REAL TIME TRAJECTORY OPTIMIZATION FOR VISION BASED NAVIGATION WITH AEROBATIC FIXED-WING VEHICLES

by

Adam Polevoy

A thesis submitted to Johns Hopkins University

in conformity with the requirements for the degree of

Master of Science in Engineering

Baltimore, Maryland

May, 2020

Abstract

Fixed-wing unmanned aerial vehicles (UAVS) pose advantages in energy efficiency, endurance, and speed, but also pose disadvantages in maneuverability. These maneuverability challenges can be addressed by exploiting high angle of attack maneuvers. However, navigation with fixed-wing UAVs in constrained spaces is still extremely difficult when the system state and environment are unknown.

This essay investigates the use of vision sensors in autonomous navigation of aerobatic fixed-wing UAVs. Perception aware NMPC is explored through the integration of a visibility metric into the trajectory optimization problem. Additionally, a novel frontier-based NMPC method, which improves obstacle avoidance capabilities while mapping, is proposed. These methods are evaluated in a realistic real-time simulation.

Primary Reader and Advisor: Dr. Joseph Moore

Secondary Reader: Dr. Simon Leonard

Acknowledgments

I would like to thank Dr. Joseph Moore, my research advisor, for his guidance and advice throughout the research and writing process. I am extremely grateful for his willingness to dedicate his time to help guide this research. I would also like to thank all the people at the Applied Physics Laboratory whose expertise helped to make this research possible. Additionally, thank you to Dr. Simon Leonard for his helpful and constructive suggestions. These suggestions have been key to the refinement of this essay.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Fixed-wing unmanned aerial vehicles (UAVs) pose significant advantages in energy efficiency, endurance, and speed when compared to rotary wing UAVs. For this reason, fixed-wings are particularly well suited for tasks requiring long distance flight. However, for tasks requiring navigation in constrained spaces, fixed-wing UAVs pose serious challenges. Constrained spaces, for the purposes of this essay, are defined as environments which impose substantial constrains on vehicle dynamics. In conventional flight regimes, these vehicles possess large minimum turn radii; in more aggressive flight regimes spanning larger flight envelopes, these vehicles lack the differentially-flat representations available to their rotary-wing counterparts. Together, these factors make achieving fast collision-free flight in tight spaces with fixed-wing UAVs a challenging underactuated control problem.

Despite these challenges, there are cases in which fixed-wing UAVs are an attractive choice over rotary-wing UAVs. Such scenarios exist when high

maneuverability is required intermittently across long distances. For example, it may be necessary to travel long distances prior to carrying out a mission that requires maneuvering around obstacles. Similarly, when navigating in urban environments or tunnels, a vehicle may be required to navigate both long straightaways and tight turns. A fixed-wing UAV would be ideal for these scenarios, if it could achieve both low-energy, low angle-of-attack flight and more maneuverable post-stall flight to execute tight turns. In (Basescu and Moore, 2020), the authors demonstrate a fixed-wing UAV control system capable of such behavior, but assume full-state feedback and a known environment.

While the control problem addressed in (Basescu and Moore, 2020) is challenging, it is greatly simplified by ignoring the impacts of onboard perception to agile fixed-wing UAV navigation. Rapid attitude changes are necessary for maneuverable post-stall flight. However, this results in deleterious effects for many onboard sensors, such as cameras. The control problem also depends on the seeding of previously generated trajectories to warm start the optimization problem. This can be difficult to perform successfully in unknown spaces since new obstacles may be revealed and potentially make previously generated trajectories infeasible. Additionally, sensors commonly used for mapping and SLAM, such as LIDAR, are too heavy for many fixed-wing models. These issues motivate investigation into the use of vision sensors in autonomous navigation of aerobatic fixed-wing UAVs in constrained environments.

1.2 Related Work

In the past decade, significant research has been dedicated towards improving control strategies for fixed-wing UAVs. Optimal trajectory generation has been a key focus (Milam, Franz, and Murray, 2002). Executing aerobatic and post stall maneuvers is especially important for improving overall maneuverability, and thus obstacle avoidance, capabilities. Agile turnaround maneuvers, which have been investigated for this purpose, provide a good example (Matsumoto et al., 2010). Agile turnaround utilizes post stall maneuvers to reduce travel distance or turn radius to improve obstacle avoidance capabilities.

Nonlinear model-predictive control (NMPC) has provided a robust method suited for fixed-wing UAV control in recent research. Nonlinear model-predictive controllers (NMPC) and linear quadratic regulator trees (LQR-Trees) have been utilized to perform perching, a complex post stall maneuver (Moore, Cory, and Tedrake, 2014). NMPC has also been utilized to control fixed-wing UAVs in environments with large disturbances, such as high wind conditions (Stastny, Dash, and Siegwart, 2017). NMPC methods have also been extended to account for state uncertainty; recently, Tube NMPC was introduced as a method for belief space motion planning for nonlinear systems (Garimella et al., 2018). This essay directly builds upon the direct NMPC approach proposed in (Basescu and Moore, 2020).

Uncertainty in vehicle dynamics, environmental knowledge, and state estimation all pose challenges to control. Dadkhah provided a survey of common existing methods, as of 2011, to deal with these uncertainties (Dadkhah and

Mettler, 2012). However, simultaneously performing mapping, state estimation, and navigation for fixed-wing UAVs is less prevalent in research. Lidar, which is commonly used for mapping and localization in robotics systems, is usually too heavy for use on smaller fixed-wing UAVs. While GPS sensors are useful for localization, they they cannot be used in GPS denied locations. Recently, stereo vision has been investigated for use in mapping and state estimation (Mourikis and Roumeliotis, 2007a, Sun et al., 2018, Brossard, Bonnabel, and Barrau, 2018). Stereo cameras can be much lighter than lidar sensors, which makes them well suited for use on UAVs.

Simultaneous mapping and navigation is especially challenging when high velocity motion occurs in constrained environments. Within the past few years, machine learning methods have been proposed to address high speed navigation. Machine learning models have been trained to approximate how the short path heuristic of control differs from the optimal control strategy (Richter and Roy, 2017). This was useful for improving path planning around corners in an unknown map for systems with narrow fields of views. One source proposes the use of deep neural networks for predicting unknown map regions (Katyal et al., 2019). This could allow robots to make navigation decisions based upon predicted obstacle locations. Bansal proposes the use of Convolutional Neural Networks (CNNs) on camera images to select waypoints that would avoid obstacles (Bansal et al., 2019). This method has been combined with Hamilton-Jacobi reachability analysis to ensure safe autonomous navigation (Bajcsy et al., 2019).

Active sensing can be defined as "weighting future information gain and

cost" to decide future actions (Mihaylova et al., 2003). Mihaylova provides an overview of possible methods for incorporating covariance matrices and probability density functions into optimization problems to achieve this goal. Active sensing can be used to improve information gathering in order to more efficiently explore maps and to minimize state uncertainty. Leung et al. performed NMPC while minimizing the trace of a Extended Kalman Filter (EKF) covariance matrix as a method of information gathering (Leung et al., 2006). Ny and Pappas proposed a "suboptimal non-greedy trajectory optimization scheme" that could tackle active sensing problems more efficiently (Le Ny and Pappas, 2009). Recently, Cauchy-Schwarz Quadratic Mutual Information (CSQMI) has been used to improve map exploration efficiency (Charrow et al., 2015). In this method, control actions were chosen such that the CSQMI was maximized. Frey introduces methods for efficiently performing observability-aware trajectory optimization and provides background on common uncertainty metrics (Frey, Steiner, and How, 2019).

1.3 Contribution

This essay explores the use of vision sensors in autonomous navigation of aerobatic fixed-wing UAVs in constrained environments. Specifically, the issues of trajectory generation in unknown constrained environments and state uncertainty associated with high speed and post stall motion are addressed. This research integrates and augments existing control, mapping, and state estimation strategies with data from vision sensors to improve fixed-wing UAV obstacle avoidance during autonomous flight.

The first contribution of this essay is the integration, and evaluation, of a visibility metric with direct NMPC of a fixed-wing UAV. In this proposed control strategy, obstacle avoidance and dynamical feasibility are formulated as hard constraints while improved visibility of environmental features are formulated as soft costs. The second contribution of this essay is a novel frontier-based NMPC method. This method aims to improve obstacle avoidance capabilities of fixed-wing UAVs in unknown environments. The effectiveness of the visibility cost in improving state estimation and of the frontier-based NMPC in improving obstacle avoidance are evaluated in simulation.

A final contribution is the simulation of state-estimation, mapping and planning in parallel to demonstrate that real-time performance can be achieved.

1.4 Organization

Chapter 2 contains a review of core concepts that are utilized in the rest of the essay. Chapter 3 contains an overview of the system dynamics and sensors. Chapter 4 reviews the NMPC framework, proposed in (Basescu and Moore, 2020) that is built upon in the following chapters. Chapter 5 addresses state estimation and experiments with a visibility cost function to improve state estimation. Chapter 6 expands the NMPC framework for planning with frontier based planning for improved performance in an unknown map. Chapter 7 presents simulation results for performing planning, mapping, and state estimation in parallel. Chapter 8 concludes this essay.

Chapter 2

Concept Overview

This chapter provides an overview of base concepts utilized in this essay. This essay builds upon a control strategy that is achieved in four stages: RRT generation, spline-based smoothing, direct trajectory optimization, and local linear feedback control. Additionally, 3D stereo reconstruction and mapping are utilized throughout the essay.

2.1 Rapidly-exploring Random Tree

Rapidly-exploring Random Tree (RRT) is a sampling-based motion planning method (LaValle, 1998). This method has a relatively simple implementation and is well suited to high dimensional spaces.

The algorithm is specified in Algorithm 1.

2.2 Direct Transcription

Direct Transcription (Canon, Cullum Jr, and Polak, 1970) is a commonly used method for optimal trajectory generation. Trajectories are discretized

Algorithm 1 GENERATE_RRT(x_{init}, K, Δt)

τ.init(x_{init})
for $k = 1$ to K **do**
  $x_{rand} \leftarrow$ RANDOM_STATE()
  $x_{near} \leftarrow$ NEAREST_NEIGHBOR(x_{rand}, τ)
  $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near})
  $x_{new} \leftarrow$ NEW_STATE(x_{near},u,Δt)
  τ.add_vertex(x_{new})
  τ.add_edge(x_{near},x_{new},u)
end for
Return τ

into knot points, where each knot point is defined by the state $x_k = x(t_k)$ and the control $u_k = u(t_k)$. At each knot point, a cost function is evaluated and constraints are imposed. Different cost functions can be chosen to optimize trajectories over different metrics, such as time or distance to goal. This method of optimal trajectory generation is especially useful for imposing constraints directly on both controls and states.

A simple direct transcription problem can be written as

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & J(\mathbf{x},\mathbf{u},\mathbf{t}) = \phi(x_N, t_N) + \sum_{i=0}^{N-1} L_k(x_k, x_{k+1}, u_k, u_{k+1}) \\ \text{s.t.} \quad & S_k(x_k, x_{k+1}, u_k, u_{k+1}) = 0 \; \forall \; k \\ & c(x_k, u_k, t_k) \leq 0 \; \forall \; k \\ & \psi(x_k, u_k, t_k) \leq 0 \end{aligned} \tag{2.1}$$

where L_k is the cost function, ϕ_k is the final cost function, S_k is the discrete approximation of the dynamics, and c and ψ are additional constraints (Kobilarov, 2019b).

2.3 Time-Varying Linear Quadratic Regulator

Time-Varying Linear Quadratic Regulator (TVLQR) is a widely used trajectory tracking method. The goal of this method is to generate a control, $\mathbf{u}$, which stabilizes the system state, $\mathbf{x}$, to a given trajectory. The system has dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ and the trajectory is defined by $\mathbf{x}_0$ and $\mathbf{u}_0$. The cost function to be minimized can be written as

$$J = \frac{1}{2}\|\mathbf{x}(t_f) - \mathbf{x}_0(t_f)\|^2_{\mathbf{Q}_f} + \frac{1}{2}\int_{t_0}^{t_f} (\|\mathbf{x}(t) - \mathbf{x}_0(t)\|^2_{\mathbf{Q}} + \|\mathbf{u}(t) - \mathbf{u}_0(t)\|^2_{\mathbf{R}})$$

Where $\mathbf{Q}$, $\mathbf{Q_f}$, and $\mathbf{R}$ are costs on the state, final state, and control errors respectively. The system is linearized as $\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t)$ using

$$\mathbf{A}(t) = \frac{\partial \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0(t))}{\partial \mathbf{x}}$$

$$\mathbf{B}(t) = \frac{\partial \mathbf{f}(\mathbf{x}_0(t), \mathbf{u}_0(t))}{\partial \mathbf{u}}$$

The stabilizing control signal is calculated as

$$\mathbf{u}(t, \mathbf{x}(t)) = \mathbf{K}(\mathbf{x}(t) - \mathbf{x}_0(t)) + \mathbf{u}_0(t). \tag{2.2}$$

where $\mathbf{K}$ is calculated by integrating the Riccati ODE

$$-\dot{\mathbf{S}}(t) = \mathbf{A}(t)^T\mathbf{S}(t) + \mathbf{S}(t)\mathbf{A}(t) - \mathbf{S}(t)\mathbf{B}(t)\mathbf{R}^{-1}\mathbf{B}(t)^T\mathbf{S}(t) + \mathbf{Q}$$

$$\mathbf{S}(t_f) = \mathbf{Q}_f$$

backwards in time from $t = t_f$ to $t = t_0$ (Kobilarov, 2019a).

2.4 Nonlinear Model Predictive Control

Nonlinear Model Predictive Control (NMPC), also referred to as nonlinear receding horizon control, is an iterative optimization-based control strategy for nonlinear systems (Findeisen and Allgöwer, 2002). The control problem is formulated to solve an optimal control problem over a finite prediction horizon, in which the system dynamics are predicted. The control problem is recalculated at a predetermined frequency, thus continuously replanning the control. This replanning is useful for dealing with system disturbance and constantly updating environment observations.

2.5 Unscented Kalman Filtering

The unscented kalman filter (UKF) (Julier and Uhlmann, 1997) addresses approximation issues of the extended kalman filter (EKF). Since EKF linearizes the dynamics using a first order approximation, it can diverge when the system nonlinearities are large. UKF utilizes the unscented transformation, which can calculate the statistics of a random variable after it undergoes a nonlinear transformation. For a random variable $\mathbf{x}$ with mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}_{xx}$, sigma points χ_i and weights W_i are generated.

$$\begin{aligned} \chi_0 &= \bar{\mathbf{x}} & W_0 &= \frac{k}{n+k} \\ \chi_i &= \bar{\mathbf{x}} + (\sqrt{(n+k)\mathbf{P}_{xx}} & W_i &= \frac{1}{2(n+k)} \\ \chi_{i+n} &= \bar{\mathbf{x}} - (\sqrt{(n+k)\mathbf{P}_{xx}}\,)_i & W_{i+n} &= \frac{1}{2(n+k)} \end{aligned} \tag{2.3}$$

where there are $2n + 1$ sigma points and $k \in \mathbb{R}$ is a scaling parameter. Each sigma point is passed through the nonlinear transform to obtain the transformed sigma points. The mean of the transformed random variable is the weighted average of the transformed sigma points, and the covariance is the weighted outer product of the transformed sigma points.

The unscented transform is integrated into the kalman filter by augmenting the state and covariance with process and noise terms. Sigma points are drawn from the resulting random variable. When performing the prediction and update steps, the state and covariance are updated using the unscented transformation, rather than using linearized process and measurement models as done in EKF

2.6 Stereo 3D Reconstruction

Stereo cameras are able to estimate the range to objects by comparing the images from two cameras. This allows for the 3D reconstruction of an environment from stereo images. In order to reconstruct a 3D scene, disparity images are first computed. Konolige provides details of a workflow for constructing disparity images from stereo images (Konolige, 1998).

1. Geometry correction: Correct distorted stereo images into standard form
2. Image Transformation: Transform each pixel of grayscale images into more appropriate form
3. Area Correlation: Compare small areas around pixels to others in a search window

4. Extrema Extraction: Best matches are determined from correlation, resulting in a disparity image.

5. Post-filtering: Filters can be used to reduce noise on disparity image.

The resulting disparity image indicates the distance, in pixels, between corresponding points in the two stereo camera images. Given the camera intrinsics and the pose transformation between the two matrices, depth can be determined from the disparity image. For a simple stereo camera, where the cameras are offset from each other by a horizontal baseline, b, depth z_i can be calculated from disparity d_i as

$$z_i = \frac{bf}{d_i} \tag{2.4}$$

for each pixel (Shen, 2020). f represents the focal length of the cameras.

2.7 OctoMap

An OctoMap (Hornung et al., 2013) is a 3D mapping approach that makes use of octrees and probabilistic occupancy estimation. An octree is a data structure that can be used to represent 3D space. Each parent node is a cube, and the child nodes split the cube into eight smaller cubes. If all children nodes have the same state, occupied or unoccupied, then the parent can be assigned that state. This data structure makes querying occupancy of 3D space more efficient.

OctoMap stores occupancy probabilistically in octree nodes as a method of dealing with sensor noise and changing environments. The probability of

occupancy of node $P(n)$ given measurement z_t is calculated as

$$L(n|z_{1:t}) = L(n|z_{1:t-1}) + L(n|z_t) \tag{2.5}$$

$$L(n) = log\left[\frac{P(n)}{1 - P(n)}\right]$$

A threshold is applied to this probability to determine occupancy of a node.

Chapter 3

System Overview

3.1 Dynamics

The fixed-wing dynamics used in this essay are the same as those found in (Basescu and Moore, 2020). Here follows a brief review of those system dynamics.

The equations of motion are described as

$$\begin{aligned}
\dot{\mathbf{r}} &= \mathbf{R}_b^r \mathbf{v} \\
\dot{\boldsymbol{\theta}} &= \mathbf{R}_\omega^{-1} \boldsymbol{\omega} \\
\dot{\boldsymbol{\delta}} &= \mathbf{u}_{cs} \\
\dot{\delta}_t &= a_t \delta_t + b_t u_t \\
\dot{\mathbf{v}} &= \mathbf{f}/m - \boldsymbol{\omega} \times \mathbf{v} \\
\dot{\boldsymbol{\omega}} &= \mathbf{J}^{-1}(\mathbf{m} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega})
\end{aligned} \tag{3.1}$$

The input to the system is $\mathbf{u} = \{\omega_{ar}, \omega_{al}, \omega_e, \omega_r, u_t\}^T$ and the state is $\mathbf{x} = \{\mathbf{r}^T, \boldsymbol{\theta}^T, \boldsymbol{\delta}^T, \dot{\delta}_t, \mathbf{v}^T, \boldsymbol{\omega}^T\}^T$, where

- $\mathbf{r} = \left[x_r, y_r, z_r\right]^T$ is the center of mass position in the world frame
- $\boldsymbol{\theta} = \left[\phi, \theta, \psi\right]^T$ are the (z-y-x) euler angles
- $\boldsymbol{\delta} = \left[\delta_{ar}, \delta_{al}, \delta_e, \delta_r\right]^T$ are the control surface deflections
- $\mathbf{v} = \left[v_x, v_y, v_z\right]^T$ is the center of mass velocity in the body-fixed frame
- $\boldsymbol{\omega} = \left[\omega_x, \omega_y, \omega_z\right]^T$ is the angular velocity in the body-fixed frame

The forces acting on the vehicle, $\mathbf{f}$, and the moments acting on the center of mass, $\mathbf{m}$, are calculated as follows

$$
\begin{aligned}
\mathbf{f} &= \sum_i \mathbf{R}_{s_i}^b \mathbf{f}_{s_i} + \mathbf{R}_b^{r\,T} g \mathbf{e}_{z_r} + \sum_i \mathbf{R}_t^b \mathbf{f}_t \\
\mathbf{m} &= \sum_i (\mathbf{r}_{s_i} \times \mathbf{R}_s \mathbf{f}_{s_i}), \\
\mathbf{f}_s &= f_{n,s_i} \mathbf{e}_{y_s} = \frac{1}{2} C_n \rho |\mathbf{v}_{s_i}|^2 S_i \mathbf{e}_{y_s} \\
\mathbf{v}_{s_i} &= \mathbf{R}_{s_i}(\mathbf{v} + \boldsymbol{\omega} \times \mathbf{r}_h + \gamma_i \mathbf{v}_{bw}) + (\mathbf{R}_{s_i} \boldsymbol{\omega} + \boldsymbol{\omega}_{s_i}) \times \mathbf{r}_{s_i} \\
\mathbf{v}_{bw} &= v_{bw} \mathbf{e}_x = \sqrt{\|\mathbf{v}_p\|_2^2 + \frac{2\delta_t}{\rho S_{disk}}} - \|\mathbf{v}_p\|_2. \\
\mathbf{r}_{s_i} &= l_h \mathbf{e}_x + \mathbf{R}_s (l_s \mathbf{e}_{x_s})
\end{aligned}
\tag{3.2}
$$

Listed below are definitions for the referenced variables.

