# EFFICIENT ALGORITHMS FOR RISK-AVERSE AIR-GROUND RENDEZVOUS MISSIONS

BY

GABRIEL BARSI HABERFELD

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Doctoral Committee:

Professor Naira Hovakimyan, Chair
Professor Srinivasa Salapaka
Professor Dusan Stipanovic
Professor Petros Voulgaris
Professor Evangelos Theodorou

# ABSTRACT

Demand for fast and inexpensive parcel deliveries in urban environments has risen considerably in recent years. A framework is envisioned to enforce efficient last-mile delivery in urban environments by leveraging a network of ride-sharing vehicles, where Unmanned Aerial Systems (UASs) drop packages on said vehicles, which then cover the majority of the distance before final aerial delivery. By combining existing networks we show that the range and efficiency of UAS-based delivery logistics are greatly increased. This approach presents many engineering challenges, including the safe rendezvous of both agents: the UAS and the human-operated ground vehicle. This dissertation presents tools that guarantee risk-optimal rendezvous between the two vehicles. We present mechanical and algorithmic tools that achieve this goal. Mechanically, we develop a novel aerial manipulator and controller that improves in-flight stability during the pickup and drop-off of packages. At a higher level and the core of this dissertation, we present planning algorithms that mitigate risks associated with human behavior at the longest time scales.

First, we discuss the downfalls of traditional approaches. In aerial manipulation, we show that popular anthropomorphic designs are unsuitable for flying platforms, which we tackle with a combination of lightweight design of a delta-type parallel manipulator, and $\mathcal{L}_1$ adaptive control with feedforward. In planning algorithms, we present evidence of erratic driver behavior that can lead to catastrophic failures. Such a failure occurs when the UAS depletes its resource (battery, fuel) and has to crash land on an unplanned location. This is particularly dangerous in urban environments where population density is high, and the probability of harming a person or property in the event of a failure is unsafe. Studies have shown that two types of erratic behavior are common: speed variation and route choice. Speed variation refers to a common disregard for speed limits combined with different levels of comfort per driver. Route choice is conscious, unconscious, or purely random action of deviating

from a prescribed route. Route choice uncertainty is high dimensional and complex both in space and time. Dealing with these types of uncertainty is important to many fields, namely traffic flow modeling. The critical difference to our interpretation is that we frame them in a motion planning framework. As such, we assume each driver has an unknown stochastic model for their behavior, a model that we aim to approximate through different methods.

We aim to guarantee safety by quantifying motion planning risks associated with erratic human behavior. Only missions that plan on using all of the UAS's resources have inherent risk. We postulate that if we have a high assurance of success, any mission can be made to use more resources and be more efficient for the network by completing its objective faster. Risk management is addressed at three different scales. First, we focus on speed variation. We approach this problem with a combination of risk-averse Model Predictive Control (MPC) and Gaussian Processes. We use risk as a measure of the probability of success, centered around estimated future driver position. Several risk measures are discussed and CVaR is chosen as a robust measure for this problem. Second we address *local* route choice. This is route uncertainty for a single driver in some region of space. The primary challenge is the loss of gradient for the MPC controller. We extend the previous approach with a cross-entropy stochastic optimization algorithm that separates gradient-based from gradient-free optimization problems within the planner. We show that this approach is effective through a variety of numerical simulations.

Lastly, we study a city-wide problem of estimating risk among several available drivers. We use real-world data combined with synthetic experiments and Deep Neural Networks (DNN) to produce an accurate estimator. The main challenges in this approach are threefold: DNN architecture, driver model, and data processing. We found that this learning problem suffers from vanishing gradients and numerous local minima, which we address with modern self-normalization techniques and mean-adjusted CVaR. We show the model's effectiveness in four scenarios of increasing complexity and propose ways of addressing its shortcomings.

*To my mom, for her love and support.*

# ACKNOWLEDGMENTS

I would first like to thank my mother Eva, without whom I'd never be where I am. Her unconditional support was paramount to all the events leading to this degree and many other milestones.

Next, I would like to acknowledge my adviser, Prof. Naira Hovakimyan, for her support and input in all aspects of this research. Her guidance throughout my five years at the University of Illinois culminated not only in this dissertation but in a successful career, I'll be forever thankful for every opportunity she has provided me with.

Last but not least I thank my friends and colleagues: Thiago Marinho, Arun Lakshmanan, Alexandre Ormiga, Andrew Patterson, Aditya Gahlawat, Zhouhuan Wu, and many others, for their help, suggestions, discussions, and, most importantly, the enjoyable times we all shared. Additionally, I thank all of my friends in Brazil, who always supported me in all my aspirations.

In the two years leading up to the deposit of this dissertation, mankind faced the relentless threat of the COVID-19 pandemic. I wish that anyone reading this in the future has recovered from the generational trauma and that we remember all those who left us. As such, I end these acknowledgments by thanking scientists, doctors, and medical professionals, who worked tirelessly to save lives and develop vaccines. We are eternally grateful.

