned aerial vehicle," *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, pp. 1757–1764, 2001.

[35] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, B. R. Donald, K. M. Lynch, and D. Rus, Eds. Wellesley, MA: A. K. Peters, 2001, pp. 293–308.

[36] J. B. Saunders, B. Call, A. Curtis, R. W. Beard, and T. W. McLain, "Static and dynamic obstacle avoidance in miniature air vehicles," *AIAA Infotech@Aerospace*, September 2005.

[37] J. Saunders and R. Beard, "Obstacle avoidance using circular paths," *Guidance, Navigation and Control Conference*, 2007.

[38] D. R. Nelson, B. Barber, T. W. McLain, and R. W. Beard, "Vector field path following for miniature air vehicles," *IEEE Transactions on Robotics*, vol. 23, pp. 519–529, 2007.

[39] "Procerus Technologies," http://procerusuav.com/.

[40] R. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain, and M. A. Goodrich, "Autonomous vehicle technologies for small fixed-wing uavs," *Journal of Aerospace Computing, Information, And Communication*, vol. 2, pp. 92–102, January 2005.

[41] M. S. Geyer and E. N. Johnson, "An integrated top-down approach to obstacle avoidance for unmanned aerial vehicles," *AIAA Guidance, Navigation and Control Conference and Exhibit*, August 2007.

[42] M. Bertozzi, A. Broggi, and A. Fascioli, "Vision-based intelligent vehicles: State of the art and perspectives," *Robotics and Autonomous Systems*, pp. 1–16, 2000.

[43] J. Kim and Y. Suga, "An omnidirectional vision-based moving obstacle detection in mobile robot," *International Journal of Control, Automation, and Systems*, vol. 5, no. 6, pp. 663–673, December 2007.

[44] I. Ohya, A. Kosaka, and A. Kak, "Vision-based navigation by a mobile robot with obstacle avoidanceusing single-camera vision and ultrasonic sensing," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 969–978, August 1998.

[45] L. Kavraki, M. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transaction On Robotics and Automation*, vol. 14, no. 1, pp. 166–171, Feburary 1998.

[46] N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, MN, 1996, pp. 113–120.

[47] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[48] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 2000, pp. 995–1001.

[49] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” October 1998, tR 98-11, Computer Science Dept., Iowa State University.

[50] S. M. LaValle and J. J. Kuffner, “Randomized kinodynamic planning,” vol. 20, no. 5, pp. 378–400, May 2001.

[51] M. Saha and J.-C. Latombe, “Finding narrow passages with probabilistic roadmaps: the small step retraction method.” International Conference on Intelligent Robots and Systems, August 2005, pp. 622–627.

[52] F. Schwarzer, M. Saha, and J.-C. Latombe, “Adaptive dynamic collision checking for single and multiple articulated robots in complex environments,” *IEEE Transaction On Robotics*, vol. 21, no. 3, pp. 338–353, June 2005.

[53] R. Ghosh and C. Tomlin, “Maneuver design for multiple aircraft conflict resolution,” *Proceedings of the American Control Conference*, June 2000.

[54] A. Richards, J. Bellingham, M. Tillerson, and J. How, “Coordination and control of multiple UAVs,” *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2002.

[55] J. Barraquand, B. Langlois, and J. C. Latombe, “Numerical potential field techniques for robot path planning,” *IEEE Transaction On Systems, Man, and Cybernetics*, vol. 22, no. 2, pp. 224–241, March/April 1992.

[56] T. Lozano-Perez and M. Wesley, “An algorithm for planning collision free paths among polyhedral obstacles,” *Communication of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[57] P. O’Dunlaing and C. Yap, “A retraction method for planning the motion of a disc,” *Journal of Algorithms*, vol. 6, no. 1, pp. 104–111, 1982.

[58] J. Saunders and R. Beard, “Reactive vision based obstacle avoidance with camera field of view constraints,” *Guidance, Navigation, and Control Conference*, 2008.

[59] J. Saunders, R. Beard, and J. Byrne, “Vision-based reactive multiple obstacle avoidance for micro air vehicles,” *American Control Conference*, 2009.