- m is the mass

- $\mathbf{J}$ is the inertia tensor (about the center of mass)
- $\mathbf{R}_b^r$ is the rotation from the body-fixed frame to the world frame
- $\mathbf{R}_{\boldsymbol{\omega}}$ is the rotation which maps the euler angle rates to an angular velocity in body-fixed frame.
- $\mathbf{f}_{s_i}$ are the forces due to the aerodynamic surfaces
- $\mathbf{f}_t$ are the forces due to the propeller, $\begin{bmatrix}\delta_t & 0 & 0\end{bmatrix}^T$
- $\mathbf{R}_{t_i}^b$ is the rotation of the thrust source with respect to the body fixed frame
- $\mathbf{R}_{s_i}^b$ is the rotation of the aerodynamic surface reference frame with respect to the body fixed frame
- $\mathbf{r}_h$ is the displacement from the vehicle center of mass to the aerodynamic surface "hinge" point
- $\mathbf{R}_{s_i}$ is the rotation that transforms vectors in the body-fixed frame into the aerodynamic surface frame
- $\boldsymbol{\omega}_{s_i}$ is the rotation rate of the aerodynamic surface in the aerodynamic surface frame
- $\mathbf{v}_{bw}$ is the velocity due to the backwash of the propeller.
- δ_t is the thrust input
- S_{disk} is the area of the actuator disk
- ρ is the density of air

- $\mathbf{v}_p$ is the velocity at the propeller
- γ is an empirically determined backwash velocity coefficient.
- $C_n = 2 \sin \alpha_w$ comes from the flat plate model in (Hoerner, 1985)
- $g\mathbf{e}_{z_r}$ is the gravity vector in the world frame
- $\mathbf{R}_s$ is the rotation between the aerodynamic surface frame and the body fixed frame
- l_h is the length of the from the center of gravity to the surface hinge
- l_s is the length of the surface hinge to the surface center
- $\mathbf{e}_x$ is the unit vector in the x direction of the body frame
- $\mathbf{e}_{x_s}$ is the unit vector in the x direction of the surface frame

3.2 Sensors

The following sensors were simulated to perform mapping and state estimation.

3.2.1 LIDAR

While the main goal of this essay is implement robust navigation with vision based sensors, a LIDAR sensor was simulated for algorithm development. While most LIDAR sensors are too heavy for use on a small fixed-wing UAV, LIDAR is relatively simple compared to camera sensors. LIDAR often has much large ranges and less noise than cameras and are commonly used to

perform mapping, and simultaneous mapping and localization (SLAM), on robotic systems. Thus, in simulation, LIDAR provides a simple sensor for use in algorithm testing and development. The LIDAR sensor simulated in this essay is attached to a gimbal on the fixed wing; the roll and pitch of the sensor are fixed to zero. Therefore, the lidar stays horizontal.

3.2.2 Stereo Camera

Stereo camera sensors are, generally, much lighter than LIDAR; therefore, they are well suited to fixed wing UAVs. They are not only more complex to simulate, but also more complex algorithms must be implemented to utilize them for mapping and state estimation. While an RGBD camera may also be applicable, the state estimate algorithm utilized in this essay, UKF S_MSCKF, uses a stereo camera for state estimation. Thus, to simplify the simulation, the stereo camera is utilized for both state estimation and mapping. The simulated stereo camera is also attached to a gimbal on the fixed wing.

3.2.3 Inertial Measurement Unit

An Inertial Measurement Unit (IMU) is simulated for use in state estimation. It outputs the linear acceleration and angular velocity of the system. The simulated IMU has added gaussian noise.

Chapter 4

Nonlinear Model Predictive Control

This essay expands upon the NMPC approach proposed in (Basescu and Moore, 2020). An overview of this approach, utilized in a known map with a known state, is reviewed here. The aim of this control strategy is to fly the fixed-wing from an initial state to a goal state without colliding into obstacles. The control strategy achieves this in four stages: RRT generation, spline-based smoothing, direct trajectory optimization, and local linear feedback control. During each control interval, a trajectory is planned to a time horizon, which is selected along a smoothed RRT path. This occurs at a control frequency of 5 Hz.

4.1 RRT Generation

Once a new control is to be generated, the system is simulated forward by the control period, 0.02 sec, to determine what it's approximate state will be once the control trajectory is generated. From the position of this state to the

position of the goal state, an RRT is generate with a 10% bias towards the goal position. The RRT is generated in three dimensional Eulidean space (x, y, z). The path from the start to goal positions is pruned.

4.2 Spline-Based Smoothing

G2 Continuous Cubic Bézier Spiral Path Smoothing (G2CBS) is used to create a continuous, smooth curve, $F(s)$ from the RRT path. Details of this approach are presented in (Yang and Sukkarieh, 2010).

The path is reparametrized in terms of time. This aids in picking a more accurate time-horizon endpoint for use in direct trajectory optimization. Curvature of the path is calculated as,

$$\kappa(s) = \frac{\sqrt{(z''y' - y''z')^2 + (x''z' - z''x')^2 + (y''x' - x''y')^2}}{(x'^2 + y'^2 + z'^2)^{\frac{3}{2}}}$$

where

$$F'(s) = \begin{bmatrix} x' & y' & z' \end{bmatrix} \quad and \quad F''(s) = \begin{bmatrix} x'' & y'' & z'' \end{bmatrix}$$

Curvature is mapped to velocity as,

$$v(s) = \frac{d\mathbf{x}}{dt}(s) = v_{max} - \kappa(s) * m, s \in [0, s_p]$$

where $\mathbf{x}(s)$ is the full path as a function of s, v_{max} is the maximum velocity for the kinematic model, m is the linear kinematic mapping parameter which relates curvature to velocity, and s_p is the overall path length. The path is then reparametrized by time with,

$$t = \int_0^s \frac{1}{v(s)} ds, s \in [0, s_p]$$

.

4.3 Direct Trajectory Optimization

In this context, trajectory generation is formatted as a feasibility problem where constraints are imposed on the trajectory dynamics and the end state. This feasibility problem can be expressed as a direct transcription problem. Direct transcription is utilized to generate a dynamically feasible and obstacle free path from a start state to an end state (Pardo et al., 2016). The start state is determined by simulating the current state forward, as described above. The end state is determined by evaluating the position and velocity at time horizon T_H along the time parameterized smoothed RRT path.

This feasibility problem is written as

$$
\begin{aligned}
\min_{\mathbf{x}_k, \mathbf{u}_k, h} \quad & 0 \\
\text{s.t.} \quad & \forall k \in \{0, \dots N\} \text{ and} \\
& \mathbf{x}_k - \mathbf{x}_{k+1} + \frac{h}{6.0}(\dot{\mathbf{x}}_k + 4\dot{\mathbf{x}}_{c,k} + \dot{\mathbf{x}}_{k+1}) = 0 \\
& \mathbf{x}_f - \boldsymbol{\delta}_f \leq \mathbf{x}_N \leq \mathbf{x}_f + \boldsymbol{\delta}_f \\
& \mathbf{x}_i - \boldsymbol{\delta}_i \leq \mathbf{x}_0 \leq \mathbf{x}_i + \boldsymbol{\delta}_i \\
& \mathbf{x}_{min} \leq \mathbf{x}_k \leq \mathbf{x}_{max}, \;\; \mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} \\
& d(\mathbf{x}) \geq r \\
& h_{min} \leq h \leq h_{max}
\end{aligned}
\tag{4.1}
$$

where

$$\dot{\mathbf{x}}_k = \mathbf{f}(t, \mathbf{x}_k, \mathbf{u}_k), \;\; \dot{\mathbf{x}}_{k+1} = \mathbf{f}(t, \mathbf{x}_{k+1}, \mathbf{u}_{k+1})$$

$$\mathbf{u}_{c,k} = (\mathbf{u}_k + \mathbf{u}_{k+1})/2$$

$$\mathbf{x}_{c,k} = (\mathbf{x}_k + \mathbf{x}_{k+1})/2 + h(\dot{\mathbf{x}}_k - \dot{\mathbf{x}}_{k+1})/8$$

$$\dot{\mathbf{x}}_{c,k} = \mathbf{f}(t, \mathbf{x}_{c,k}, \mathbf{u}_{c,k}). \tag{4.2}$$

The referenced variables are defined as,

- $\mathbf{x}$ is the system state
- $\mathbf{u}$ is the control actions
- h is the time step
- δ_f are the bounds on the desired final
- δ_i are the bounds on the initial state
- N is the number of knot points.
- d is the minimum distance from state $\mathbf{x}$ to an obstacle.
- r is the minimum allowable distance to obstacles
- $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ are the state bounds
- $\mathbf{u}_{min}$ and $\mathbf{u}_{max}$ are the control bounds

Note that the feasibility problem is formulated as a constrained optimization problem with no cost function. As a result, the cost function is set to zero. The distance function, $d(x)$ is generated using a known OctoMap (Hornung et al., 2013). The Sparse Nonlinear Optimizer (SNOPT) was used to solve this optimization problem (Gill, Murray, and Saunders, 2005).

4.4 Local Linear Feedback Control

A time-varying linear quadratic regulator (TVLQR) is used to track the trajectory generated by direct transcription. The control is calculated as

$$\mathbf{u}(t, \mathbf{x}) = \mathbf{K}(\mathbf{x} - \mathbf{x}_0(t)) + \mathbf{u}_0(t). \tag{4.3}$$

where $\mathbf{K}$ is calculated by integrating

$$-\dot{\mathbf{S}}(t) = \mathbf{A}(t)^T\mathbf{S}(t) + \mathbf{S}(t)\mathbf{A}(t) - \mathbf{S}(t)\mathbf{B}(t)\mathbf{R}^{-1}\mathbf{B}(t)^T\mathbf{S}(t) + \mathbf{Q} \tag{4.4}$$

$$\mathbf{A}(t) = \frac{\partial \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0(t))}{\partial \mathbf{x}}$$

$$\mathbf{B}(t) = \frac{\partial \mathbf{f}(\mathbf{x}_0(t), \mathbf{u}_0(t))}{\partial \mathbf{u}}$$

backwards in time from $t = T$ to $t = 0$.

The referenced variables are defined as,

- $\mathbf{x}$ is the system state
- $\mathbf{u}$ is the control actions
- $\mathbf{x}_0(t)$ is the state trajectory generated by direct transcription
- $\mathbf{u}_0(t)$ is the control trajectory generated by direct transcription

- **f** is the system dynamics
- **Q** is the cost on the state
- **R** is the cost on the control
- $\mathbf{Q}_f$ is the final cost, which defines $S(T)$

Chapter 5

State Estimation and Perception Aware Trajectory Optimization

5.1 Perception Aware Trajectory Optimization

Rapid attitude changes can make it difficult to reliably observe features in the environment, which may negatively impact state estimation. Perception aware trajectory optimization aims to generate trajectories that allow the system to better observe features and thus improve state estimation. This can achieved by integrating a visibility term into the trajectory generation cost function.

A visibility cost term for limited field of view cameras is presented in (Frey, Steiner, and How, 2019). It aims to bring features into the center of the field of view of the camera. The proposed visibility cost term is

$$\sigma(x,l) = \begin{cases} \frac{1}{2}(cos(a\theta)+1) & |\theta| < \theta_{max} \\ 0 & else \end{cases} \tag{5.1}$$

$$\theta = cos^{-1}(\frac{{}^{c}\mathbf{r}^{T}\hat{\mathbf{e}}}{||{}^{c}\mathbf{r}||_2}) \in [0,\pi]$$

where x is the camera state, l is a features, $\hat{\mathbf{e}}$ is the optical axis, ${}^{c}\mathbf{r}$ is the feature location in the camera frame, θ_{max} is the maximum view angle, and $a = \pi/\theta_{max}$ is a scaling term.

We can naturally extend the NMPC feasibility problem to an optimization problem where we use the cost term to affect behavior while still ensuring dynamic constraint satisfaction. Integrating this term into the direct transcription problem, we have

$$
\begin{aligned}
\min_{\mathbf{x}_k, \mathbf{u}_k, h} \quad & \sum_{k=1}^{N} \sum_{i=1}^{M} -\sigma(x_k, l_i) \\
\text{s.t.} \quad & \forall k \in \{0, \dots N\} \text{ and} \\
& \mathbf{x}_k - \mathbf{x}_{k+1} + \frac{h}{6.0}(\dot{\mathbf{x}}_k + 4\dot{\mathbf{x}}_{c,k} + \dot{\mathbf{x}}_{k+1}) = 0 \\
& \mathbf{x}_f - \boldsymbol{\delta}_f \leq \mathbf{x}_N \leq \mathbf{x}_f + \boldsymbol{\delta}_f \\
& \mathbf{x}_i - \boldsymbol{\delta}_i \leq \mathbf{x}_0 \leq \mathbf{x}_i + \boldsymbol{\delta}_i \\
& \mathbf{x}_{min} \leq \mathbf{x}_k \leq \mathbf{x}_{max}, \;\; \mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} \\
& d(\mathbf{x}) \geq r \\
& h_{min} \leq h \leq h_{max}
\end{aligned}
\tag{5.2}
$$

where

$$\dot{\mathbf{x}}_k = \mathbf{f}(t, \mathbf{x}_k, \mathbf{u}_k), \;\; \dot{\mathbf{x}}_{k+1} = \mathbf{f}(t, \mathbf{x}_{k+1}, \mathbf{u}_{k+1})$$

$$\mathbf{u}_{c,k} = (\mathbf{u}_k + \mathbf{u}_{k+1})/2$$

$$\mathbf{x}_{c,k} = (\mathbf{x}_k + \mathbf{x}_{k+1})/2 + h(\dot{\mathbf{x}}_k - \dot{\mathbf{x}}_{k+1})/8$$

$$\dot{\mathbf{x}}_{c,k} = \mathbf{f}(t, \mathbf{x}_{c,k}, \mathbf{u}_{c,k}).$$

l_i, ..., l_M are all features observed within the last 20 time steps (0.2 seconds).

5.2 State Estimation

State estimation is an important aspect for any robotic system. While ground truth state is accessible within a motion capture system, it is not accessible in many situations. If a fixed-wing UAV is to be deployed in the real world, it must have access to accurate state estimation. State estimation can be difficult for fixed-wing UAVs. Since the system moves in three dimensional space at high velocities with many degrees of freedom, state estimation must be fast and accurate in order to perform trajectory generation, obstacle avoidance, and control.

5.2.1 S-UKF-LG

For this essay, an Unscented Kalman Filter implementation of the Stereo Multi-State Constraint Kalman Filter, proposed in (Brossard, Bonnabel, and Barrau, 2018), is utilized for state estimation. Sun et al. proposed the stereo

multistate constraint Kalman filter (S-MSCKF) to perform stereo visual inertial odometry on UAVs (Sun et al., 2018). This estimator had a computational cost similar to state of the art monocular methods. This work was built heavily upon a prior work for monocular visual inertial odometry which formulated a multistate constraint Kalman filter (MSCKF) (Mourikis and Roumeliotis, 2007a). Brossard et. al proposed S-UKF-LG, which built upon the S-MSCKF by utilizing an Unscented Kalman Filter (UKF) (Brossard, Bonnabel, and Barrau, 2018). Furthermore, the UKF was formulated specifically for use on Lie Groups, as formulated in their prior paper (Brossard, Bonnabel, and Condomines, 2017a). S-UKF-LG removed the need to compute Jacobians and takes into account the Lie group structure of the system's state.

This method has good accuracy when compared to other visual inertial odometry methods and can be performed efficiently. Brossard reported that S-UKF-LG performed well in terms of root mean square error (RMSE) on the EuRoC dataset (Burri et al., 2015) compared to S-MSCKF. While OKVIS (Leutenegger et al., 2015) and VINS-MONO (Qin, Li, and Shen, 2018) outperformed S-UKF-LG, they did so at the cost of significantly more CPU Load.

Here follows a brief overview of the S-UKF-LG algorithm.

5.2.1.1 Dynamical and Measurement Model

The dynamical model of the IMU states is represented as

$$\begin{aligned}
\frac{\partial \mathbf{R}}{\partial t} &= \mathbf{R}(\boldsymbol{\omega} - \mathbf{b}_\omega + \mathbf{n}_\omega)_\times \\
\frac{\partial \mathbf{v}}{\partial t} &= \mathbf{R}(a - \mathbf{b}_a + \mathbf{n}_a) + \mathbf{g} \\
\frac{\partial \mathbf{x}}{\partial t} &= \mathbf{v} \\
\frac{\partial \mathbf{b}_\omega}{\partial t} &= \mathbf{n}_{\mathbf{b}_\omega} \\
\frac{\partial \mathbf{b}_a}{\partial t} &= \mathbf{n}_{\mathbf{b}_a}
\end{aligned} \tag{5.3}$$

where $\mathbf{R}$ is the orientation of the body frame, $\mathbf{v}$ is the velocity, $\mathbf{x}$ is the position, and $\mathbf{b}_\omega$ and $\mathbf{b}_a$ are the IMU biases

The dynamical model of the camera states is represented as

$$\begin{aligned}
\frac{\partial \mathbf{T}^{IC}}{\partial t} &= \mathbf{0} \\
\frac{\partial \mathbf{T}_i^C}{\partial t} &= \mathbf{0},\ i\ =\ 1, ..., N
\end{aligned} \tag{5.4}$$

where $\mathbf{T^{IC}}$ is the transformation between the IMU and left camera frame and $\mathbf{T_i^C}$ are previously recorded left camera poses.

The noise of the IMU is represented as

$$\mathbf{n} = [\mathbf{n}_\omega^T\ \mathbf{n}_a^T\ \mathbf{n}_{\mathbf{b}_\omega}^T\ \mathbf{n}_{\mathbf{b}_a}^T]^T \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{5.5}$$

Where $\mathbf{n}_\omega$, $\mathbf{n}_a$, $\mathbf{n}_{\mathbf{b}_\omega}$, $\mathbf{n}_{\mathbf{b}_a}$ are the various white Gaussians noises and $\mathbf{Q}$ is

the noise covariance matrix. The state and error are embedded on Lie Groups. Please refer to (Brossard, Bonnabel, and Barrau, 2018) for the necessary mathematical background. After this embedding, the IMU error is represented as

$$\xi = [\xi_{\mathbf{R}}^T \; \xi_{\mathbf{v}}^T \; \xi_{\mathbf{x}}^T \; \xi_{\omega}^T \; \xi_{a}^T \; \xi_{IC}^T \; \xi_{C_1}^T \; ... \; \xi_{C_N}^T]^T \tag{5.6}$$

5.2.1.2 Propagation Step

The IMU state is propagated using a 4-th order Runge-Kutta numerical integration. To propogate the IMU uncertainty, first the rate of change of the IMU uncertainty is linearized.

$$\dot{\xi} = \mathbf{F}\xi + \mathbf{G}\mathbf{n} \tag{5.7}$$

where

$$F = \begin{bmatrix} 0 & 0 & 0 & 0 & -R \\ (g)_\times & 0 & 0 & -\mathbf{R} & -(v)_\times \mathbf{R} \\ 0 & \mathbf{I} & 0 & 0 & -(x)_\times \mathbf{R} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$G = \begin{bmatrix} 0 & \mathbf{R} & 0 & 0 \\ \mathbf{R} & (\mathbf{v})_\times \mathbf{R} & 0 & 0 \\ 0 & (\mathbf{x})_\times \mathbf{R} & 0 & 0 \\ 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{I} \end{bmatrix}$$

Then, the discrete state transition matrix and noise covariance matrix are calculated.

$$\Phi_n = \Phi(t_{n+1}, t_n) = exp_m(\int_{t_n}^{t_{n+1}} \mathbf{F}(\tau) d\tau) \tag{5.8}$$

$$\mathbf{Q}_n = \int_{t_n}^{t_{n+1}} \Phi(t_{n+1}, \tau) \mathbf{G}\mathbf{Q}\mathbf{G}^T \Phi(t_{n+1}, \tau)^T d\tau \tag{5.9}$$

The covariance matrix is propagated as follows

$$\mathbf{P}_{n+1} = \begin{bmatrix} \mathbf{P}_{n+1}^{II} & \Phi_n \mathbf{P}_n^{IC} \\ \mathbf{P}_N^{(}CI)\Phi_n^T & (P)_n^{CC} \end{bmatrix} \tag{5.10}$$

$$\mathbf{P}_{n+1}^{II} = \Phi_n \mathbf{P}_n^{II} \Phi_n + \mathbf{Q}_n$$

When new images are received, the camera state and covariance matrix are augmented.

$$\mathbf{P}_{n+1} = \begin{bmatrix} \mathbf{I} \\ \mathbf{J} \end{bmatrix} \mathbf{P}_{n+1} \begin{bmatrix} \mathbf{I} \\ \mathbf{J} \end{bmatrix}^T \tag{5.11}$$

$$\mathbf{J} = \begin{bmatrix} \mathbf{I} & 0 & 0 & \mathbf{0}_{3\times 6} & \mathbf{R} & 0 & 0 & \mathbf{0}_{3\times 6N} \\ 0 & 0 & \mathbf{I} & (0)_{3\times 6} & (\mathbf{x})_\times \mathbf{R} & 0 & \mathbf{R} & \mathbf{0}_{3\times 6N} \end{bmatrix}$$

5.2.1.3 Update Step

The state and covariance are updated based on measurements as follows. For each feature position and estimated feature position (Mourikis and Roumeliotis, 2007b), the residual is calculated as

$$\mathbf{r}_i^j = \mathbf{y}_i^j - {}^{UKF}\mathbf{y}_i^j = {}^{UKF}\mathbf{H}_i^j \xi + \mathbf{H}_{p^j}^i \xi_{\mathbf{p}^j} + \mathbf{n}_i^j \tag{5.12}$$

$$\mathbf{H}_{\mathbf{p}^j}^i = \mathbf{J}_i^j \begin{bmatrix} (\overline{\mathbf{R}}_i^L)^T \\ (\overline{\mathbf{R}}_i^R)^T \end{bmatrix}$$

where $(\overline{\mathbf{R}}_i^L)^T$ and $(\overline{\mathbf{R}}_i^R)^T$ are the orientations of the i-th left and right cameras and

$$J_i^j = \begin{bmatrix} 1/\overline{z}_l & 0 & -\overline{x}_l/\overline{z}_l^2 \\ 0 & 1/\overline{z}_l & -\overline{y}_l/\overline{z}_l^2 \\ 1/\overline{z}_r & 0 & -\overline{x}_r/\overline{z}_r^2 \\ 0 & 1/\overline{z}_r & -\overline{y}_r/\overline{z}_r^2 \end{bmatrix}$$

${}^{UKF}\mathbf{y}_i^j$ and ${}^{UKF}\mathbf{H}_i^j$ are numerically inferred using UKF-LG (Brossard, Bonnabel, and Condomines, 2017b).