*"The saddest aspect of life right now is that science gathers knowledge faster than society gathers wisdom."*

Isaac Asimov, 1988

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

Modern shipping solutions can accumulate more than half of the total shipping cost on the transportation portion between the final distribution center and the customer [3]. This is known as the *last-mile problem*. Our proposed framework consists of using the existing large networks of ride-sharing services (Uber, Lyft) to cover most of the distance from the final distribution center to the customer. This process uses knowledge of these vehicles' destinations to plan deliveries, where an Unmanned Aerial System (UAS) carries the parcel from the distribution center and places it on a moving vehicle, or picks up a package from a moving vehicle and delivers it to a final location. An example scenario is illustrated in Figures 1.1 and 1.2. The critical concern is driver behavior. An erratic driver adds an undesirable risk to the two stages of the mission: (1) landing safely on the moving vehicle to drop the parcel and (2) flying back to the distribution center. During the landing maneuver, small erratic driver inputs are corrected with the aerial manipulation system. At this stage, we are concerned with maintaining effective actuation with robust flight performance. During planning and long-distance flights, the primary source of risk and uncertainty arises from the long-term inexact driver behavior, where a driver might be slower, faster, or erratic in route choice. We refer to the long horizon planning problem as a rendezvous. Environmental factors such as wind, package mass, sloshing of package contents, battery age, and others contribute to these safety concerns, however, we do not focus on that set of disturbances in this dissertation. This dissertation includes efficiency in its title, so as complexity increases we maintain focus on real-time performance and scalability. An argument often thrown at claims of real-time performance is that in the future computation platforms will eventually become fast enough to run any complex algorithm. Although true we counter that argument with two equally true and inevitable facts. First is that there will always be a smaller platform with a less powerful chip in it, so a big drone with a big

chip today, will have similar computational capacity to the small drone with a small chip of tomorrow. Second is that power consumption will always benefit range. If a powerful chip ten years from now can run at half capacity for most of the mission that lower power affords more range.

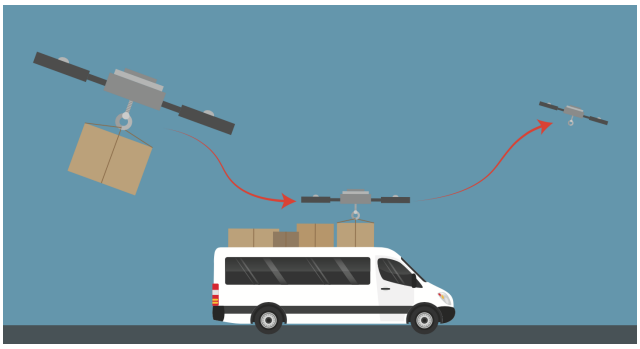


Figure 1.1: Air-ground rendezvous procedure: the UAS needs to meet an uncontrollable ground vehicle with uncertain trajectory

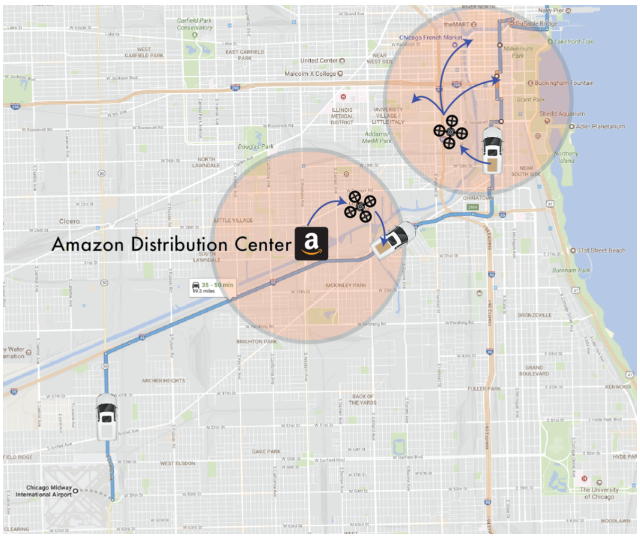



Figure 1.2: Air-ground rendezvous procedure: UASs intercept the vehicle at various points to complete the delivery. In this example an Amazon package is carried by a ride-sharing vehicle departing from Chicago Midway Airport bound to Downtown Chicago.

## 1.1 Aerial Manipulation

Aerial manipulation has undoubtedly become a major robotic research area in the past few years. The industry has driven this interest with investments

in areas such as inspection, mapping, agriculture, age in place, and urban environments. The need for an autonomous aerial vehicle endowed with a robotic arm arises when a simple gripper attached to the bottom of the UAS does not possess enough reach and/or Degrees-of-Freedom (DoF) to execute a given mission with sufficient robustness. Specifically, we argue that adding a manipulator to the flying robots performing the operation shown in Figure 1.2 can facilitate it.

Previous works tackled this problem in one of two ways. Either the system has a large UAS to manipulator mass ratio, giving it high flight actuation overhead, or it is developed for a constrained set of tasks, both of which make the stabilization problem easier to tackle. The former, although certainly feasible, is not representative of the current trend in UAS downsizing [4], and is hardly scalable. The latter is efficient and scalable but undermines the idea of a versatile platform reminiscent of having a flying multirole robotic arm.

Currently, aerial manipulation systems tend to be endowed with a serial manipulator, most notably [5, 6] recently attained strong results. The reasoning for this is understandable, seeing as serial manipulators are more accessible to model, control, design, and implement. They also provide the UAS with the much-needed reach, an area in which these types of manipulators excel. It does, however, come at a cost: the dynamics are *strong*, in the sense that the force/torque wrench generated at the base of the manipulator (i.e. the UAS) is high in magnitude.