[60] T. Netter and N. Franceschini, “A robotic aircraft that follows terrain using a neuromorphic eye,” *Conference Intelligent Robots and System*, 2002.

[61] Y. Bar-Shalom, “Update with out-of-sequence measurements in tracking: Exact solution,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 3, pp. 769–777, July 2002.

[62] L. Hong, S. Cong, and D. Wicker, “Multirate interacting multiple model (MRIMM) filtering with out-of-sequence GMTI data,” *IEEE Proceedings - Radar, Sonar and Navigation*, vol. 150, no. 5, pp. 333–343, October 2003.

[63] M. Mallick, J. Krant, and Y. Bar-Shalom, "Multi-sensor multi-target tracking using out-of-sequence measurements," *Fifth International Conference on Information Fusion*, vol. 38, no. 17, pp. 135–142, October 2002.

[64] K. Zhang, X. Li, and Y. Zhu, "Optimal update with out-of-sequence measurements," *IEEE Transactions on Signal Processing*, vol. 53, no. 6, pp. 1992–2004, June 2005.

[65] R. Frezza and C. Altafini, "Autonomous landing by computer vision: an application of path following in se(3)," in *Conference on Decision and Control*, December 2004.

[66] I. H. Wang, V. N. Dobrokhodov, I. I. Kaminer, and K. D. Jones, "On vision-based target tracking and range estimation for small uavs," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, August 2005.

[67] L. Ma, C. Cao, V. Dobrokhodov, I. Kaminer, K. Jones, and I. Kitsios, "Rapid motion estimation of a target moving with time-varying velocity," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, August 2007.

[68] L. Ma, C. Cao, N. Hovakimyan, C. Woolsey, V. Dobrokhodov, and I. Kaminer, "Development of a vision-based guidance law for tracking a moving target," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, August 2007.

[69] A. D.Wu, E. N. Johnson, and A. A. Proctor, "Vision-aided inertial navigation for flight control," *Journal Of Aerospace Computing, Information, and Communication*, vol. 2, no. 9, pp. 348–360, 2005.

[70] J. Redding, T. W. McLain, R. W. Beard, and C. Taylor, "Vision-based target localization from a fixed-wing miniature air vehicle," *Journal Of Intelligent and Robotic Systems*, vol. 2, no. 9, pp. 348–360, 2005.

[71] J. Saunders and R. Beard, "Tracking a target in wind using a micro air vehicle with a fixed angle camera," *American Control Conference*, 2008.

[72] C. J. Schumacher, P. R. Chandler, M. Pachter, and L. Pachter, "Constrained optimization for uav task assignment," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2004.

[73] C. Schumacher, P. Chandler, and S. Rasmussen, "Task allocation for wide area search munitions," *Proceedings of the American Control Conference*, 2002.

[74] P. Chandler, M. Pachter, S. Rasmussen, and C. Schumacher, "Multiple task assignment for a UAV team," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2002.

[75] S. Rasmussen and T. Shima, "Branch and bound tree search for assigning cooperating UAVs to multiple tasks."

[76] J. Saunders, S. Rasmussen, and C. Schumacher, "Combining collision avoidance and operator workload reduction with cooperative task assignment and path planning," *American Control Conference*, 2007.

[77] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal position," *American Journal of Math*, vol. 79, pp. 497–516, 1957.

[78] J. C. Hill, J. K. Archibald, W. C. Stirling, and R. L. Frost, "A multi-agent system architecture for distributed air traffic control," *Guidance, Navigation, and Control Conference*, pp. 1–11, 2005.

[79] J. M. Grasmeyer and M. T. Keennon, "Development of the black widow micro air vehicle," *39th AIAA Aerospace Sciences Meeting and Exhibit*, no. AIAA Paper No. 2001-0127, January 2001.

[80] K. Kaiser, N. Gans, and W. Dixon, "Localization and control of an aerial vehicle through chained, vision-based pose reconstruction," *American Control Conference*, pp. 5934–5939, 2007.

[81] J. W. Grzywna, A. Jain, J. Plew, and M. C. Nechyba, "Rapid development of vision-based control for MAVs through a virtual flight testbed," *International Conference on Robotics and Automation*, 2005.