These equations are then stacked to form the full residual, $\mathbf{r}^j$. This is projected into the null space of $\mathbf{H}^i_{\mathbf{p}^j}$, which results in

$$\mathbf{r}^j_o = \mathbf{H}^j_o \xi + \mathbf{n}^j_o \tag{5.13}$$

These equations are stacked to form $\mathbf{r}_o$ and $\mathbf{H}_o$. The updated state is

$$\overline{\chi}^{\times} = exp(\bar{\xi})\overline{\chi} \tag{5.14}$$

$$\overline{\xi} = \mathbf{K}\mathbf{r}_o$$

$$\mathbf{K} = \mathbf{P}_n \mathbf{H}_o^T / \mathbf{S}$$

$$\mathbf{S} = \mathbf{R} + \mathbf{H}_o \mathbf{P}_n \mathbf{H}_o^T$$

The updated covariance matrix is

$$\mathbf{P}_n^{\times} = \mathbf{P}(\mathbf{I} - \mathbf{K}\mathbf{H}_o) \tag{5.15}$$

5.2.2 UKF S_MSCKF Validation

The system is simulated while performing both S-UKF-LG and a naive IMU state integration to evaluate if the S-UKF-LG algorithm is suitable for fixed-wing UAV state estimation. The system is controlled using the true state and has access to the true map.

The simulation map represents a hallway with corners and branches. The fixed wing is initialized in the middle of the hallway, and the goal point is around two corners. These corners are challenging to plan through due to the restricted field of view. The dynamics are run at 100 Hz with euler

integration. Ten knot points are used for direct transcription, with Hermite-Simpson Integration as the integration method between knot points. SNOPT is used for numerical optimization and is capped at 5 major iterations, which is similar to what has been observed on hardware.

For the constraints on direct transcription, strong bounds on final knot points were enforced:

$$\delta_f = [0.1, 0.1, 0.1, 0.1, 1, 0.1, 100, 100, 100, 100, 3, 3, 0.5, 0.5, 0.5, 0.5]$$

These strong bounds correspond to the maximum distance the final knot point is allowed to vary from the desired final state, as formulated in Equation 4.1. The diagonals of the cost matrices $\mathbf{Q}$ and $\mathbf{R}$ for TVLQR were set to,

$$diag(\mathbf{Q}) = [10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10]$$

$$diag(\mathbf{R}) = [1, 1, 1, 1]$$

with final costs:

$$diag(\mathbf{Q_f}) = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$

The stereo camera used in this essay is simulated in Gazebo. Gazebo is a powerful platform, capable of simulating dynamics and graphics. It is also able to simulate sensors through plugins, and has tight Robot Operating System (ROS) integration. This has made it a powerful tool for designing robotics systems and algorithms.

A mesh, which mirrors the occupancy map used in the fixed wing simulation, was created and loaded into Gazebo. This mesh is textured with a brick

pattern, shown in Figure 5.1. A stereo camera Gazebo plugin uses this mesh and an assigned model pose to generate stereo images (Figure 5.2). These images are published at a rate of 30 Hz.

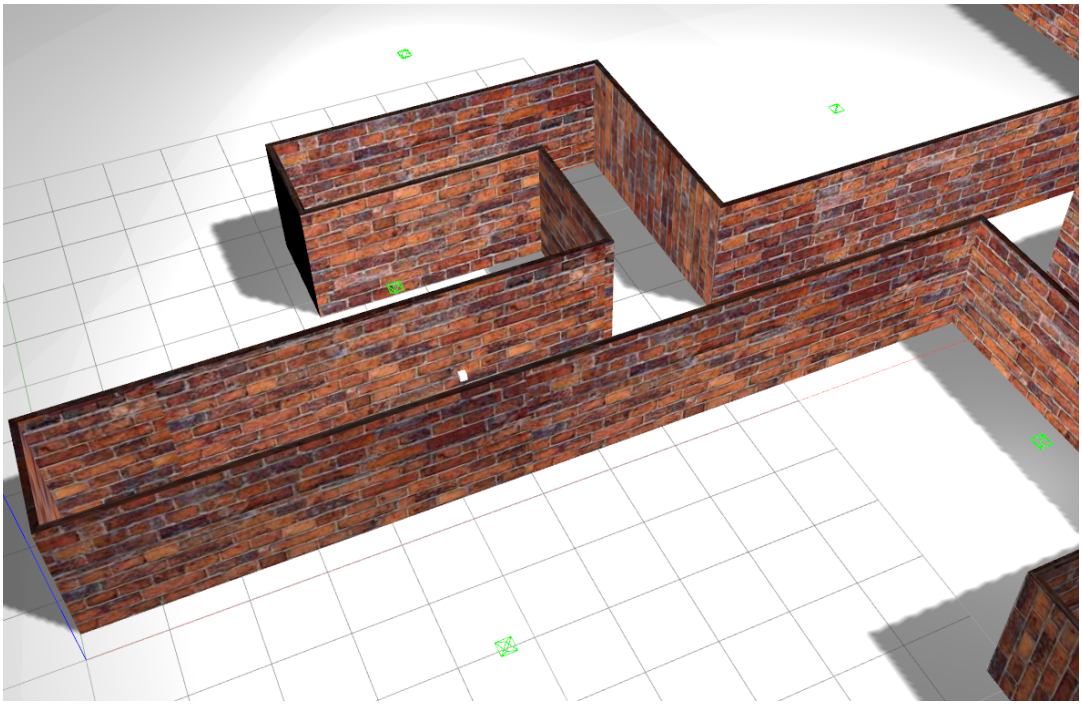

Figure 5.1: Gazebo Map

The camera images are 320x240 pixels and have no distortion. The horizontal field of view is 1.5 radians, which is similar to that of commercially available stereo/depth cameras (add source). The IMU has Gaussian noise added to acceleration ($\frac{m}{s^2}$) and angular velocity ($\frac{rad}{s}$) measurements, $\mathcal{N}(0, \text{diag}(1,1,1))$ and $\mathcal{N}(0, \text{diag}(0.1, 0.1, 0.1))$ respectively. The acceleration and angular velocity measurements also have added biases, $[0.1, 0.1, 0.1]$ and $[0.01, 0.01, 0.01]$ respectively.

When the simulation is run, the state acquired with IMU integration rapidly diverges, while the state estimate acquired with S-UKF-LG stays

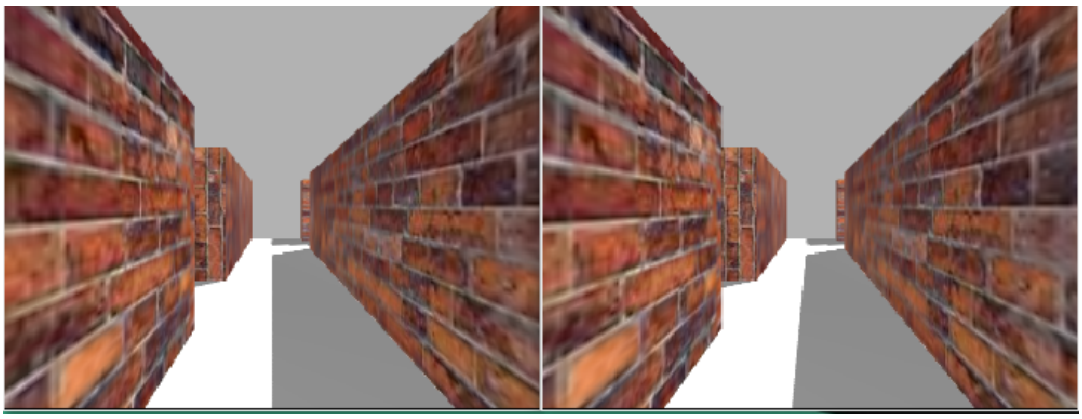

Figure 5.2: Stereo Image from Gazebo

within an error bound of the ground truth state. Figures 5.3, 5.4, and 5.5 display visualizations of the ground truth state, the estimated state, and the IMU integrated state in an example simulation run. In Figure 5.3, the blue fixed-wing is the ground truth state, the green fixed-wing is the IMU integrated state, and the red fixed-wing is the S-UKF-LG state estimation. The white trajectory is the smoothed RRT, the yellow trajectory is direct transcription trajectory currently being tracked, and the green trajectory is direct transcription trajectory being optimized for next control interval.

5.2.3 Simultaneous State Estimation and Planning

The S-UKF-LG state estimation can be used to control the system. The simulation was run ten times, and resulting trajectories are plotted in Figure 5.6. Each simulation is run with a different random number generation seed, resulting in different IMU measurements and RRT sampling. An example simulation run is shown in Figure 5.7. Each simulation run was successful; the fixed-wing was able to navigate the hallway, avoid walls, and reach within 0.5

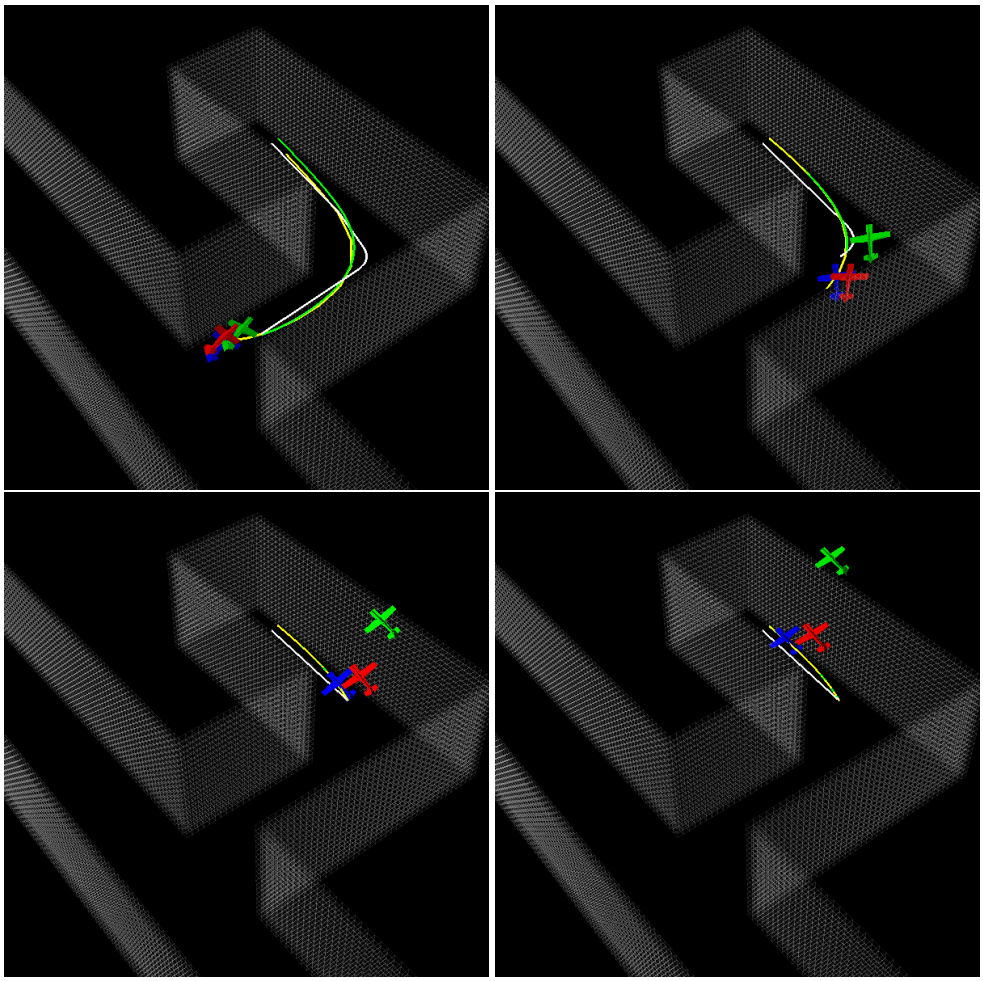

Figure 5.3: S-UKF-LG validation

Blue fixed-wing UAV: ground truth state
Green fixed-wing UAV: IMU integrated state
Red fixed-wing UAV: S-UKF-LG state estimation.
White trajectory: smoothed RRT
Yellow trajectory: direct transcription trajectory currently being tracked
Green trajectory: direct transcription trajectory being optimized

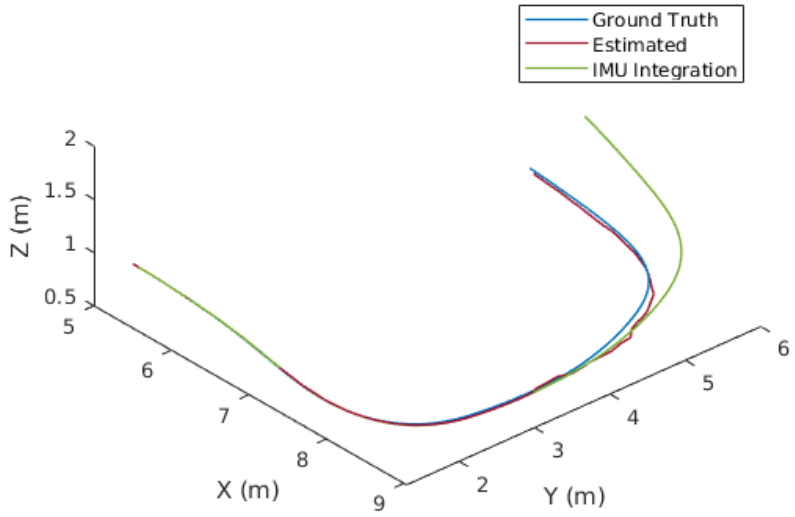

Figure 5.4: Visualization of trajectories in Figure 5.3

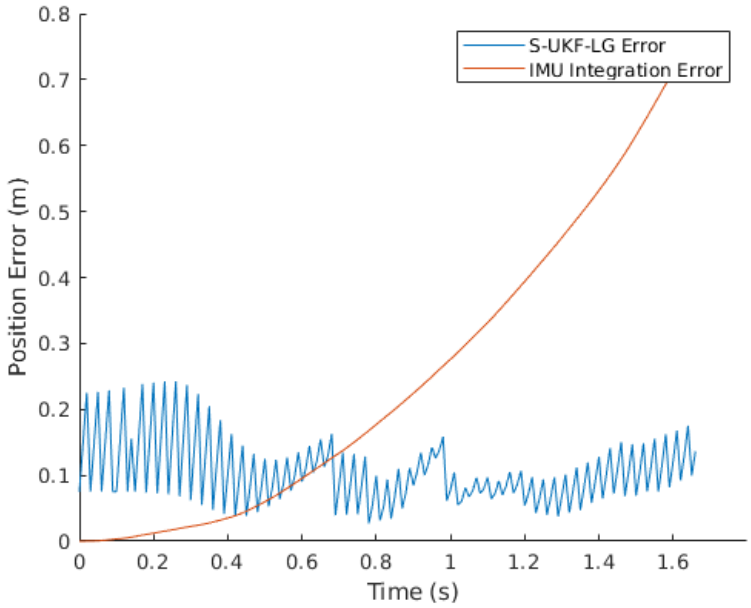

Figure 5.5: Errors associated with trajectories in Figure 5.3

meters of the goal position.

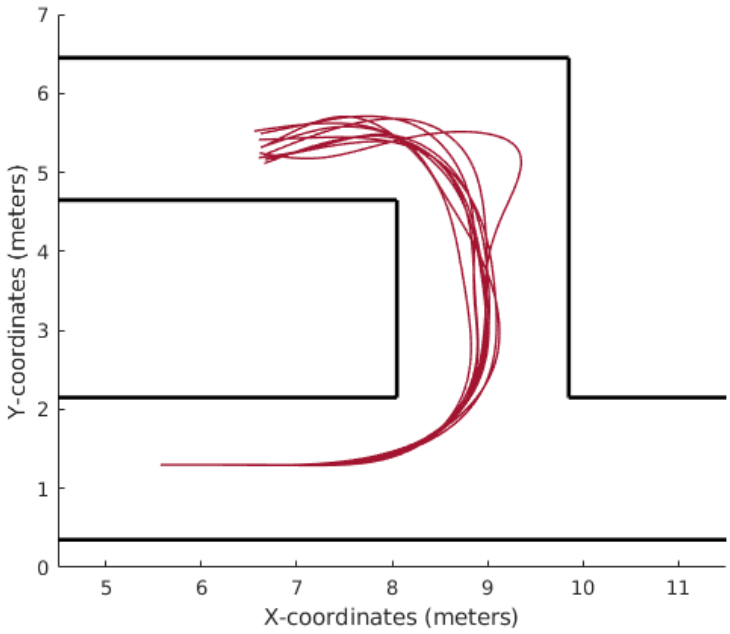

Figure 5.6: Trajectories while controlling fixed-wing with S-UKF-LG state estimation

5.3 Simulation and Results

Simulation was performed with and without the visibility cost. Across ten runs, the positional error and the trace of the positional covariance of the state estimation were collected. In the case of a high-speed fixed-wing UAV, features are continuously updated and are coming in and out of the field of view. Therefore, the cost function may fight against the constraints imposed upon trajectory generation by penalizing features leaving the field of view and rewarding features entering the field of view. The objective of the resulting constrained optimization problem is to find a balance in which the constraints are met while feature observability is maximized.

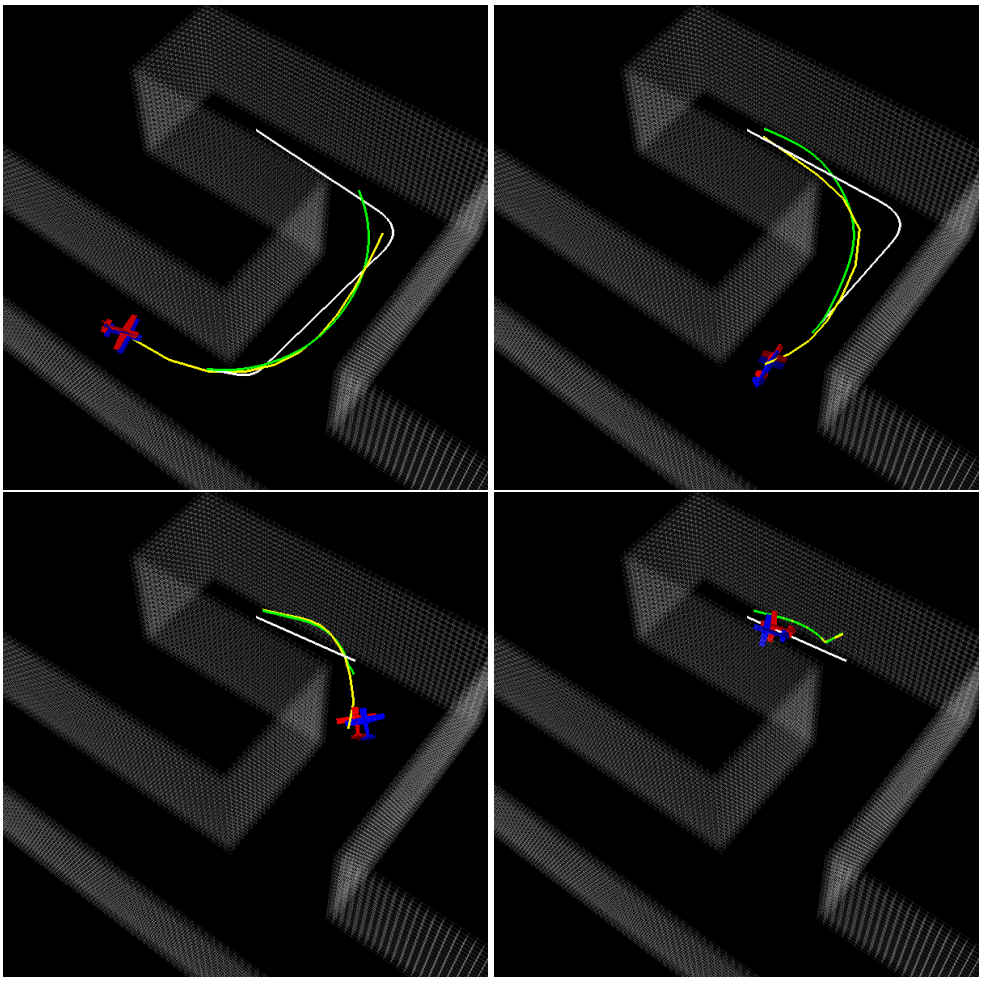

Figure 5.7: Control using state estimation from S-UKF-LG

Blue fixed-wing UAV: ground truth state
Green fixed-wing UAV: IMU integrated state
Red fixed-wing UAV: S-UKF-LG state estimation.
White trajectory: smoothed RRT
Yellow trajectory: direct transcription trajectory currently being tracked
Green trajectory: direct transcription trajectory being optimized

Unfortunately, no significant improvements in state estimation were observed while using the visibility cost. Although the error and covariance trace while using the visibility cost tended to be smaller at the final state (Figures 5.8 and 5.9), it was not significantly so. The mean error along the entire trajectory was almost identical, and the mean covariance trace tended to be slightly larger (Figures 5.10 and 5.11).