In this dissertation, the use of a parallel manipulator is explored. Due to its mass properties and actuator topology, the Delta-type manipulator introduces fewer disturbances to the UAS as well as higher end-effector bandwidth at the cost of a smaller work volume and a complicated model. By designing the manipulator around the UAS these shortcomings are attenuated. In addition, the usefulness of the base to end-effector parallelism as well as the increased speed and precision is attained. In [7] the authors successfully implement a similar parallel manipulator in a UAS, achieving satisfactory performance and task execution. The work utilizes most of the components of this dissertation's approach. Their design mounts the manipulator in such a way that most of the disturbances are transmitted to the (non-critical) yaw axis and, although interesting, this topology severely constrains the set of possible tasks. Ultimately, this makes the vehicle fall into the task-specific category, as it is unable to grasp objects beneath the UAS and, at the same time, imposes a

constant mass asymmetry. In [8] the authors implemented a 6-DoF version of the Delta-type topology, however, their goals revolved around stabilization of the end-effector, instead of pick-and-place tasks.

The design and control of an aerial manipulation system with a low UAS to manipulator mass ratio and satisfactory reach and end-effector bandwidth is developed in this work to provide critical contributions to aerial manipulation research. We build upon previous works [9], which will be contrasted against the design presented here. The UAS is designed to have little flight actuation overhead as well as wide task execution versatility. In Fig. 1.3 a small-scale representation of a UAS capable of dropping off or picking up packages on a moving vehicle is shown. This is achieved by the novel integration of a parallel 3-DoF manipulator with a compact, high-performance UAS. The manipulator is designed to attain a large workspace and fast servo motors are used to reach desired performance indexes. A torque compensating feedforward controller is added to the baseline autopilot to account for the fast dynamics of the manipulator. In addition, an $\mathcal{L}_1$ adaptive controller is designed to account for unknown disturbances and improve tracking performance. A trajectory generation technique is developed where an optimal control algorithm minimizes energy while accounting for the kinematics of the coupled system. The design process of all components is discussed extensively.

Figure 1.3: Unmanned Aerial System equipped with a 3-DoF parallel manipulator.

## 1.2 Rendezvous Planning

It is not an immediate realization that there is an inherent risk in a rendezvous operation. Traditionally, the naive solution is to only fly as far as to where there are enough resources (battery, fuel, energy) to return to a safe landing location. This is a severely sub-optimal solution we wish to avoid. The closest problem to the one we are dealing with here involves commercial aviation long-haul flights [10]. Long-haul flights are usually over oceans or polar caps, where safe landing is impossible. These flights also require the entire fuel capacity of modern airliners and rely on vanishing mass to maximize range. That is, as the flight progresses, the plane's mass decreases monotonically as it spends fuel, decreasing the amount of thrust necessary to maintain flight, and ultimately increasing range. Computing range and fuel requirements for these flights are usually not problematic since industry models are well established and, most importantly for this dissertation, the target landing location does not move. This, however, is not the case when planning a rendezvous with a human driver. Here, we consider dropping a package off on top of a driver or

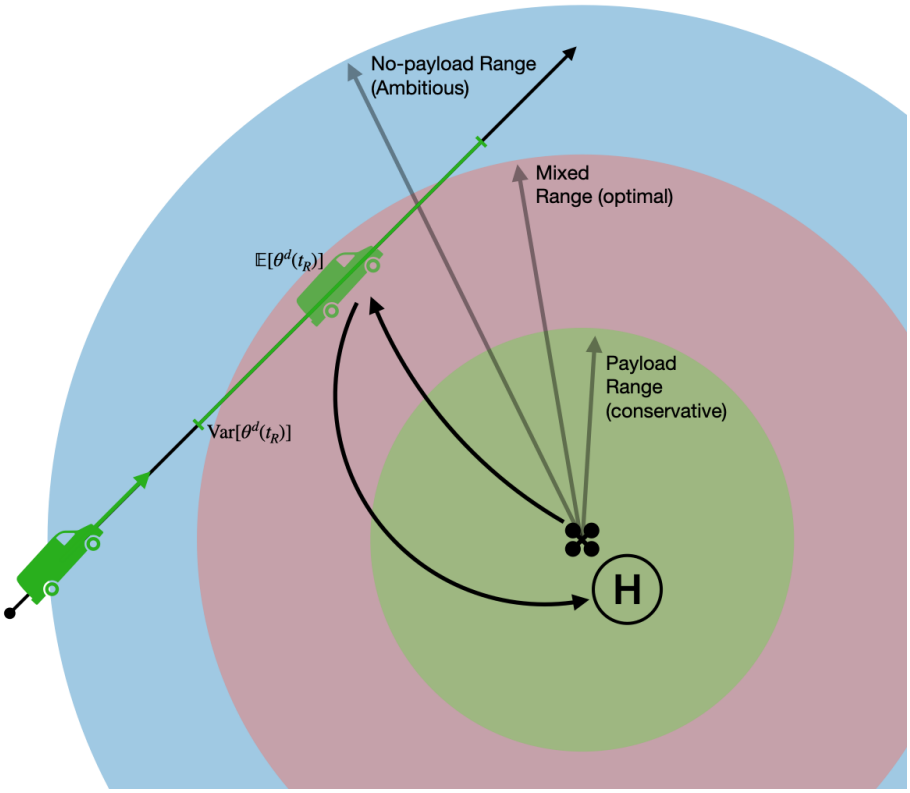



Figure 1.4: Ranges of a payload-varying rendezvous mission.

picking one up. In both cases, the mass of the UAS changes in time, much like an airliner spending fuel. Critically, in the case of a human driver, the landing location is always changing, not because the vehicle is moving but because its future position is uncertain.

Consider Figure 1.4. If we assume a payload is present the entire flight range is severely reduced and assuming no payload does not provide a useful planning stage. We conclude that the optimal solution is to operate with a mixed-payload model. Assuming the payload varies throughout the mission has risk implications. Should the rendezvous not happen precisely in time and space, utilized resources will be higher (since target rendezvous time and location are optimal) causing resource depletion and a subsequent crash landing. To the best of our knowledge, this is a novel problem. The problem of performing a rendezvous with (or intercepting) a moving target is not new. However, these problems fall into two distinct categories: interception of a target on a known path or interception of a target with an unknown trajectory [11]. The interception of a target following a known path is seen as trivial. With full knowledge of the target behavior, infinitely many trajectories will intercept

the target at a chosen time [12]. In these scenarios, the goal is to find the *optimal* trajectory for the interception. Added uncertainties such as model discrepancies and environmental disturbances are often included but do not change the architecture.

The opposite problem is the one that intercepts a target following an unknown path. Depending on the objective and constraints, this scenario is significantly more challenging. Robust solutions such as constant line-of-sight angle laws [13, 14] can guarantee interception at some point in time, but do not satisfy optimality. Modern solutions such as trajectory prediction schemes [12] aim to predict the target trajectory to then plan an intercept course.

In this dissertation, we argue that the problem depicted in Figure 1.4 does not fall in either category, thus requiring a custom solution. Although we know that the target's spatial constraints (ground vehicle) are determined by the roads it can travel on, we do not know its temporal trajectory nor its route choice with certainty. This, combined with the large scale of the problem and its risk constraints, makes the available tools unsuitable. Feedback control laws cannot account for these same constraints, and, at this scale, for this application, spatial trajectory prediction is computationally hard to execute and unnecessary given our geometric knowledge.

The solution developed in this dissertation revolves around challenges in scalability and embedded platforms, always focusing on computational resources and algorithmically provable safety. We adopt concepts of modern aviation, vision-based planning, and financial markets into this problem. At the core of the framework are risk aversion and risk estimation. Unlike generic stochastic planning algorithms, risk-aversion focuses on tailor-made measures that tackle specific aspects of a distribution. We use this feature to target the relationship between human behavior and flight resources. Machine Learning, Model Predictive Control (MPC), and probabilistic reasoning are applied in three different ways to provide a complete solution to the last-mile rendezvous problem in urban environments.

The first aspect of human behavior that imprints risk for motion planning is speed variation. Drivers inherently feel different levels of comfort concerning speed. An obvious example is on highways, where it is socially expected to travel above the speed limit, with different drivers exceeding the speed limit by different amounts. The implication here is obvious. If the driver does not match a prior motion model, when the UAS reaches the planned rendezvous

point it will either miss the driver or have to hover in place and wait for them. In either case, the UAS consumes more resources to either complete the rendezvous or return to the landing location. We approach this problem with Model Predictive Control (MPC), Gaussian Processes (GP), and probabilistic reasoning. The MPC module revolves around the computation of a Point-of-No-return location; similar to long-haul flights this is the point where a return to home is possible with the package. After this point, a rendezvous must be made so that the UAS is lighter and consumes less power on the way back. A GP model builds a behavioral predictor that tracks how far the driver is deviating from the traffic flow speed and the certainty of that prediction. Lastly, we propose probabilistic reasoning heuristics that are computationally efficient and guarantee that that the mission is always within the risk margins specified by the designer.

The second aspect of human behavior is the route choice. This is tackled at the local scale for a single driver, and the city-wide scale when choosing among multiple drivers. At the local scale, a modification of the speed variation algorithm will lose gradient information in the MPC module. We solve this with a cross-entropy [15] stochastic optimization scheme that finds optimal rendezvous locations by sampling from a GP using Deterministic Training Conditionals. We also expand probabilistic reasoning to include the choice of a target path when multiple options are possible. At the city-wide scale, we aim to find the driver least likely to deviate too far from a prescribed route. We achieve this by building a driver model and route dataset, on which a Deep Neural Network (DNN) is successfully trained. We show that adequate results are attained for smaller cities, whereas larger ones require a more sophisticated solution.