These results indicate that this cost term is not a viable method for improving state estimation for the given system and environment. It may be that the optimizer (SNOPT) is unable to significantly minimize the cost function within the major iteration constrain imposed. The cost functions ineffectively could also be caused by the fact that the fixed-wing is navigating a feature rich environment. Thus, the system may be able to view enough features no matter which direction the camera is pointed. Future experiments can be run to investigate if the visibility cost could generate trajectories that improve state estimation in feature deficient environments, such as hallways with relatively blank walls.

5.4 Simplified Test Case

In order to further analyze the visibility cost function, a simulation of a greatly simplified test case was run. In this test case, the goal state is straight down the hallway instead of around corners. A single group of features is placed to the left of the hallway. Additionally, the camera is simulated at 100Hz to maximize the amount of camera data received.

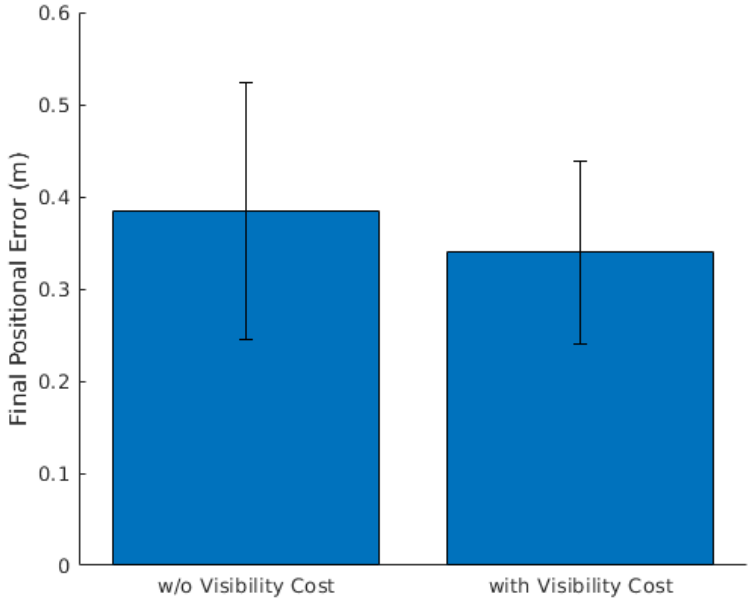

Figure 5.8: Positional error at final state with and without visibility cost

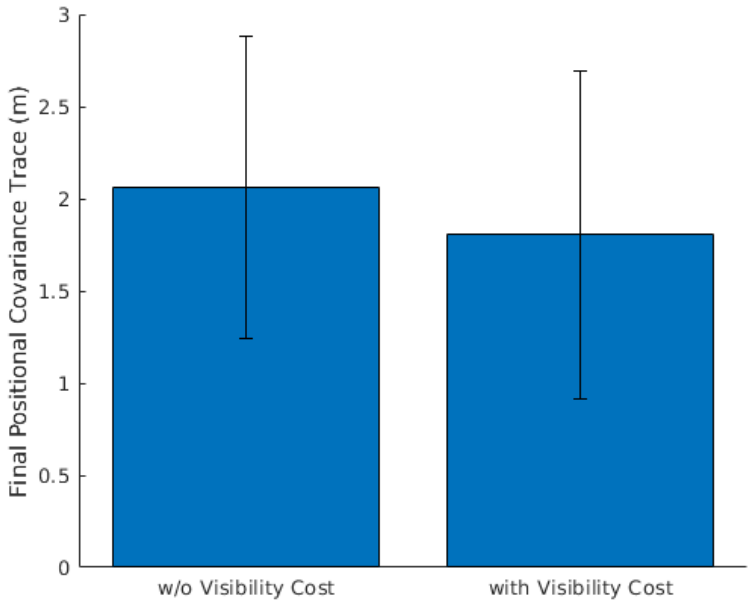

Figure 5.9: Positional covariance trace at final state with and without visibility cost

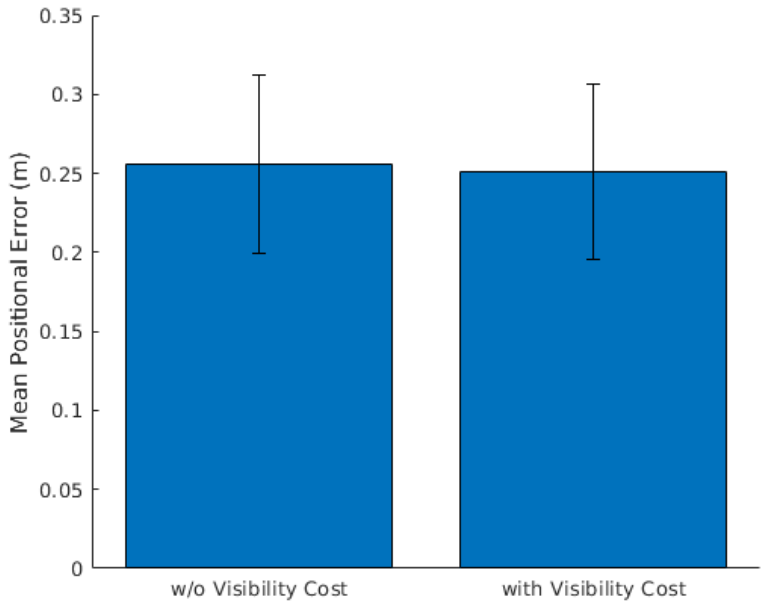

Figure 5.10: Mean positional error along trajectory with and without visibility cost

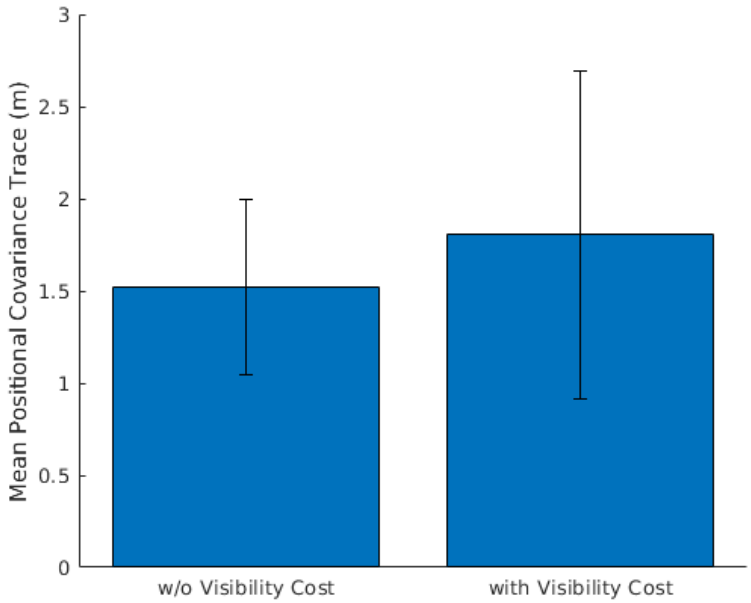

Figure 5.11: Mean positional covariance trace along trajectory with and without visibility cost

This test case was run ten times with and without the visibility cost function. The use of the cost function improves feature visibility along the trajectory and slightly reduces positional covariance trace. However, the positional error of state estimation is not improved: the mean final error without the visibility cost was 0.2503 m and 0.2776 m with the visibility cost. The results of this test case are shown in Figures 5.12, 5.13, and 5.14.

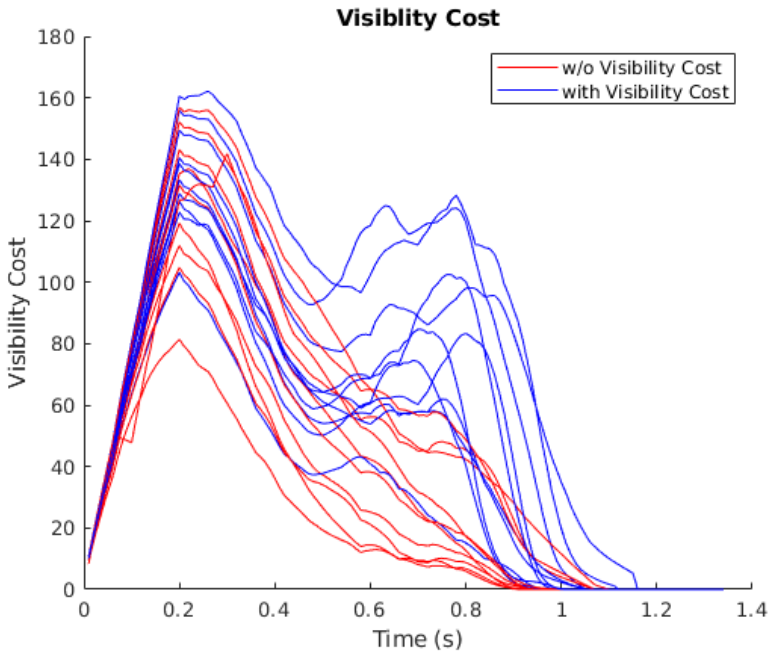

Figure 5.12: Visibility along trajectories with and without visibility cost

Note that there is a second spike in visibility while using the visibility cost. This occurs because the fixed wing slows and briefly rotates such that the camera points towards the features. Also note the covariance appears to diverge towards the end of the simulation. This is due to the fact that there are no features at the end of the hallway. However, there is a minor decrease in covariance before features move out of the field of view.

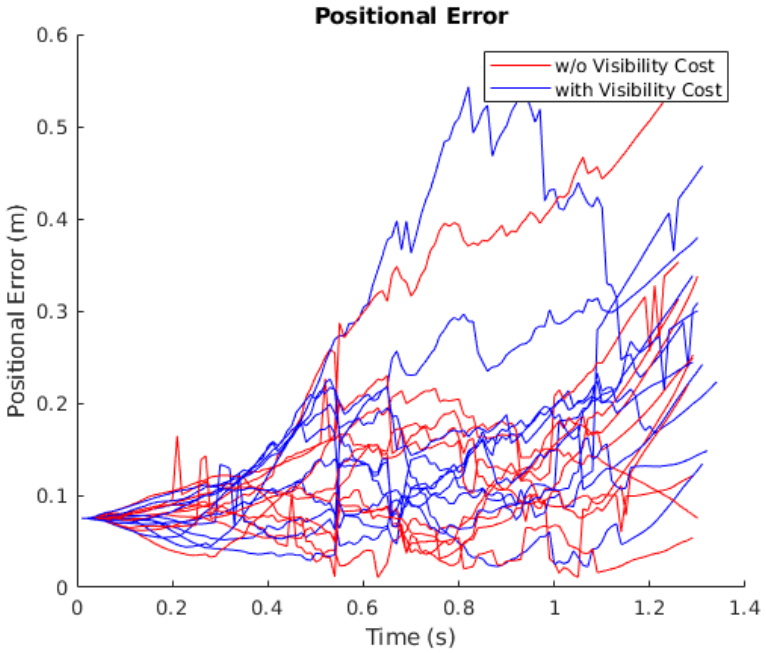

Figure 5.13: Positional error along trajectories with and without visibility cost

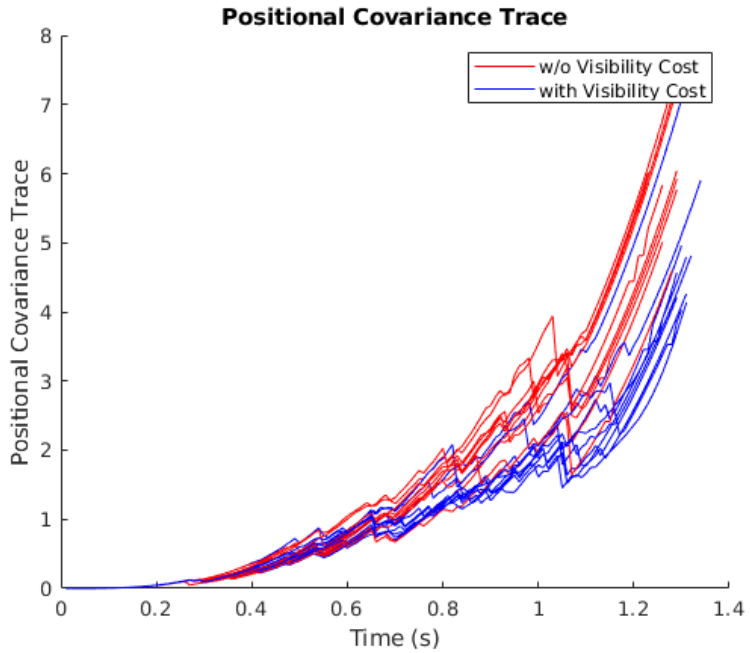

Figure 5.14: Positional covariance Trace along trajectories with and without visibility cost

This test case verifies that the cost function is able to improve feature visibility despite the constraints placed on the trajectory generation problem. The minor covariance improvement and lack of error improvement suggests that this cost function may not be an effective method for improving state estimation for this system.

Chapter 6

Simultaneous Mapping and Navigation

6.1 Issues Posed

While the framework outlined in Chapter 4 is quite effective with a known map, there are challenges to using this framework in an unknown environment. If the system plans a trajectory that passes through an unseen obstacle, this trajectory will not be feasible. Since this trajectory is used as a seed for the next one, this can pose issues even once the obstacle is sensed. Given that the optimizer is time constrained, it may not be able to generate a feasible trajectory in time if seeded with this infeasible one. This situation can occur when the system is turning a corner in a hallway map. The sensor is unable to see the wall or obstacles around the corner due to a limited field of vision.

The main way to address this challenge is to generate trajectories that have a very low chance of being generated in obstacles in the unknown region of the map. This can be done in a couple ways: only plan trajectories in known areas of the map, or plan trajectories based off a prediction of where obstacles

will be in the unknown map.

6.2 Mapping

We perform mapping using an OctoMap (Hornung et al., 2013), which utilizes an Octree and probability thresholds to maintain occupancy maps. The system is navigating a "2.5D" environment where the walls are uniform along the z dimension. Due to this uniformity, the OctoMap is only built along the x and y dimensions. While generating the RRT and perform direct transcription, obstacle constraints are imposed according on the x and y coordinates. Therefore, 3D trajectories are generated using the 2D map. While building this map with LIDAR measurements is fairly straightforward, building it with stereo camera images requires more processing.

Here follows the mapping proccess using the stereo camera sensor. First, 3D reconstruction is performed from the two stereo images. For this essay, stereo_image_proc was utilized (source). This ROS node undistorts and colorizes the images. It then calculates a disparity map using OpenCV's block matching algorithm. Using the camera's intrinsic properties, a 3D point cloud of the viewed scene can be generated (Figure 6.1). This point cloud is the filtered using the Point Cloud Library (PCL).

First, the Point Cloud is transformed from the camera frame to the world frame. Then, it is down-sampled using a Voxel Grid with a leaf size of (0.3 m, 0.3 m, 0.3 m) (Figure 6.2). This down-sampling greatly reduced the number of points, which is important for efficiently adding it to the OctoMap. The ground points are removed, and a slice of the point cloud around the camera's height is

extracted (Figure 6.3). A slice size of 0.4 m was used. The remaining points are projected onto the $z = 0$ plane (Figure 6.4). The result of these transformations is a measurement which resembles the LIDAR sensor's measurement; a point cloud measurment contained to the horizontal plane.

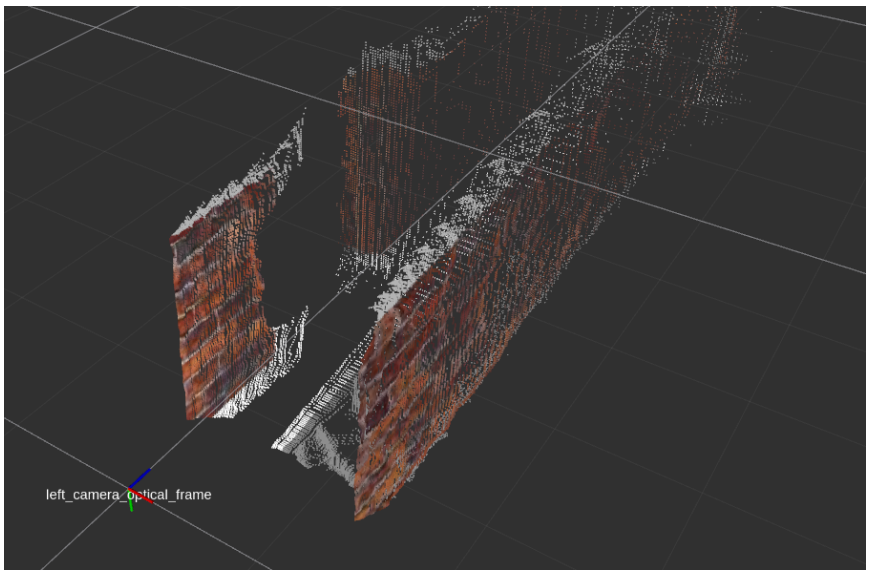

Figure 6.1: 3D Reconstruction

6.3 Frontier Search

The occupancy map can be defined by three regions: occupied, unoccupied, and unknown. Regions which have been sensed to contain obstacles are occupied, while open space is defined as unoccupied. Regions of the map that have not been observed are unknown. Frontiers define edges between unoccupied and unknown regions of the map.