## 1.3 Related work

Several papers have considered risk measures in planning and handling uncertainties in an MPC framework, as summarized in [16, 17], and shown in [18, 19, 20, 21, 22, 23, 24]. In [19], the authors study uncertainty propagation to ensure chance constraints on a race car; results show that the algorithms can learn uncertainty in the dynamics, associating risk with the unknown dynamics, and plan so that the trajectories are safe. In [20], the authors provide

stability proofs for a linear MPC controller, which minimizes time-consistent risk metrics in a convex optimization form. These papers focus on operating in a constrained environment or under controlled assumptions to provide uniform guarantees. Our work's key difference is that we relinquish online risk constraint satisfaction to external heuristics, widening the solver's capabilities and flexibility at the cost of a more conservative solution. Apart from fundamental results in this field, such as [23], modern developments in [24] show that the increased computational capacity enables executing risk-minimization in real-time for a variety of systems. Few papers have been published concerning highly stochastic rendezvous problems. Most notably in [25] the authors compute optimal trajectories in refueling missions, but in their work most of the uncertainty is environmental and local, whereas we consider epistemic and large-scale uncertainties.

# CHAPTER 2

# METHODS

In this chapter, we describe and review the foundation of the methods used in this dissertation. First, and core to many of the algorithms discussed in our work, is Model Predictive Control (MPC). Model Predictive Control is used as a concurrent planner and controller for the UAS. Next, we present risk theory and risk measures, both of which are used in estimating the impact of human behavior on the MPC-generated plan. Lastly, we discuss risk-based probabilistic reasoning and some of the intuition behind modeling human behavior aspects with risk measures.

## 2.1 Model Predictive Control

Model Predictive Control is an optimization-based optimal control scheme. Also referred to as Receding Horizon Control (RHC), its main functionality relies on using the system dynamics to plan for some small time horizon in the future. When contrasted against classical control and modern model-based control MPC is different in many ways, of which the most notable are:

1. Online constraint management (as opposed to ad hoc).
2. Subset of modern model-based controllers.
3. High computational cost and hard to show guarantees.
4. Can be used as a dedicated or shared planner.

We highlight item 1. This is the property that sets MPC apart from other methods. On one end of model-based control, methods such as a Linear Quadratic Regulator (LQR) do not handle constraints, and usually, if the designer needs constraint satisfaction this is done ad-hoc with if-then-else heuristics. On the other end, Optimal Control Problems (OCP) aim to solve the

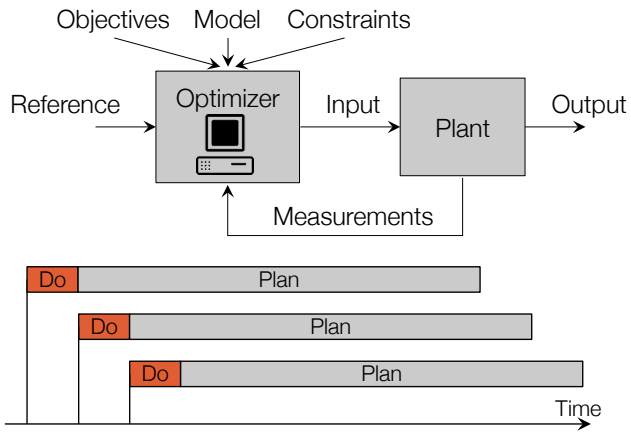



Figure 2.1: Structure of Receding Horizon Control, the basis of MPC [1].

entire problem in one execution. That is, given some dynamic system and constraints, what are the control inputs that achieve the objective. The downfall of this ambitious approach is that in the real world model uncertainties and external disturbances will make the system deviate from the nominal trajectory and the previously computed control inputs are no longer valid. MPC tackles this deficiency in an inelegant but incredibly effective way.

The MPC framework outline is depicted in Figure 2.1. In summa, MPC solves a smaller version of the full OCP repeatedly. To achieve this the main sacrifice is that we no longer consider the entire mission, instead just some time ahead of the present (usually in the order of a few seconds, referred to as *time horizon*). Other approximations are often used in constraints and dynamics to convexify the problem. The reason why MPC is effective revolves around how fast it runs. The idea is to re-compute the control inputs as often as possible to compensate for model uncertainties and external disturbances affecting the system. The speed in which we compute new control inputs is dictated by the complexity of the OCP inside it. Consider the following linear MPC example:

$$\begin{aligned} U^\star(k) = \arg\min_U \quad & J(x_{t+k}, u_{t+k}) && \text{cost function} \\ \text{s.t.} \quad & x_{t+k+1} = A_k x_{t+k} + B_k u_{t+k} && \text{system dynamics} \\ & g_{x,u} \leq 0 \\ & h_{x,u} = 0 \end{aligned} \tag{2.1}$$

where

$$J(x_{t+k}, u_{t+k}) = \|x_N - x_N^r\|_P + \sum_{k=n}^{N-1} \left\|x_{t+k} - x_{t+x}^r\right\|_Q + \sum_{k=m}^{M-1} \|u_{t+k}\|_R$$

and $\|\cdot\|_Q$, $\|\cdot\|_R$, and $\|\cdot\|_P$ are weighted norms for positive semi-definite diagonal matrices $Q$, $R$, and $P$. Equation (2.1) is a linear optimization problem with quadratic cost function and affine constraints. This means that it is convex and solvable with a Quadratic Program (QP). A QP is the fastest form of an MPC, and often it is possible to pre-compute control inputs in a region of the state-space for soaring control rates. In this dissertation we define convexity as in [26]. We say that $f : \mathbb{R}^n \mapsto \mathbb{R}$ is convex $\forall\ x_1, x_2 \in \mathbb{R}^n$ if and only if

$$f(\alpha x_1 + \beta x_2) \leq \alpha f(x_1) + \beta f(x_2)$$

such that $\alpha + \beta = 1$ and $\alpha,\ \beta \geq 0$. Additionally, $f : \mathbb{R}^m \mapsto \mathbb{R}^n$ is affine if there exists a linear function $l(x) : \mathbb{R}^m \mapsto \mathbb{R}^n$ and a vector $b \in \mathbb{R}^n$ such that

$$f(x) = l(x) + b\ \forall\ x \in \mathbb{R}^m.$$

The interesting problems rarely fall into this category with most being approximated down to a linear system to obtain a QP. A general formulation for MPC problems is

$$\begin{array}{rll} U^\star(k) = \arg\min_U & \sum_{k=0}^{N} L(x_{t+k}, u_{t+k}) & \text{cost function} \\ \text{s.t.} & x_{t+k+1} = f(x_{t+k}, u_{t+k}) & \text{system dynamics} \\ & g_{x,u} \leq 0 & \\ & h_{x,u} = 0 & \end{array} \tag{2.2}$$

where the cost function, system dynamics, and constraints are now generic. Modern MPC techniques use disturbance information to further improve tracking performance. In the next section we outline these methods.

**Example 2.1 (Linear Model Predictive Control)** *Consider the double integrator system*

$$\dot{p} = v,$$
$$\dot{v} = u,$$

*that we wish to control to the origin in minimum time. The initial conditions are arbitrary and the control effort $u \in \mathbb{R}$ is limited to the unit range $[-1, 1]$.*

The analytic solution to this problem is a tenet of optimal control theory. We provide the solution for completeness and refer to [27] for other examples and further reading in traditional optimal control. We can intuitively realize that the origin ($p = v = 0$) is reachable from any point in the state-space and that there is a minimum time in which this can be achieved (i.e. the boundedness of the control input makes the system not have an arbitrarily fast response). Let $x(t) = (p(t), v(t))$ be the state vector, and assume that $x(t_0)$ is given. Let $L$ be the running cost and set $L = 1$ as we seek a time-optimal control law. The Hamiltonian for the double integrator dynamics is

$$H(x, u) = \lambda_0 + \lambda_1 v + \lambda_2 u.$$

We aim to find an optimal control law $u^\star(t)$ with associated optimal costates $\lambda^\star = (\lambda_1^\star, \lambda_2^\star)$. Applying Pontryagin's Maximum Principle (PMP) [28] we first satisfy PMP(2) with the adjoint equation

$$\begin{bmatrix} \dot{\lambda}_1^\star \\ \dot{\lambda}_2^\star \end{bmatrix} = \begin{bmatrix} -H_p^\star \\ -H_v^\star \end{bmatrix} = \begin{bmatrix} 0 \\ -\dot{\lambda}_1^\star \end{bmatrix},$$

where $H^\star$ is the Hamiltonian evaluated with the optimal control law $u^\star$. The adjoint equation concludes that $\lambda_1^\star$ is constant and that $\lambda_2^\star$ is a linear combination of $\lambda_1$ and time: $\lambda_1^\star = a$, $\lambda_2^\star = -at + b$. Now we use PMP(1) to maximize the Hamiltonian and obtain

$$u^\star(t) = \operatorname{sign}(\lambda_2^\star(t)) = \begin{cases} 1, & \lambda_2^\star(t) > 0 \\ -1, & \lambda_2^\star(t) < 0. \end{cases} \tag{2.3}$$

As discussed in [27, Sec. 4.3] the case where $\lambda_2^\star(t) = 0$ only happens in isolated points in time (in this case, only once) and Equation (2.3) is well defined away from this point. With this, we conclude that the optimal control law $u^\star(t)$ is a bang-bang controller that can only attain a value of $\pm 1$ and switches sign only once. If we were controlling a real system, any uncertainties (unmodelled dynamics, disturbances, sensor noise) would make it impossible to apply this control law in an open loop. Thus we next solve this problem using MPC and verify that it converges to the behavior we just derived.

We now solve Example 2.1 with a receding horizon controller that implements the following linear optimization problem:

$$\begin{aligned} U^\star(k) = \arg\min_U \quad & \sum_{k=0}^{N} L(x_{t+k}, u_{t+k}) \\ \text{s.t.} \quad & p_{t+k+1} = p_{t+k} + v_{t+k} T_s \\ & v_{t+k+1} = v_{t+k} + u_{t+k} T_s \\ & |u_{t+k}| - 1 \leq 0, \end{aligned} \tag{2.4}$$

with $x = (p, v)$, $L(x, u) = p^2$, and $T_s$ being the discretization time. Notice that we do not minimize time directly. By minimizing the position error of every forward-looking prediction we minimize time nonetheless, although implicitly. In Figure 2.2 the MPC performance is depicted, where we verify that optimal single switching behavior was achieved. This ordinary problem presents an