While unknown space can be treated as unoccupied for trajectory generation, it is a safer option to avoid planning trajectories in unknown space, if

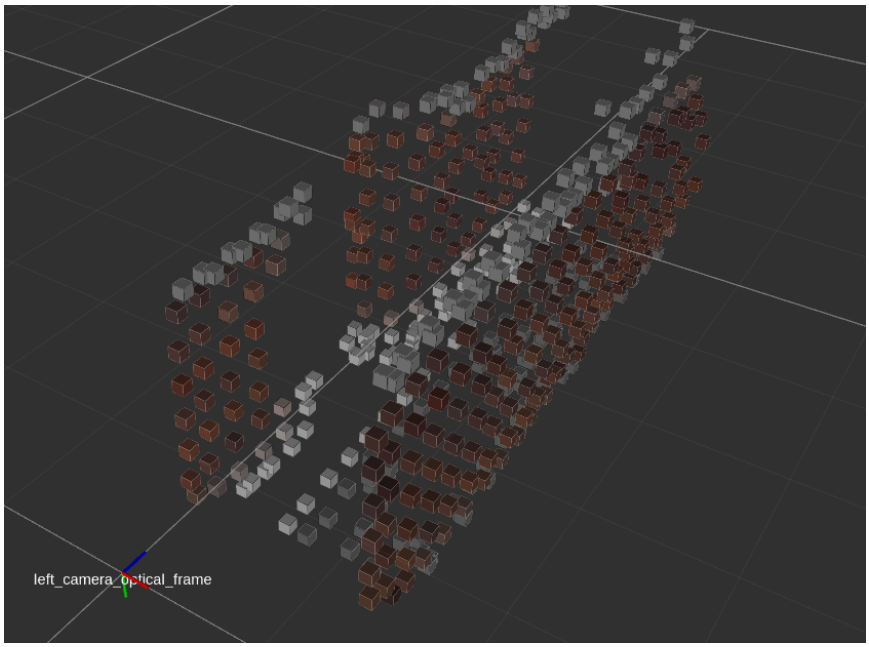

Figure 6.2: Down-sampled Point Cloud

Figure 6.3: Sliced Point Cloud

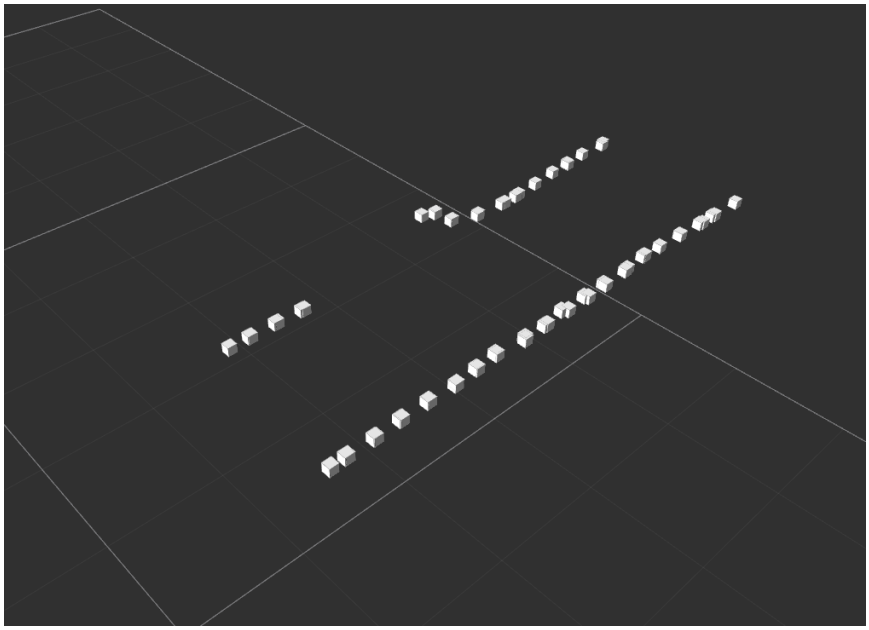

Figure 6.4: Projected Point Cloud

possible. In order to do so, frontiers between known and unknown regions of the map are searched for. These frontiers are generated using a flood fill algorithm. The algorithm takes in an occupancy grid where each cell is either occupied, unoccupied or unknown. The cell closest to the system's current state is set as the initial pixel. Each unvisited neighboring pixel is visited. If the neighboring pixel is occupied, then it's neighbors are not marked to be visited. If it is unoccupied, then it's neighbors are marked to be visited. If it is unknown, then it is marked as a frontier, and it's neighbors are not marked to be visited.

Once flood fill is finished, frontier cells are grouped together by adjacency into frontiers. For every frontier cell, each existing frontier is checked for adjacency. If there is one adjacent frontier, the frontier cell is added to it. If there are no eligible frontiers, a new frontier with this cell is formed. If

there are two eligible frontier, they are merged together, and the frontier cell is added to the merged frontier. Any frontiers that are too small (below a threshold of number of cells) are removed. These frontiers are likely gaps in the system's sensing, rather than true frontiers. The algorithm is outlined in Algorithm 2 and an example of the result is shown in Figure 6.5. The translucent white cells are the map, the blue cells are the observed map, and the solid white cells belong to a frontier.

6.4 Horizon Point Selection

The NMPC framework reviewed in Chapter 4 is adapted for use while simultaneously mapping by modifying the horizon point selection. The horizon point, which is selected at a time horizon along the RRT, may be in unknown regions of the map. This is can be due to both the limited field of view of the sensor and the constrained space. Since trajectories are optimized using previous trajectories as seeds, a trajectory becoming infeasible due to new obstacles being observed could be detrimental to the optimization process. To reduce risk of generating infeasible trajectories, this horizon point can be adjusted by using the detected frontiers.

6.4.1 Frontier Midpoint

At the beginning of a control interval, frontiers are searched for. After the RRT path to the goal is generated and smoothed, it is searched to see if it passes through any frontiers. If it does, that frontier is used to select the horizon point. The location of the horizon point is shifted to the mean

Algorithm 2 FIND_FRONTIERS(pos_{init}, map, $thresh$)

```
queue ← QUEUE()
boundary_points ← VECTOR()
boundaries ← VECTOR()
queue.PUSH(pos_init)
while not queue.EMPTY() do
    pos ← queue.FRONT()
    queue.POP()
    if not map.UNKNOWN(pos) then
        if map.UNOCCUPIED(pos) and not map.VISITED(pos) then
            queue.PUSH(map.NEIGHBORS(pos_init))
            map.SET_VISITED(pos)
        end if
    else
        if pos ∉ boundary_points then
            boundary_points.PUSH_BACK(pos)
        end if
    end if
end while
for boundary_point ∈ boundary_points do
    candidates ← ADJACENT_BOUNDARIES(boundary_point, boundaries)
    if candidates.EMPTY() then
        new_boundary ← VECTOR()
        new_boundary.PUSH_BACK(boundary_point)
        boundaries.PUSH_BACK(new_boundary)
    else if candidates.SIZE() = 1 then
        boundaries.AT(candidates.AT(0)).PUSH_BACK(boundary_point)
    else
        boundaries.AT(candidates.AT(0)).PUSH_BACK(boundary_point)
        boundaries ← MERGE_BOUNDARIES(boundaries, candidates)
    end if
end for
for boundary in boundaries do
    if boundary.SIZE() < thresh then
        boundaries.REMOVE(boundary)
    end if
end for
Return boundaries
```

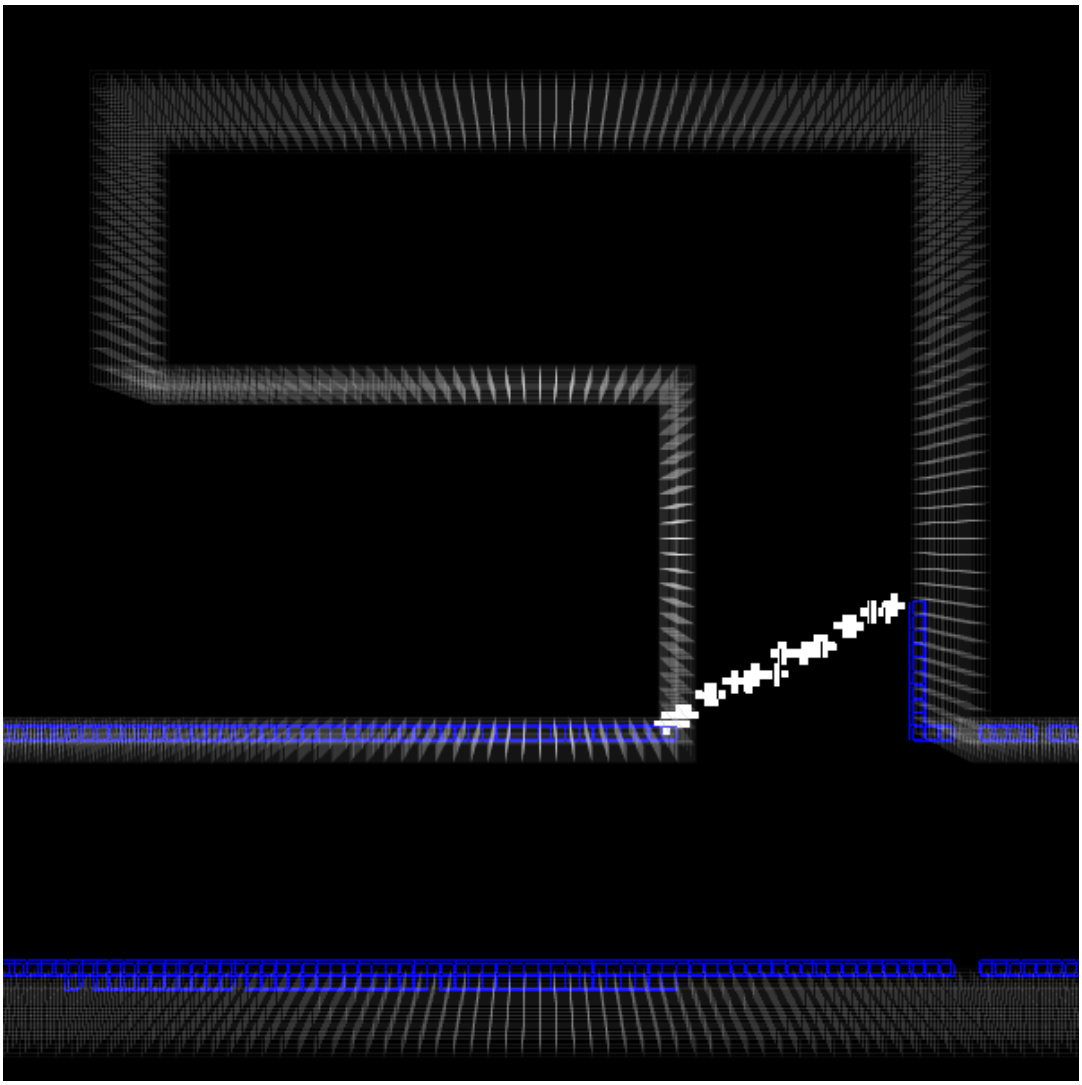

Figure 6.5: An example of a detected frontier

Translucent White Cells: true map
Blue Cells: observed map
Solid White Cells: detected frontier

location value of this frontier (Figure 6.6). If the RRT does not pass through any frontiers, then the trajectory is planned to a time horizon or the goal point, whichever comes first (Algorithm 3).

Algorithm 3 FRONTIER_MIDPOINT(*map*, *boundaries*)

```
RRT ← GENERATE_RRT()
intersect_boundaries ← DETECT_INTERSECTIONS()
if intersect_boundaries.EMPTY() then
    horizon ← TIME_HORIZON(RRT)
else
    horizon ← MIDPOINT(intersect_boundaries.AT(0))
end if
Return horizon
```

Trajectories planned to the frontier midpoints will be completely contained within known regions of the map. This will reduce the risk of planning trajectories through unseen obstacles. Since the frontiers are updated at the beginning of each frontier interval, new frontiers will take into account newly revealed obstacles. However, if obstacles are very close to the frontier, yet hidden from view, the system may have trouble avoiding collision.

6.4.2 Map Prediction

Instead of containing trajectory generation within known map regions, trajectories can also be planned within unknown regions that have a low chance of containing obstacles. While predicting unknown regions of the map is outside of the scope of this essay, doing so is an ongoing field of research (Katyal et al., 2019).

Ideally, at the beginning of each planning interval, the observed map would be passed to a map prediction function, which would return a map

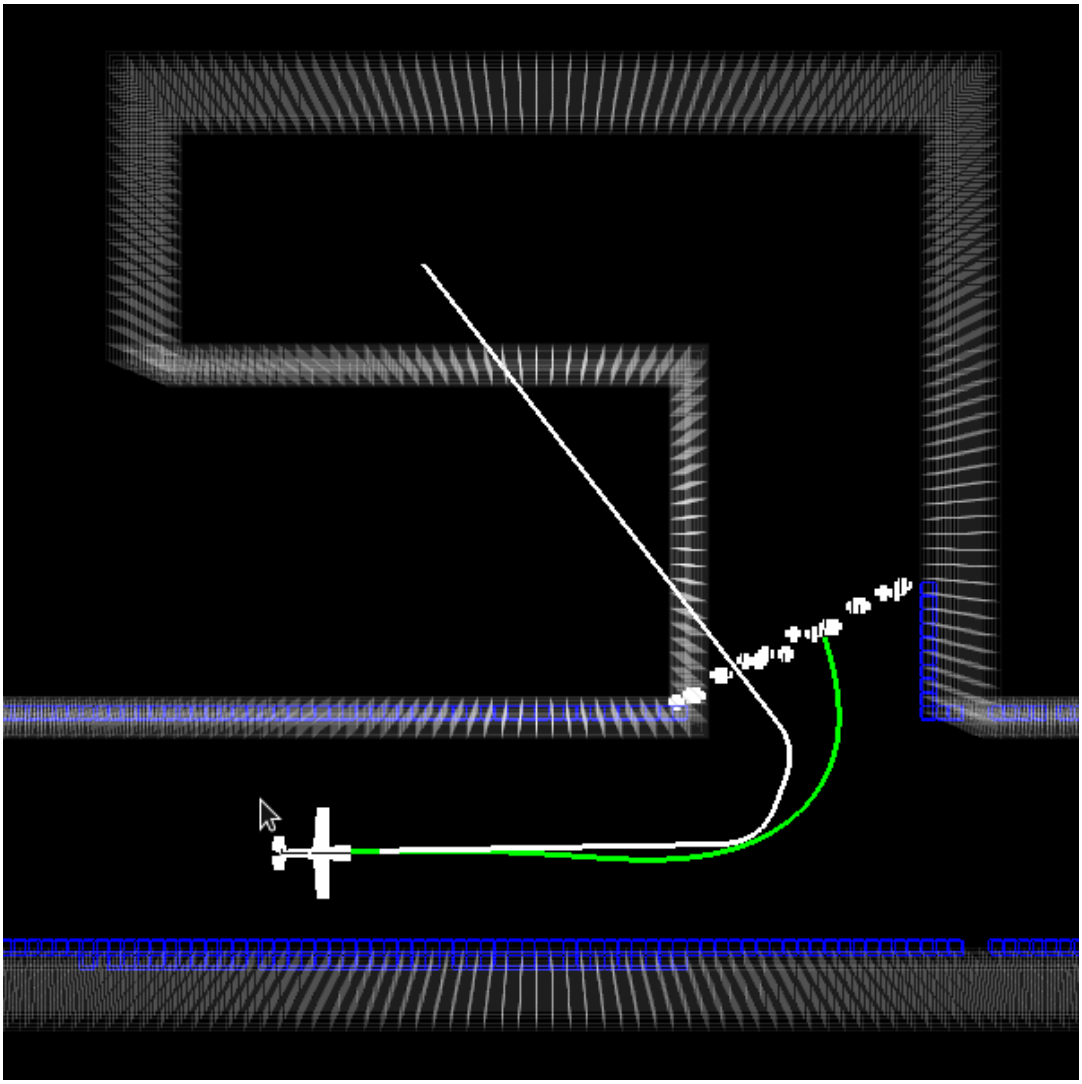

Figure 6.6: Frontier Midpoint Method

White trajectory: smoothed RRT
Green trajectory: direct transcription trajectory being optimized
Translucent White Cells: true map
Blue Cells: observed map
Solid White Cells: detected frontier

with unknown regions filled in with predicted obstacles. Instead, to determine the effectiveness of planning off of future map predictions, we assume that we have access to a perfect prediction of the future map. This prediction expands the observed map by updating it with future observations.

These future observations are obtained by simulating the sensor forward in time along the current trajectory to collect measurements. The RRT is generated in the predicted map, and the time horizon point is selected along this 'predicted RRT'. Figure 6.7 displays an example where the sensor is simulated forward by 0.1 sec. The red cells represent the predicted map. Note that the predicted map enables the trajectory being optimized (green) to better avoid the mostly unobserved wall.

6.4.3 Frontier and Map Prediction

This horizon selection method combines the frontier approach and the map prediction approach. Assuming we have access to a prediction of the future map, we may be able to improve upon frontier midpoint selection. We simulate a perfect map prediction and generate the RRT using the predicted map. However, frontiers are still generated using the current map. The horizon point location is shifted to the intersection of the RRT and the frontier (Figure 6.8). As a result, trajectories are only planned within the known map, but they are planned to a goal on the frontier that might be more likely to put the system on a trajectory within unoccupied space.

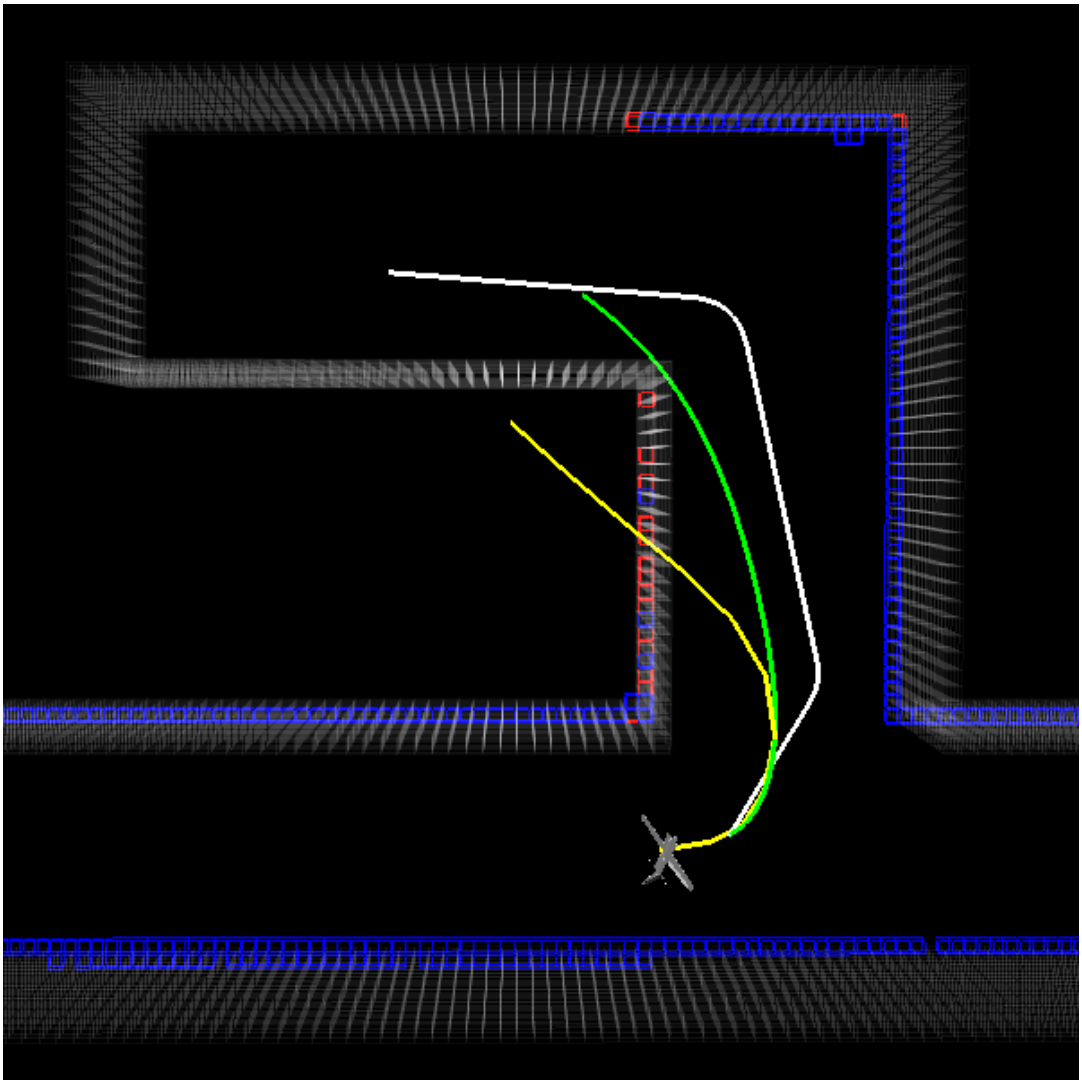

Figure 6.7: Predicted Map

Blue fixed-wing UAV: ground truth state
Green fixed-wing UAV: IMU integrated state
Red fixed-wing UAV: S-UKF-LG state estimation.
White trajectory: smoothed RRT
Yellow trajectory: direct transcription trajectory currently being tracked
Green trajectory: direct transcription trajectory being optimized
Translucent White Cells: true map
Blue Cells: observed map
Red Cells: predicted map

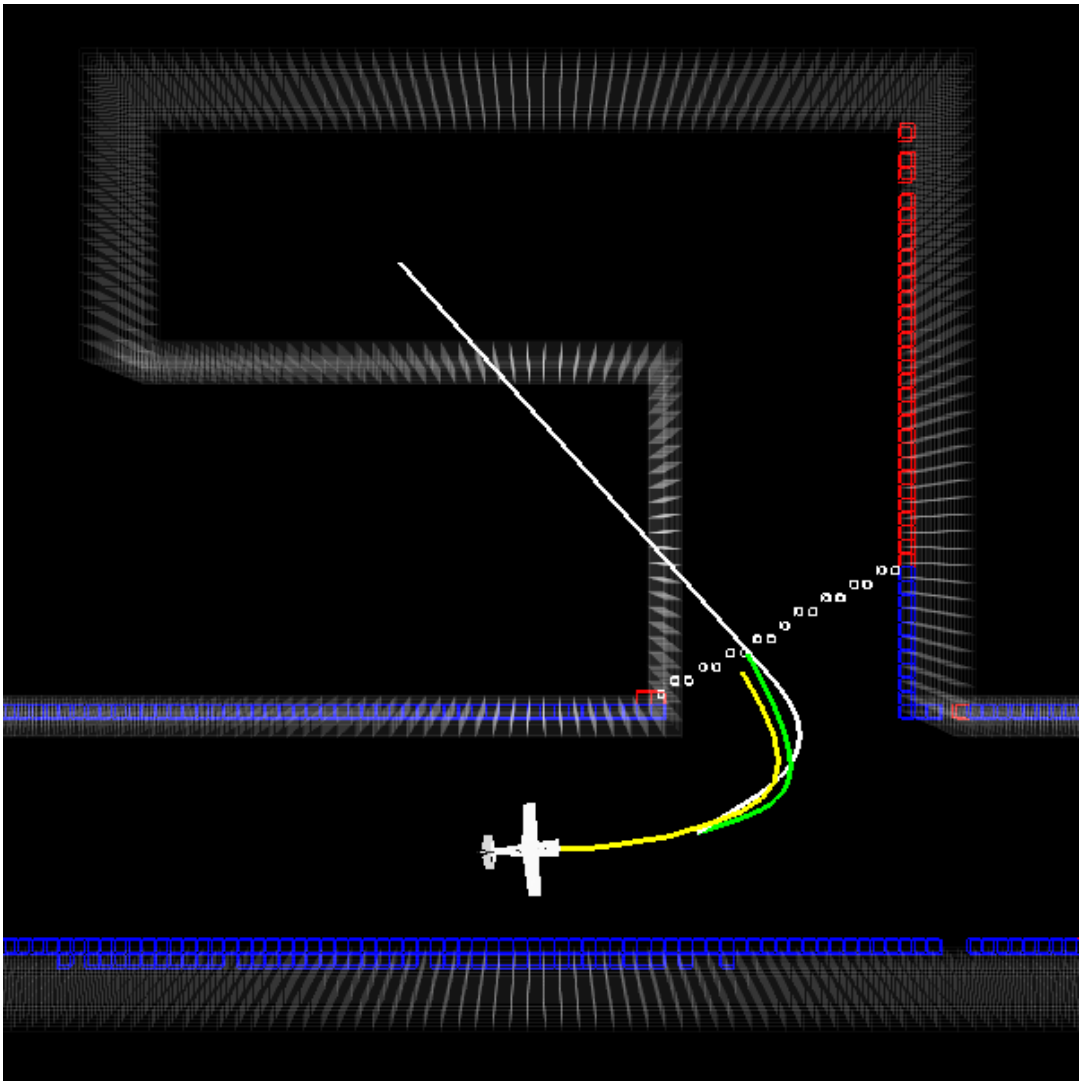

Figure 6.8: Frontier and Predicted Map

Blue fixed-wing UAV: ground truth state
Green fixed-wing UAV: IMU integrated state
Red fixed-wing UAV: S-UKF-LG state estimation.
White trajectory: smoothed RRT
Yellow trajectory: direct transcription trajectory currently being tracked
Green trajectory: direct transcription trajectory being optimized
Translucent White Cells: true map
Blue Cells: observed map
Red Cells: predicted map
Solid White Cells: detected frontier

6.5 Simulation

The system is run both with the LIDAR sensor and the stereo camera sensor. The LIDAR sensor is simulated using C++. It has a horizontal range of $\frac{\pi}{3}$ radians, 100 beams, and a range of 10 meters. It is simulated at a rate of 100 Hz. The sensor is provided with an occupancy map, which it generates measurements from using ray casting. An image of the sensor can be seen in Figure 6.9.

It is assumed that the controller has access to the ground truth system state, but not the ground truth map. The initial state is randomized to evaluate robustness. Navigation performance is compared using the time horizon, horizon point selection, and prediction methods explained above.

6.6 Results

6.6.1 Time Horizon

Directly planning on the unknown map, using the feedback motion planning framework described in the last chapter, is often unsuccessful (Figures 6.10 and 6.11. When turning the first corner of the hallway, the system's view of the inside wall is obscured. As a result, trajectories are planned through it. Once the wall is revealed to the system, the optimizer is unable to generate a viable trajectory for the system within the major iterations constraint. This is due to the fact that the prior trajectory being used for seeding is infeasible. Also note that the lidar and stereo camera sensors result in very similar maps.

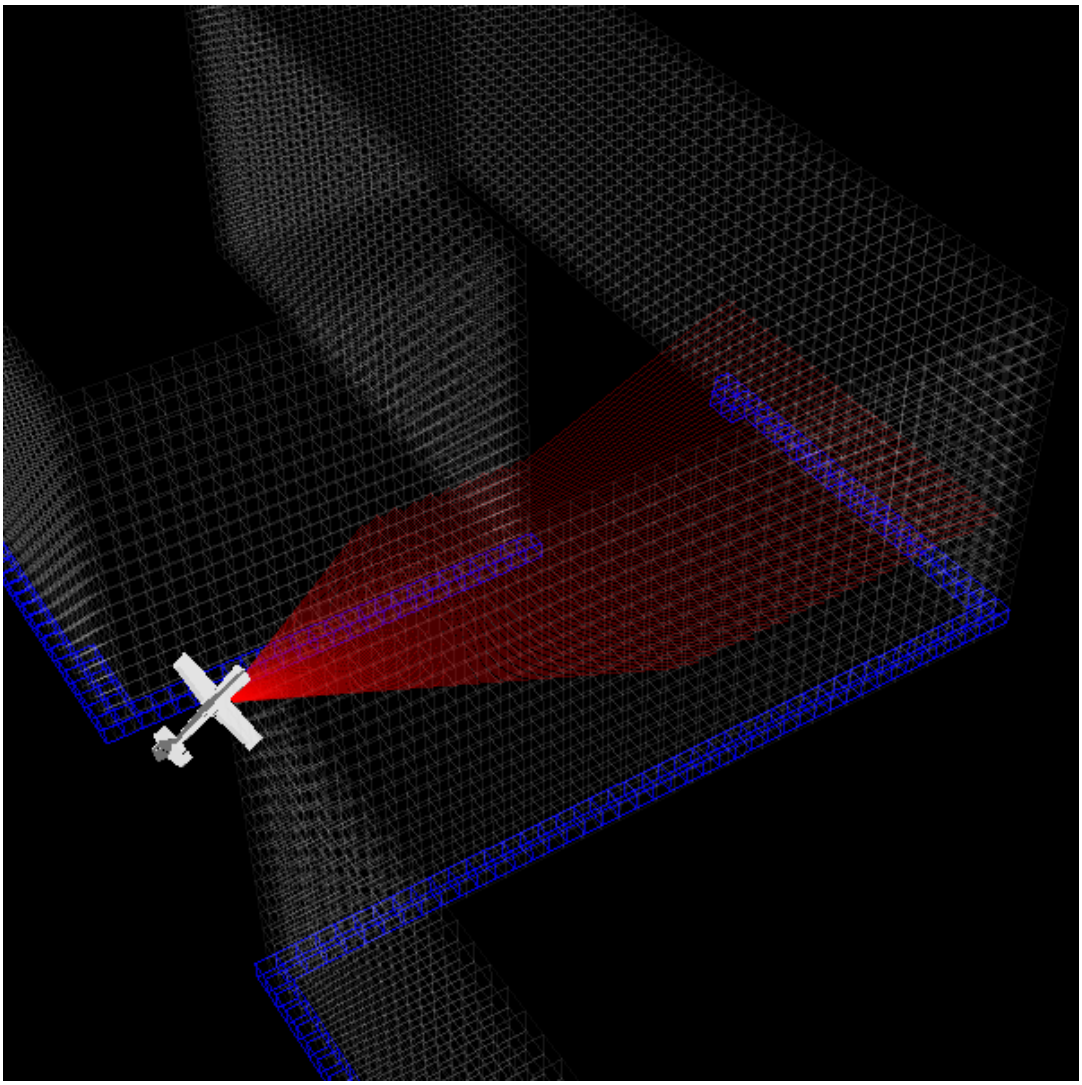

Figure 6.9: Lidar Sensor

Red lines: Lidar Beams
Translucent White Cells: true map
Blue Cells: observed map

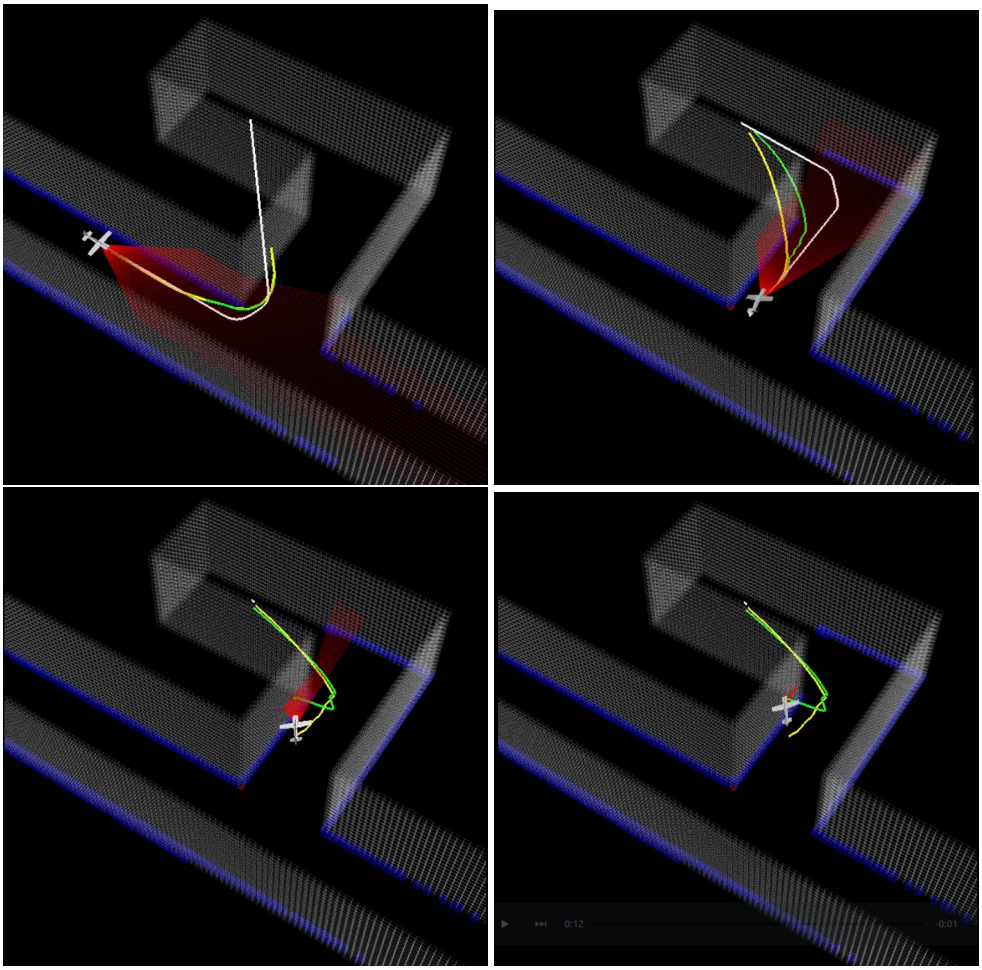

Figure 6.10: Time Horizon planning with LIDAR

Red lines: Lidar Beams
White trajectory: smoothed RRT
Yellow trajectory: direct transcription trajectory currently being tracked
Green trajectory: direct transcription trajectory being optimized
Translucent White Cells: true map
Blue Cells: observed map

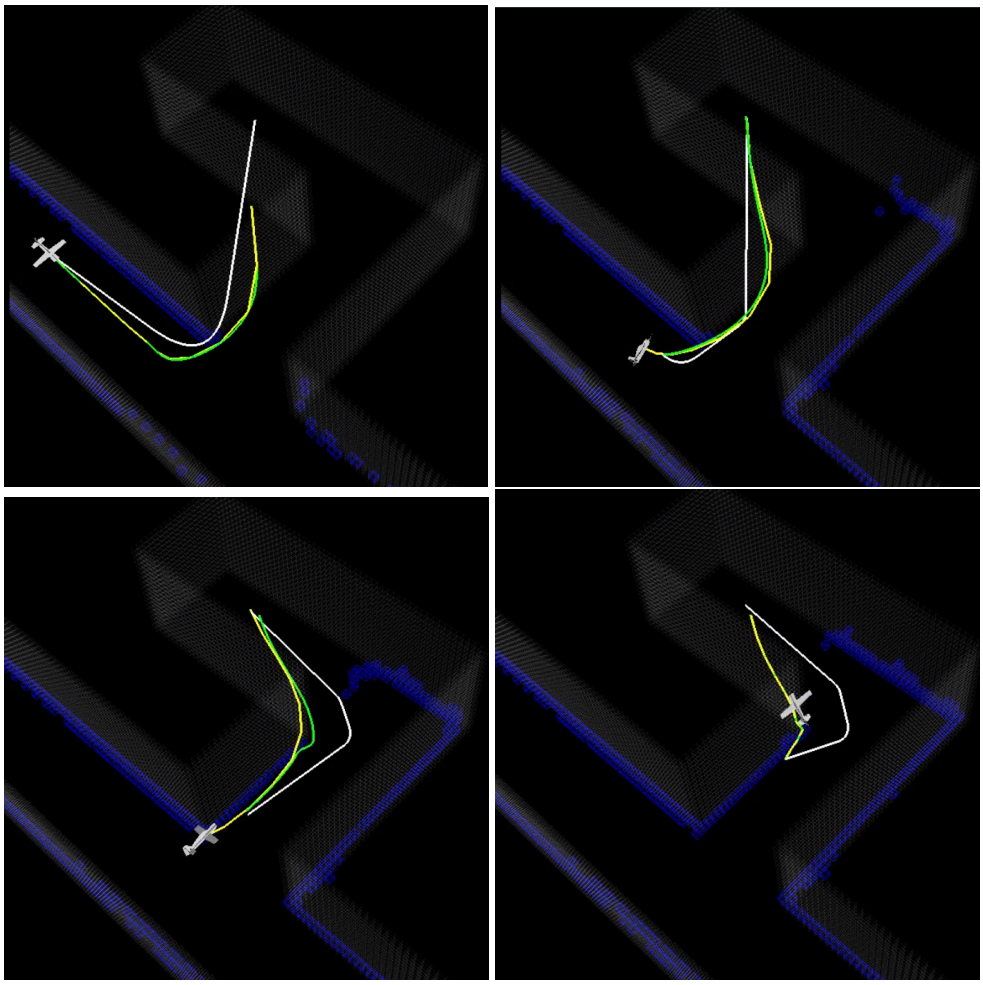

Figure 6.11: Time Horizon planning with stereo camera

Blue fixed-wing UAV: ground truth state
Green fixed-wing UAV: IMU integrated state
Red fixed-wing UAV: S-UKF-LG state estimation.
White trajectory: smoothed RRT
Yellow trajectory: direct transcription trajectory currently being tracked
Green trajectory: direct transcription trajectory being optimized
Translucent White Cells: true map
Blue Cells: observed map
Red Cells: predicted map
Solid White Cells: detected frontier

6.6.2 Frontier Midpoint

When planning to the frontier midpoint, the system is able to successfully reach the goal point (Figure 6.12).

6.6.3 Planning with Predicted Map

When planning based on the predicted map, the system is able to reach the goal point more often than when planning on the current map (Figure 6.13).

6.6.4 Planning with Frontier and Predicted Map

When planning to the intersection between the frontier and the predicted map RRT, the system is able to successfully reach the goal point (Figure 6.14).

6.6.5 Comparison

The simulation was run ten times using each strategy to compare their effectiveness. This was done with both the LIDAR sensor and the stereo camera sensor. The initial states were randomized using a uniform distribution, $\mathbf{U}[-0.2, 0.2]$ for the position states and $\mathbf{U}[-0.1, 0.1]$ for the orientation states. Figures 6.15 and 6.16 compare the trajectories of the system without frontier planning (using the standard time horizon approach), with frontier midpoint planning, and with frontier and map prediction planning.

Additionally, the simulation was run using the time horizon approach with different prediction lengths to compare the effectiveness of map prediction alone. In these cases, frontiers are not detected or utilized and the RRT is generated on the prediction map. Figures 6.17 and 6.18 compare the trajectories

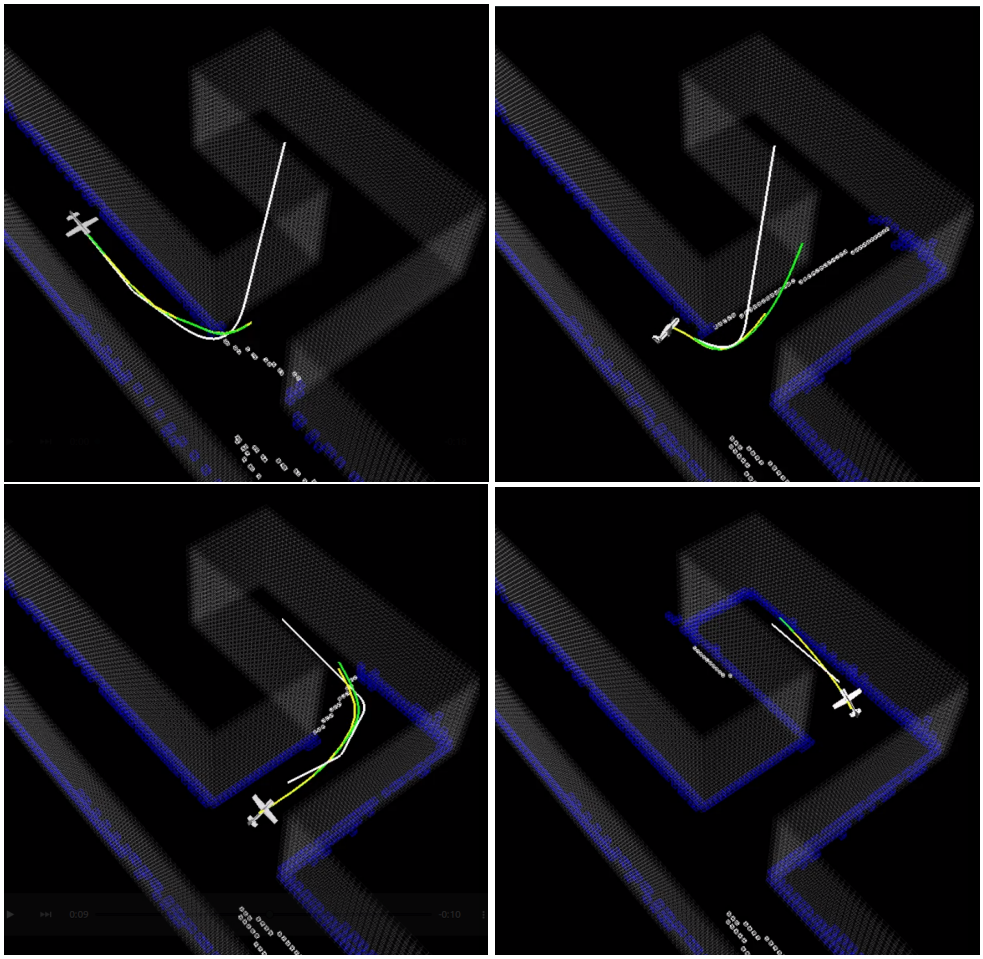

Figure 6.12: Frontier Midpoint with stereo camera

White trajectory: smoothed RRT
Yellow trajectory: direct transcription trajectory currently being tracked
Green trajectory: direct transcription trajectory being optimized
Translucent White Cells: true map
Blue Cells: observed map
Solid White Cells: detected frontier

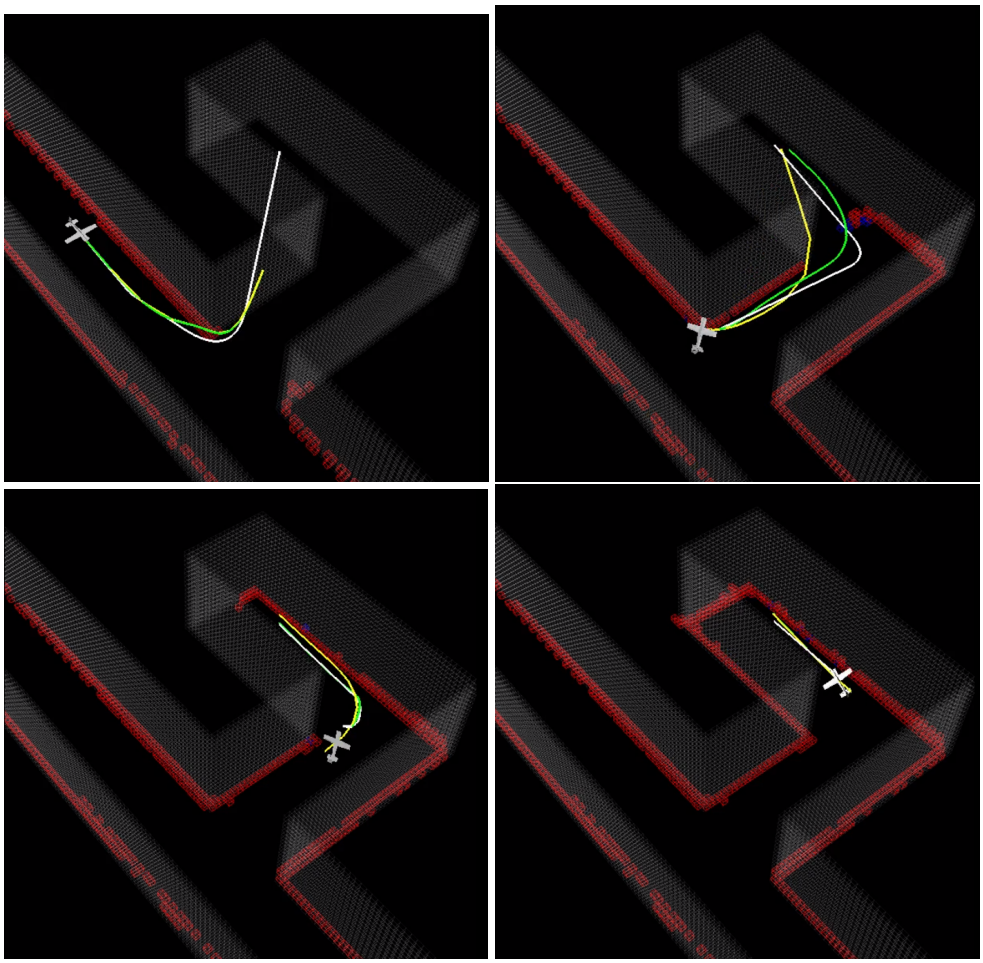

Figure 6.13: Predicted Map with stereo camera

White trajectory: smoothed RRT
Yellow trajectory: direct transcription trajectory currently being tracked
Green trajectory: direct transcription trajectory being optimized
Translucent White Cells: true map
Blue Cells: observed map
Red Cells: predicted map