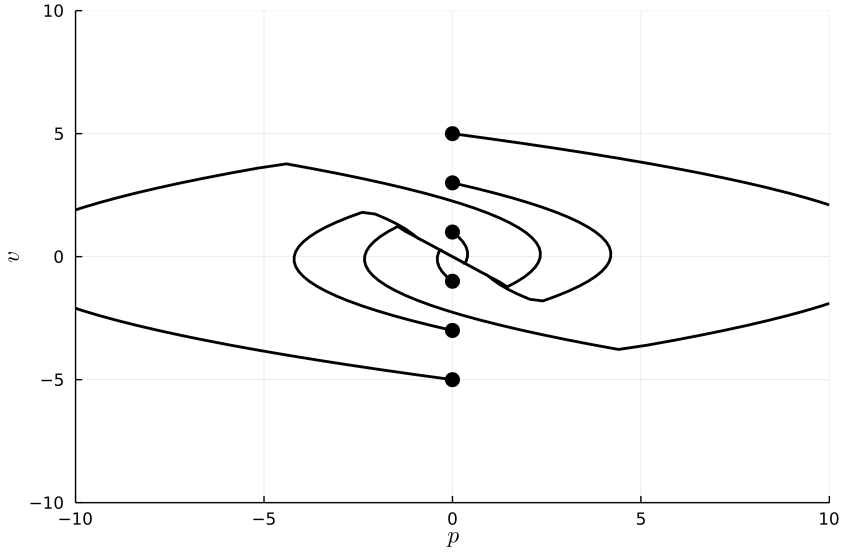



Figure 2.2: Linear MPC (2.4) from various starting positions. The MPC converges to the single switching behavior

interesting complication. We saw that using optimal control laws in *open-loop* is impractical. But what about other types of feedback control? There is a popular mentality in control systems practitioners (mostly in the industry) that feedback control can solve (albeit less optimally) most problems. Here we denominate feedback control the traditional technique of sampling sensor signals and calculating a control input with no account for future outcomes. We make this distinction because MPC has implicit feedback even though it is not referred to as a feedback controller. In the double integrator problem, we

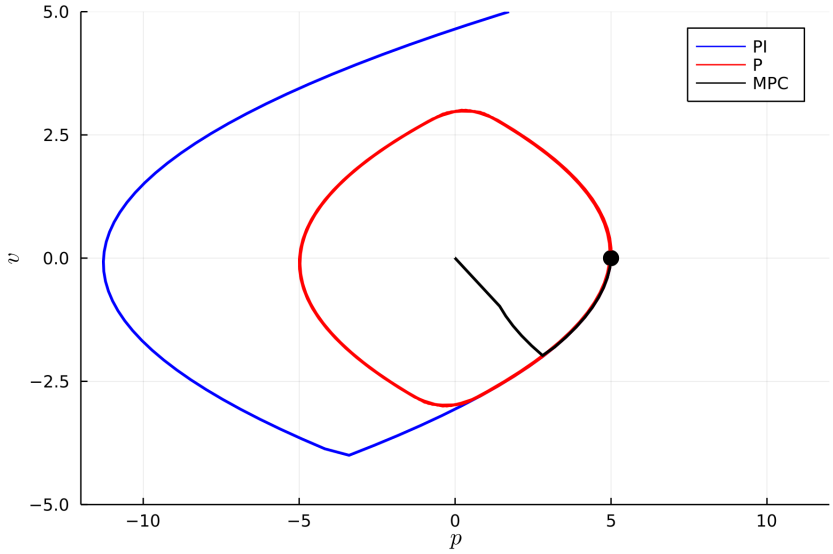



Figure 2.3: PI and P controllers fail this test.

find a counter-example to this mentality. And one that is valid and legitimate, unlike edge cases and fabricated examples that are rarely seen in real life. Consider a discrete proportional-integral controller (PI), with proportional gain $K_p = -1$, integral gain $K_i T_s = -1$, sampling time $T_s$, integrator state $I_k$, and error $e_k = p_k$. Next, let the integrator dynamics be $I_k = I_{k-1} + e_k K_i T s$, and control effort be $u_k = e_k K p + I_k$. This is the simplest formulation for a PI controller. It is clear from the dynamics in Example (2.1) and this formulation that for any initial condition $x(t_0)$ the system will oscillate for infinity with a proportional controller, and diverge for a PI controller. In the latter, the integrator state will wind up, building velocity towards the $p = 0$, once there it will wind up the other way since the error now switched signs, added with the system inertial it adds energy to the system. We show this behavior in Figure 2.3. The solution here is to use a proportional-derivative controller (PD), an approach often frowned upon due to noise and other practical aspects that hinder performance. MPC suffers non of these issues at the cost of computational complexity.

### 2.1.1 Stochastic Model Predictive Control

Stochastic MPC aims to use disturbance-related statistics to try and predict how the system will stochastically behave. Consider the following system

dynamics:

$$\dot{x}_t = f(x_t, u_t) + \omega(t)$$

with discrete-time realization

$$x_{t+k+1} = f(x_{t+k}, u_{t+k}) + \omega(t+k)$$

where $\omega(t)$ is a random variable acting on the system at time $t+k$. All of the states in the system are now probabilistic. The most basic form of Stochastic MPC (SMPC) minimizes the expectation:

$$\begin{aligned} U^\star(k) = \arg\min_U \quad & \mathbb{E}\left[\sum_{k=0}^{N} L(x_{t+k}, u_{t+k})\right] \\ \text{s.t.} \quad & x_{t+k+1} = f(x_{t+k}, u_{t+k}) + \omega(t) \\ & g_{x,u} \leq 0 \\ & h_{x,u} = 0. \end{aligned} \tag{2.5}$$