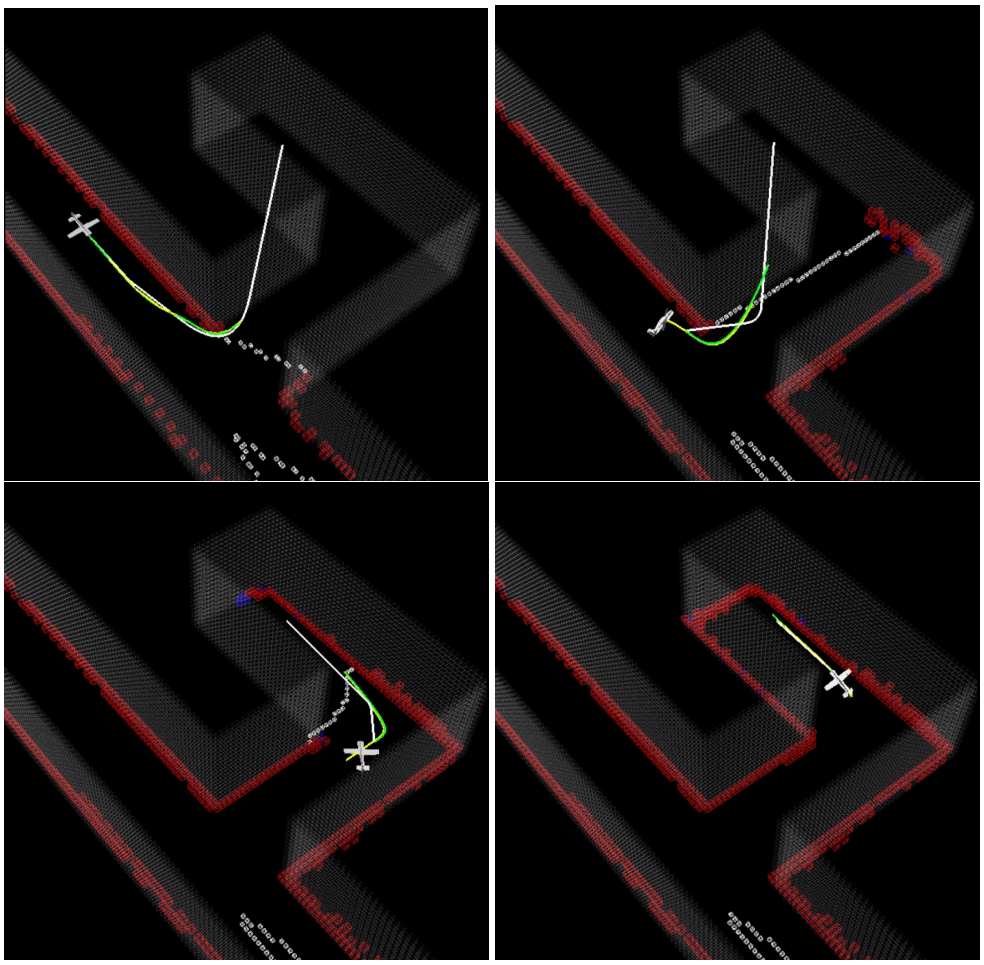

Figure 6.14: Frontier and Predicted Map with stereo camera

White trajectory: smoothed RRT
Yellow trajectory: direct transcription trajectory currently being tracked
Green trajectory: direct transcription trajectory being optimized
Translucent White Cells: true map
Blue Cells: observed map
Red Cells: predicted map
Solid White Cells: detected frontier

of the system without prediction, with 0.05 sec of prediction, and with 0.1 sec of prediction. A summary of success rates with different frontier selection methods and prediction times is show in Tables 6.1 and 6.2, respectively. A successful simulation run is defined as the system reaching within 0.5 meters of the goal position without collision. It is clear from this data that both the frontier method and the prediction method perform better; in simulation, the system tended to fly into obstacles less often.

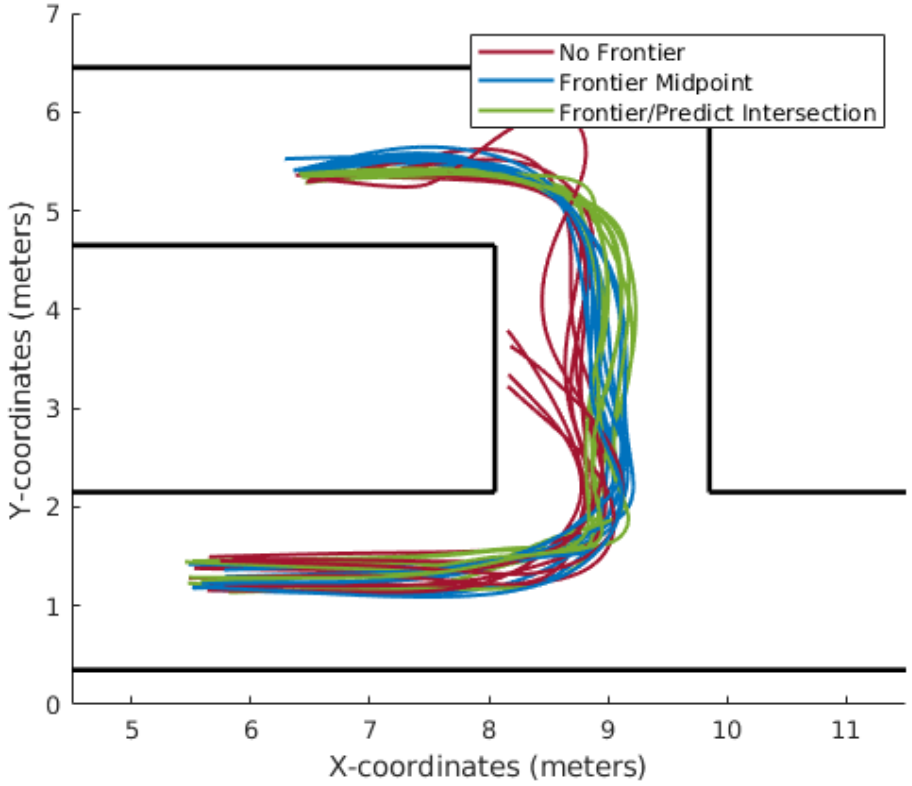

Figure 6.15: Comparison of trajectories using frontier methods using LIDAR

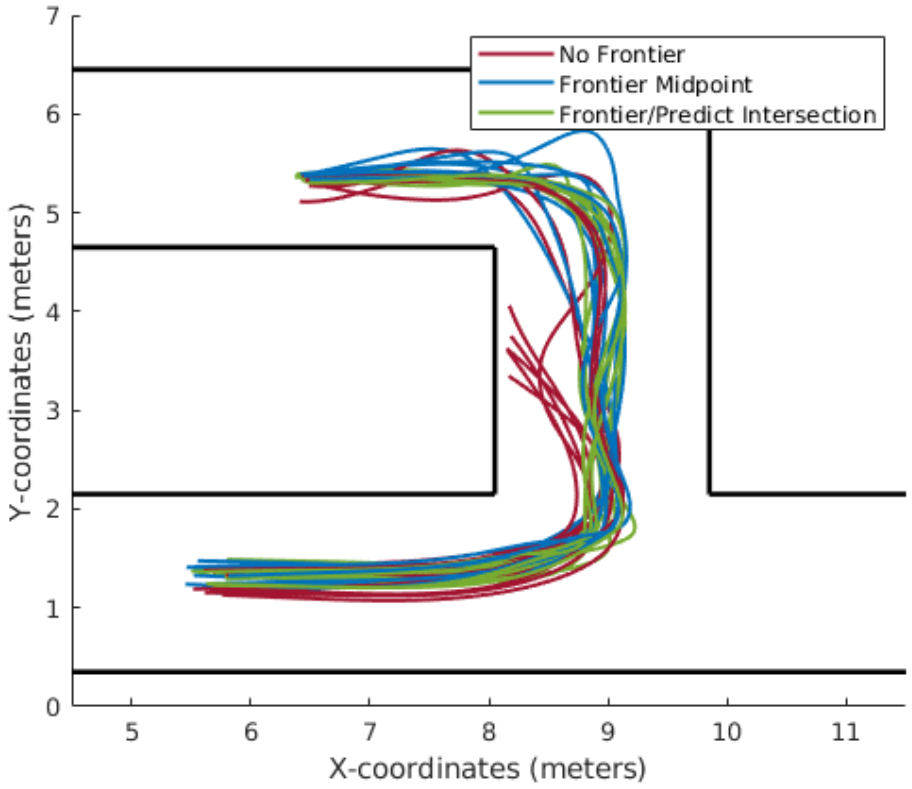

Figure 6.16: Comparison of trajectories using frontier methods using stereo camera

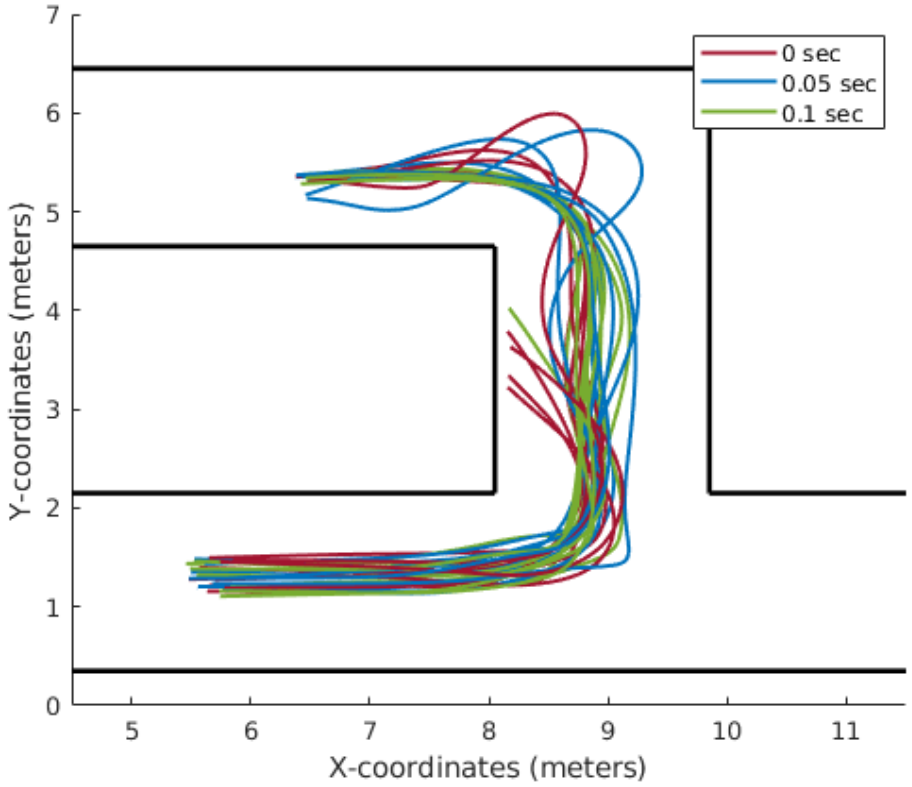

Figure 6.17: Comparison of trajectories using different map prediction times, and no frontiers, using LIDAR

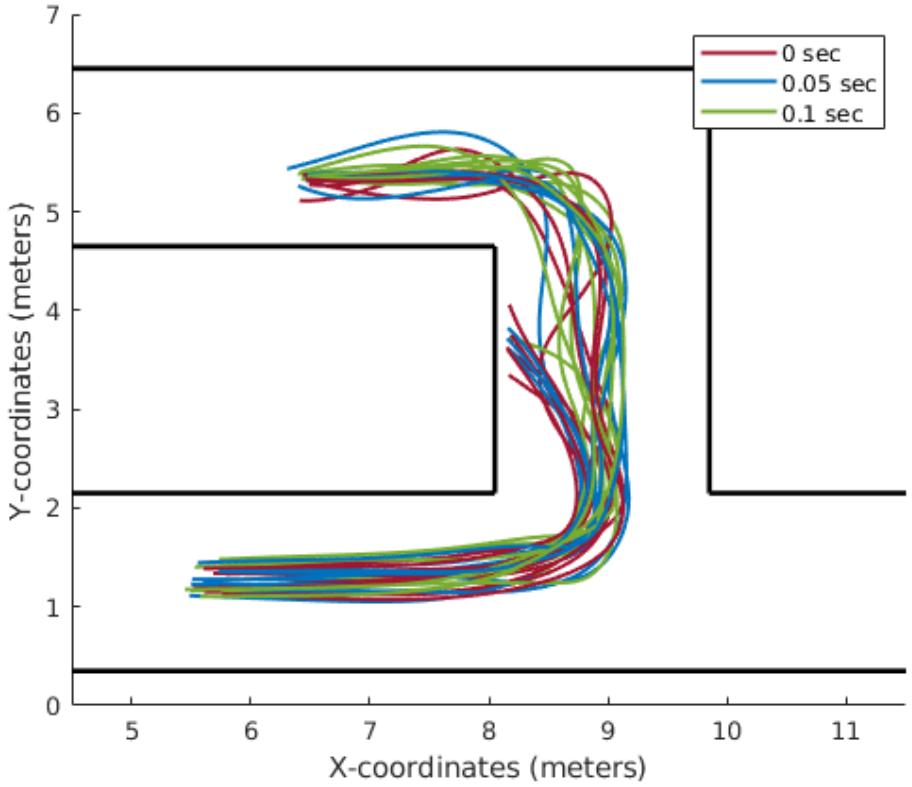

Figure 6.18: Comparison of trajectories using different map prediction times, and no frontiers, using stereo camera

Sensor	No Frontier	Frontier Midpoint	Frontier/Predict intersection
LIDAR	6/10	10/10	10/10
Camera	5/10	10/10	10/10

Table 6.1: Frontier methods success rates

Success rates across ten simulation runs. A successful simulation run is defined as the system reaching within 0.5 meters of the goal position without collision.
No Frontier: direct transcription goal position is a time horizon
Frontier Midpoint: direct transcription goal position is the midpoint of a frontier
Frontier/Predict Intersection: direct transcription goal position is the intersection of the RRT on the predicted map and a frontier on the known map

Sensor	0 sec	0.05 sec	0.10 sec
LIDAR	6/10	10/10	9/10
Camera	5/10	7/10	9/10

Table 6.2: Prediction map success rates

Success rates across ten simulation runs. For these runs, no frontiers are detected or utilized. Direct transcription goal positions are selected according to a time horzion along an RRT generated in a predicted map. A successful simulation run is defined as the system reaching within 0.5 meters of the goal position without collision.
0 sec: no predicted map is used
0.05 sec: predicted map generated by simulating sensor forward by 0.05 sec
0.10 sec: predicted map generated by simulating sensor forward by 0.10 sec

Chapter 7

Simultaneous Navigation, Mapping, and State Estimation

The fixed-wing UAV is simulated while performing mapping and state estimation; the system itself does not have access to the ground truth state or map. Given the results in Chapters 5 and 6, the simulation is run using S-UKF-LG, without the visibility cost function, and using the frontier midpoint NMPC method. The IMU has Gaussian noise added to acceleration ($\frac{m}{s^2}$) and angular velocity ($\frac{rad}{s}$) measurements, $\mathcal{N}(0, \text{diag}(1,1,1))$ and $\mathcal{N}(0, \text{diag}(0.1, 0.1, 0.1))$ respectively. The acceleration and angular velocity measurements also have added biases, $[0.1, 0.1, 0.1]$ and $[0.01, 0.01, 0.01]$ respectively. The camera is simulated at 30Hz.

The simulation is run ten times from the same initial state. Figure 7.1 displays an example run, and Figure 7.2 displays the trajectories associated with an example run. Figure 7.3 displays the error associated S-UKF-LG state estimation and with naive IMU integration. Figure 7.4 displays the trajectories of the ten simulation runs. In nine out of ten runs, the fixed-wing UAV was able to reach within 0.5 meters of the goal position. In one cases, the system

reached within one meter of the goal position, but collided with the wall at the very end of its trajectory.

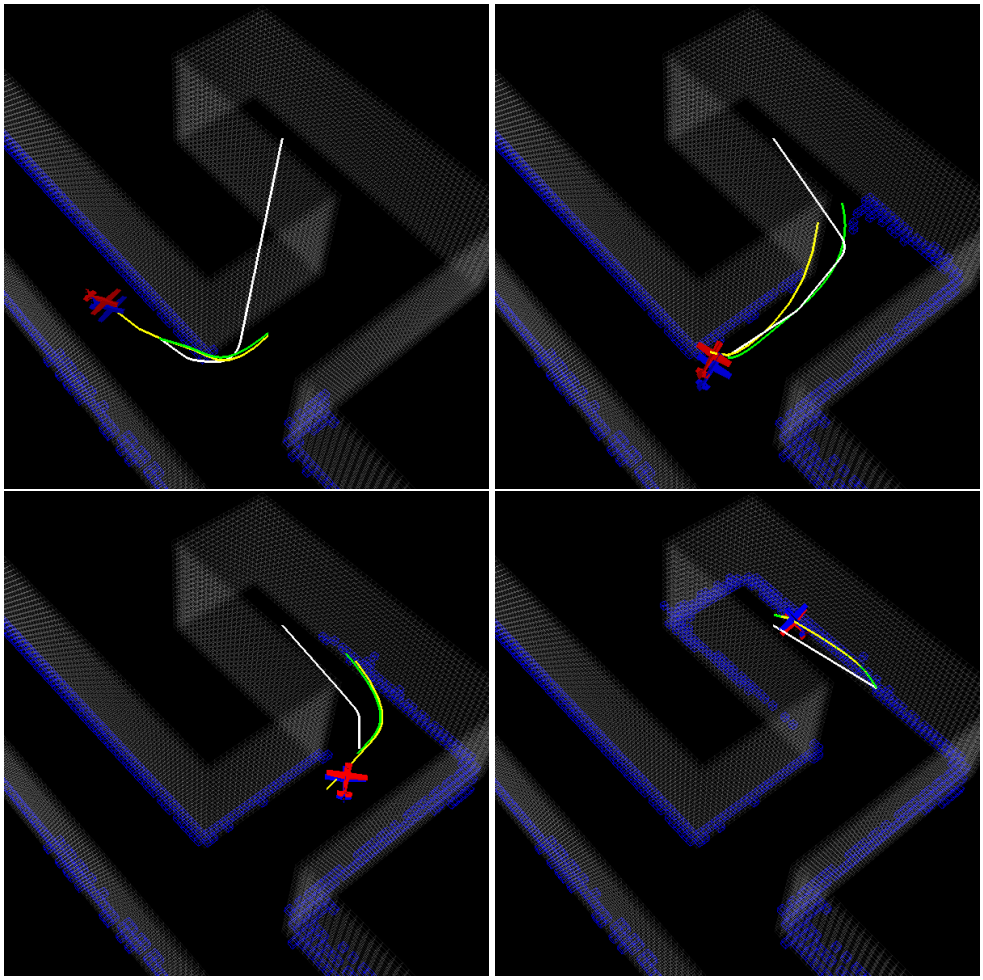

Figure 7.1: Trajectory of simulated fixed-wing while performing mapping and state estimation. Blue fixed-wing is ground truth state red fixed-wing is UKF S_MSCKF state estimation Blue map is built from camera

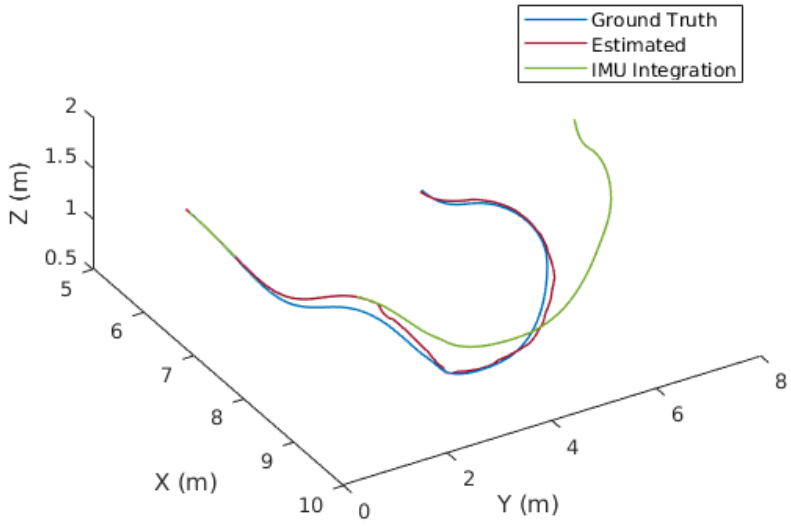

Figure 7.2: Visualization of example trajectory

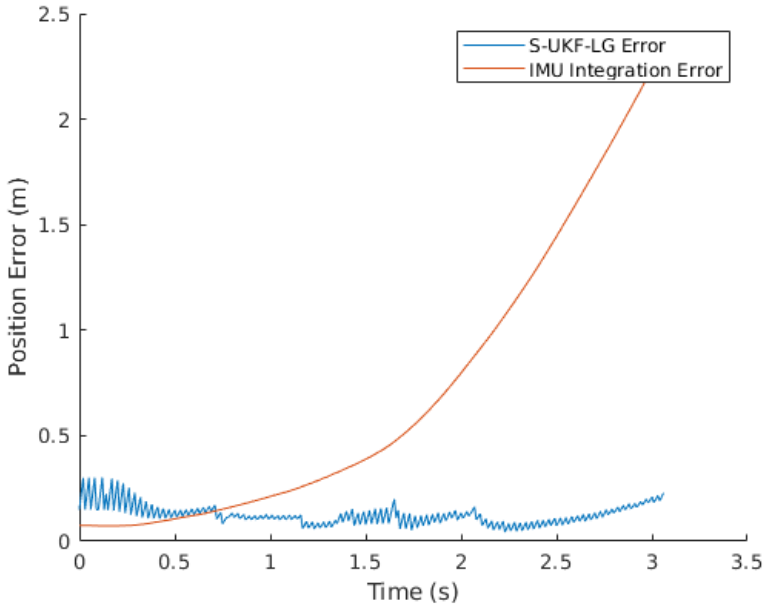

Figure 7.3: Errors associated with example trajectory

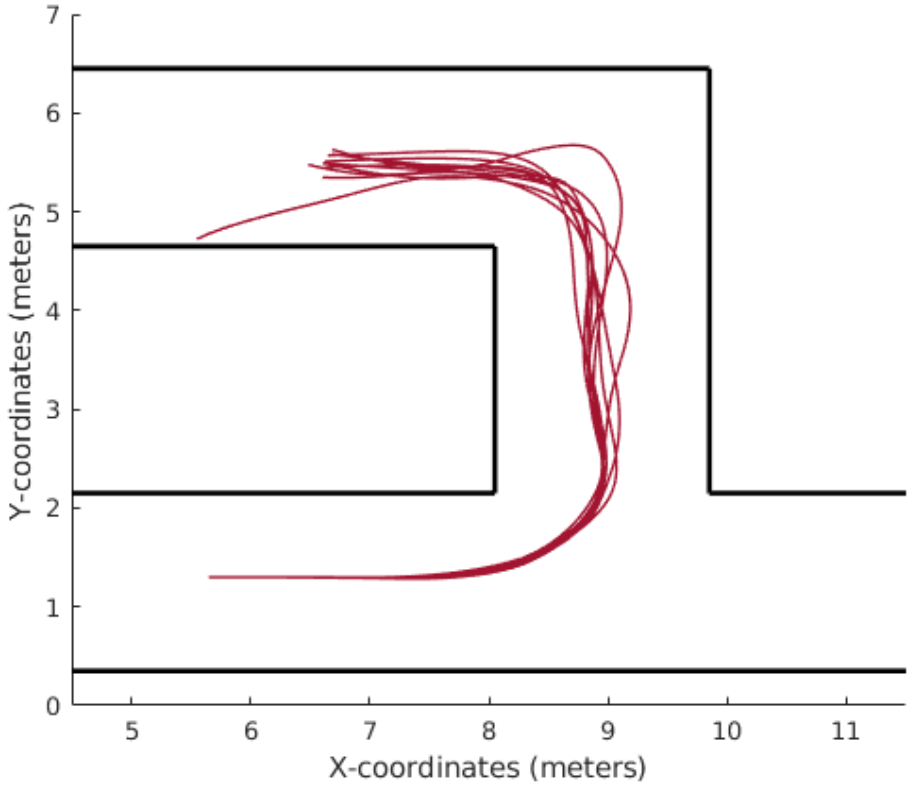

Figure 7.4: Trajectories of simulated fixed-wing while performing mapping and state estimation.

Chapter 8

Discussion and Conclusion

This essay investigate the use of stereo vision to aid in autonomous navigation of aerobatic fixed-wing UAVs. To improve state estimation, a visibility metric was integrated into a direct NMPC method and was evaluated in simulation. While results were not particularly promising, this the first instance of perception aware direct NMPC for fixed-wing UAVs that we are aware of. Perception aware trajectory generation is especially helpful in environments that contain few observable features. The simulation environment used in this essay was rich with features, which may indicate why no significant improvement was observed while using the visibility metric. Future experiments will expand upon this research using new simulation environment with varying levels of observable features.

Performing direct NMPC using a time horizon while in an unknown environment was found to be ineffective. Due to time constraints on the optimizer, the system was often unable to re-plan trajectories effectively when new obstacles were observed. Replacing the time-horizon with a frontier-based horizon resolved this issue. Furthermore, the use of map predictions for

planning was investigate. It was found that if the system could have access to an accurate prediction of the future map, even as little as a prediction of 0.1 seconds ahead, obstacle avoidance behavior was much improved. Through the use of the S-UKF-LG state estimator and the frontier-based NMPC, the fixed-wing UAV was able to navigate to the goal aread with a high rate of success, even while planning with an unknown state on an unknown map.

All simulations in this essay were run in real time. Future work will include running these experiments on board hardware. This transition will introduce new challenges, such as motion blur in camera images and external disturbances to the dynamics, such as wind. However, this will also be an opportunity to explore other vision sensors, such as RGB-D cameras and tracking cameras. These cameras are often capable of performing on-board processing, such as the Intel Realsense cameras (*Intel RealSense Technology*), that may help with the computation load.

References

Basescu, Max and Joseph Moore (2020). "Direct NMPC for Post-Stall Motion Planning with Fixed-Wing UAVs". In: *arXiv preprint arXiv:2001.11478*.

Milam, Mark B, Ryan Franz, and Richard M Murray (2002). "Real-time constrained trajectory generation applied to a flight control experiment". In: *IFAC Proceedings Volumes* 35.1, pp. 175–180.

Matsumoto, Takaaki, Atsushi Konno, Ren Suzuki, Atsushi Oosedo, Kenta Go, and Masaru Uchiyama (2010). "Agile turnaround using post-stall maneuvers for tail-sitter VTOL UAVs". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1612–1617.

Moore, Joseph, Rick Cory, and Russ Tedrake (2014). "Robust post-stall perching with a simple fixed-wing glider using LQR-Trees". In: *Bioinspiration & biomimetics* 9.2, p. 025013.

Stastny, Thomas J., Adyasha Dash, and Roland Siegwart (2017). "Nonlinear MPC for Fixed-wing UAV Trajectory Tracking: Implementation and Flight Experiments". en. In: *AIAA Guidance, Navigation, and Control Conference*. Grapevine, Texas: American Institute of Aeronautics and Astronautics. ISBN: 978-1-62410-450-3. DOI: 10.2514/6.2017-1512. URL: http://arc.aiaa.org/doi/10.2514/6.2017-1512 (visited on 09/04/2019).

Garimella, Gowtham, Matthew Sheckells, Joseph Moore, and Marin Kobilarov (2018). "Robust Obstacle Avoidance using Tube NMPC". en. In: *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation. ISBN: 978-0-9923747-4-7. DOI: 10.15607/RSS.2018.XIV.055. URL: http://www.roboticsproceedings.org/rss14/p55.pdf (visited on 08/29/2019).

Dadkhah, Navid and BÃl'rÃl'nice Mettler (2012). "Survey of Motion Planning Literature in the Presence of Uncertainty: Considerations for UAV Guidance". en. In: *Journal of Intelligent & Robotic Systems* 65.1-4, pp. 233–246. ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-011-9642-9. URL: http://link.springer.com/10.1007/s10846-011-9642-9 (visited on 09/10/2019).

Mourikis, A. I. and S. I. Roumeliotis (2007a). "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3565–3572. DOI: 10.1109/ROBOT.2007.364024.

Sun, K., K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar (2018). "Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight". In: *IEEE Robotics and Automation Letters* 3.2, pp. 965–972. ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2793349.

Brossard, M., S. Bonnabel, and A. Barrau (2018). "Unscented Kalman Filter on Lie Groups for Visual Inertial Odometry". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 649–655. DOI: 10.1109/IROS.2018.8593627.

Richter, Charles and Nicholas Roy (2017). "Learning to Plan for Visibility in Navigation of Unknown Environments". en. In: *2016 International Symposium on Experimental Robotics*. Ed. by Dana KuliÄĞ, Yoshihiko Nakamura, Oussama Khatib, and Gentiane Venture. Springer Proceedings in Advanced Robotics. Springer International Publishing, pp. 387–398. ISBN: 978-3-319-50115-4.

Katyal, K., K. Popek, C. Paxton, P. Burlina, and G. D. Hager (2019). "Uncertainty-Aware Occupancy Map Prediction Using Generative Networks for Robot Navigation". In: *2019 International Conference on Robotics and Automation (ICRA)*, pp. 5453–5459. DOI: 10.1109/ICRA.2019.8793500.

Bansal, Somil, Varun Tolani, Saurabh Gupta, Jitendra Malik, and Claire Tomlin (2019). "Combining Optimal Control and Learning for Visual Navigation in Novel Environments". In: *arXiv:1903.02531 [cs]*. URL: http://arxiv.org/abs/1903.02531 (visited on 08/29/2019).

Bajcsy, Andrea, Somil Bansal, Eli Bronstein, Varun Tolani, and Claire J. Tomlin (2019). "An Efficient Reachability-Based Framework for Provably Safe Autonomous Navigation in Unknown Environments". In: *arXiv:1905.00532 [cs]*. URL: http://arxiv.org/abs/1905.00532 (visited on 08/29/2019).

Mihaylova, Lyudmila, Tine Lefebvre, Herman Bruyninckx, Klaas Gadeyne, and Joris De Schutter (2003). "A Comparison of Decision Making Criteria and Optimization Methods for Active Robotic Sensing". en. In: *Numerical Methods and Applications*. Ed. by Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Ivan Dimov, Ivan Lirkov, Svetozar Margenov, and Zahari Zlatev. Vol. 2542. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 316–324. ISBN: 978-3-540-00608-4 978-3-540-36487-0. DOI: 10.1007/3-540-36487-0_35.

URL: http://link.springer.com/10.1007/3-540-36487-0_35 (visited on 09/04/2019).

Leung, Cindy, Shoudong Huang, Ngai Kwok, and Gamini Dissanayake (2006). "Planning under uncertainty using model predictive control for information gathering". en. In: *Robotics and Autonomous Systems* 54.11, pp. 898–910. ISSN: 09218890. DOI: 10.1016/j.robot.2006.05.008. URL: https://linkinghub.elsevier.com/retrieve/pii/S0921889006000972 (visited on 09/04/2019).

Le Ny, Jerome and George J. Pappas (2009). "On trajectory optimization for active sensing in Gaussian process models". en. In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. Shanghai, China: IEEE, pp. 6286–6292. ISBN: 978-1-4244-3871-6. DOI: 10.1109/CDC.2009.5399526. URL: http://ieeexplore.ieee.org/document/5399526/ (visited on 09/04/2019).

Charrow, Benjamin, Sikang Liu, Vijay Kumar, and Nathan Michael (2015). "Information-theoretic mapping using Cauchy-Schwarz Quadratic Mutual Information". en. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA: IEEE, pp. 4791–4798. ISBN: 978-1-4799-6923-4. DOI: 10.1109/ICRA.2015.7139865. URL: http://ieeexplore.ieee.org/document/7139865/ (visited on 09/03/2019).

Frey, Kristoffer M, Ted J Steiner, and Jonathan P How (2019). "Towards Online Observability-Aware Trajectory Optimization for Landmark-based Estimators". In: *arXiv preprint arXiv:1908.03790*.

LaValle, Steven M (1998). "Rapidly-exploring random trees: A new tool for path planning". In:

Canon, Michael D, Clifton D Cullum Jr, and Elijah Polak (1970). "Theory of optimal control and mathematical programming". In:

Kobilarov, Marin (2019b). *Applied Optimal Control Lecture Notes: Numerical Methods for Deterministic Optimal Control*.

Kobilarov, Marin (2019a). *Applied Optimal Control Lecture Notes: Linear-Quadratic Regulator Basics*.

Findeisen, Rolf and Frank Allgöwer (2002). "An introduction to nonlinear model predictive control". In: *21st Benelux meeting on systems and control*. Vol. 11. Technische Universiteit Eindhoven Veldhoven Eindhoven, The Netherlands, pp. 119–141.

Julier, Simon J and Jeffrey K Uhlmann (1997). "New extension of the Kalman filter to nonlinear systems". In: *Signal processing, sensor fusion, and target*

recognition VI. Vol. 3068. International Society for Optics and Photonics, pp. 182–193.

Konolige, Kurt (1998). "Small vision systems: Hardware and implementation". In: *Robotics research*. Springer, pp. 203–212.

Shen, Wei (2020). *Computer Vision Lecture Notes: Camera Calibration and Photogrammetry*.

Hornung, Armin, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard (2013). "OctoMap: An efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous robots* 34.3, pp. 189–206.

Hoerner, SH (1985). "Fluid-dynamic lift". In: *Hoerner Fluid Dynamics*.

Yang, Kwangjin and Salah Sukkarieh (2010). "An analytical continuous-curvature path-smoothing algorithm". In: *IEEE Transactions on Robotics* 26.3, pp. 561–568.

Pardo, Diego, Lukas Möller, Michael Neunert, Alexander W Winkler, and Jonas Buchli (2016). "Evaluating direct transcription and nonlinear optimization methods for robot motion planning". In: *IEEE Robotics and Automation Letters* 1.2, pp. 946–953.

Gill, P., W. Murray, and M. Saunders (2005). "SNOPT: An SQP algorithm for large-scale constrained optimization". In: *SIAM review* 47.1, pp. 99–131.

Brossard, M., S. Bonnabel, and J. Condomines (2017a). "Unscented Kalman filtering on Lie groups". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2485–2491. DOI: 10.1109/IROS.2017.8206066.

Burri, Michael, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, M Achtelik, and Roland Siegwart (2015). "The euroc mav datasets". In: *The International Journal of Robotics Research*.

Leutenegger, Stefan, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale (2015). "Keyframe-based visual–inertial odometry using nonlinear optimization". In: *The International Journal of Robotics Research* 34.3, pp. 314–334.

Qin, Tong, Peiliang Li, and Shaojie Shen (2018). "Vins-mono: A robust and versatile monocular visual-inertial state estimator". In: *IEEE Transactions on Robotics* 34.4, pp. 1004–1020.

Mourikis, Anastasios I and Stergios I Roumeliotis (2007b). "A multi-state constraint Kalman filter for vision-aided inertial navigation". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, pp. 3565–3572.

Brossard, Martin, Silvere Bonnabel, and Jean-Philippe Condomines (2017b). "Unscented Kalman filtering on Lie groups". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2485–2491.

Intel RealSense Technology. URL: https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html.

Adam Polevoy

7110 Balmy Dew Way, Columbia, MD • (201)-560-6265 • adampolevoy@gmail.com

EDUCATION

Johns Hopkins University **Baltimore, MD**

- Bachelor of Science: Biomedical Engineering, CS Minor (GPA: 3.92) **May 2019**
- Master of Science in Engineering: Robotics (GPA: 3.85) **Expected May 2020**

WORK EXPERIENCE

Johns Hopkins Applied Physics Laboratory (Intern, Intelligent Systems) **Baltimore, MD; May 2018 – Present**

- Real Time Trajectory Optimization for Vision Based Navigation with Aerobatic Fixed Wing Vehicles
  - Integrated and evaluated visibility metric with direct NMPC of a fixed-wing UAV
  - Formulated and evaluated novel frontier-based NMPC method for improved obstacle avoidance
- EMG Control of Wearable Robotic Suit
  - Added new features, classifiers, and devices to EMG analysis codebase
  - Ran human subject research data collection
  - Wrote codebase for automated data parsing/organization and easy analysis
  - Assessed performance of machine learning algorithms for EMG wearable robot control
- Spiking Neural Network Development
  - Assembled hierarchical databases for use in associative learning and semantic information extraction
  - Implemented framework for easy training of spiking convolutional neural networks from NengoDL
  - Trained deep spiking convolutional neural networks on computing cluster
  - Expanded spiking neural network codebase (Nengo) to include custom learning rules

Johns Hopkins University (Teaching Assistant) **Baltimore, MD; Feb. 2018 – May 2019**

- Classes: Systems and Controls, Systems Bioengineering 1, Systems Bioengineering 2
- Lecture weekly, hold review sessions, develop homework/exam questions
- Ensure students' comprehension of complex concepts through weekly 'office hours'

MoTrack Therapy LLC (Co-Founder & Programmer) **altimore, MD; Oct. 2015 – May 2018**

- Created independent start-up with small team of JHU engineering students
- Developing mobile application for interactive hand therapy and data collection
- Partnering with Johns Hopkins Hospital to undergo clinical trial

EXTRACURRICULARS

Hopkins Taekwondo Club (Assistant Instructor, Treasurer, Leadership Board) **Baltimore, MD; Oct. 2015 – Oct. 2018**

- Organizer and leader of club, ambassador between club and school
- Controls budgeting, purchases, and reimbursements for club spending
- Assists instructor with teaching and training during practices

JHU Biomedical Instrumentation & Neuroengineering Lab (Undergrad. Researcher) **Baltimore, MD; Jan. 2016 – Jan. 2018**

- Analyzed sensor characteristics, presented findings at undergraduate research event
- Developed advanced 3D-printed myoelectric prosthetic finger using Solidworks
- Programmed MATLAB code to quantify heart rate variability

TECHNICAL SKILLS

- MATLAB, Python, Java, C, C++, Linux, Arduino, ROS, Solidworks

KEY COURSEWORK & COURSE PROJECTS

Master's Research: Real Time Trajectory Optimization for Vision Based Navigation with Aerobatic Fixed Wing Vehicles **Fall 2019- Spring 2020**

- Integrated and evaluated visibility metric with direct NMPC of a fixed-wing UAV
- Formulated and evaluated novel frontier-based NMPC method for improved obstacle avoidance
- Simulated state-estimation, mapping, and planning in parallel for fixed-wing UAV to demonstrate real-time performance

Robot System Programming: AR Tag Tracking and Searching **Spring 2020**

- Implemented ROS packages enabling Turtlebot WafflePi to track and search for AR tags
- Integrated Gazebo for realistic simulation
- Integrated existing ROS packages to perform SLAM and frontier exploration

Applied Optimal Control: Perception Aware Trajectory Generation **Fall 2019**

- Implemented perception aware optimal trajectory generation for stochastic dynamical system
- Formulated visibility and covariance based cost functions for use in direct transcription
- Utilized time varying linear quadratic regulator (TVLQR) for feedback control

Robot Devices, Kinematics, Dynamics, and Control: Place & Mark with Intention **Spring 2019**

- Wrote code to teach starting and target position to UR5 robot arm
- Programmed several control schemes to plan a trajectory for the UR5 to follow

Mechatronics: Hockey Playing Robots **Spring 2019**

- Created a goalie robot and forward robot capable of playing hockey
- Designed custom chassis for robots in Solidworks
- PIXY camera, microphones, and IMUs as sensors for puck detection and orientation updating

Computer Integrated Surgery: Tool Gravity Compensation on Galen Microsurgical Robot **Spring 2019**

- Force sensor used by robot was unable to distinguish applied forces and gravitational forces/torques on tool
- Used robot kinematics and least squares method to estimate center of mass and weight of tools
- Compensated force sensor readings online using center of mass, weight, and kinematics

Algorithms for Sensor-Based Robotics: UR5 Obstacle Avoidance **Spring 2018**

- Implemented sampling-based planner to plan a path for a UR5 arm to move through a car door window
- Used Gaussian Sampling Probabilistic Road Map to avoid collision with obstacles
- Programmed in C++, utilized ROS and MoveIt!

Robot Sensors/Actuators: Smart Steering Wheel **Fall 2018**

- Created smart steering wheel capable or alerting drunk or sleepy drivers
- Arduino Uno for computation and control
- BAC calculated with MQ-3 ethanol sensor, sleepiness calculated from angle corrections measured via IMU