This formulation, albeit simple, solves many complex problems. The main deficiency of this approach is robustness. Minimizing the expectation of the cost function brings the solution closer to optimality. It does not, however, guarantee future constraint satisfaction. For safety-critical systems it makes standard SMPC unacceptable. Engineers faced with this fact developed many approximations such as stochastic slack variables [26]. A later, more popular, approach is to use Robust SMPC. Robustness is added to (2.5) with a min max optimization scheme. Such scheme aims to minimize some quantities and maximize others. In this case, we aim to minimize cost given that the negative effect of the random variable is maximized, i.e. SMPC plans for the expected system behavior, while Robust SMPC plans for the worst possible system behavior:

$$\begin{aligned} U^\star(k) = \arg\min_U \max_\omega \quad & \mathbb{E}\left[\sum_{k=0}^{N} L(x_{t+k}, u_{t+k})\right] \\ \text{s.t.} \quad & x_{t+k+1} = f(x_{t+k}, u_{t+k}) + \omega(t) \\ & g_{x,u} \leq 0 \\ & h_{x,u} = 0. \end{aligned}$$

Robust SMPC makes the obvious sacrifice of optimality to attempt safety guarantees. Later in this chapter we describe risk-averse MPC, where it seeks an optimal middle-ground between SMPC and Robust SMPC.

### 2.1.2 Sampling-Based Model Predictive Control

Not to be confused with stochastic gradient descent, Sampling-Based MPC aims to find the solution to (2.2) by sampling control inputs and computing their cost. Then, the distribution from which the control inputs are drawn from is updated to match the distribution of the fittest samples. Notable examples of this approach are Model Predictive Path Integral (MPPI) [29] and Cross-Entropy Motion Planning [15]. There are many ways of solving the OCP this way; we highlight a generic approach for a scalar system. Consider a scalar system with dynamics $x_{t+k+1} = f(x_{t+k}, u_{t+k})$ and time horizon $N \cdot T_s$ where $T_s$ is the sampling period and $N$ the number of look-ahead samples. We wish to find a sequence of inputs $U = u_0, \ldots, u_N$ that minimizes a cost function $\sum_{k=0}^{N} L(x_{t+k}, u_{t+k})$ and satisfies equality constraints $h_{x,u} = 0$ and inequality constraints $g_{x,u} \leq 0$. In this example we use Gaussian sampling distributions. Let $\mathcal{A} = \mathcal{N}(\mu, \Sigma)$ be a $N$-dimensional Gaussian distribution with mean parameter vector $\mu$ and diagonal covariance matrix $\Sigma$. Algorithm 1 outlines the method. Function Sample$(\mathcal{A}, N_s)$ returns a set $\mathcal{S} \in \mathbb{R}^N \times \mathbb{R}^{N_s}$ of $N$-dimensional samples from $\mathcal{A}$, function Rank$(\mathcal{S}, L, N_e)$ returns a similar set $\mathcal{S}_e \in \mathbb{R}^N \times \mathbb{R}^{N_e}$ containing the $N_e$-best $N$-dimensional samples from $\mathcal{S}$ along and the top sample $U^\star$. Note that in in this operation we rank the samples according to the cost function $L$. The last step updates the means and variances of $\mathcal{A}$ with the statistics of the elite sample set $S_e$.

**Algorithm 1:** Sampling Based MPC

**Parameters:** $f$: transition function
$L$: cost function, $N_s$: sample size, $N_e$: elite sample group size
**while** *not done* **do**
  $\mathcal{S} \leftarrow$ Sample$(\mathcal{A}, N_s)$
  $U^\star,\ \mathcal{S}_e \leftarrow$ Rank$(\mathcal{S}, L, N_e)$
  Send Control Input $u_0^\star$ to system
  $\mu,\ \Sigma \leftarrow$ UpdateParameter$(\mathcal{S}_e)$
**end**

## 2.2 Risk Measures

In this section we discuss risk theory and risk measurement, later moving to a recent transformation from a use purely in the financial market and economy

theory to motion planning in robotics. So far in this chapter, we introduced variations of MPC that account for stochastic behavior. Mainly, we focused on robust and stochastic MPC. The former targets safety against some undesirable random disturbance; the latter aims to attain optimality (whichever form that might take depends on the cost function). Now, we introduce the tools necessary for a new type of MPC. Risk-averse MPC includes special types of measures in its cost; hence it can be considered a special case of the others. The foundation for this approach lie in financial markets, but in recent years some optimal control practitioners transitioned these techniques to control, path planning, and policy optimization. It suffices to discuss risk measures, as it will become clear moving forward that we simply include tailored risk measures into our MPC formulations, transforming them from robust, stochastic, or sampling-based MPC, into Risk-Averse MPC.

Guégan's [21] taxonomy of risk measures divides them into two groups: measures of dispersion and downside risk measures. We shall follow this taxonomy in this exposition, and focus on downside risk measures (also referred to as safety risk measures). Dispersion measures of interest to us are mean and variance, which is trivial. This field of research was sparked by economist Harry Markowitz [30] in 1952. He showed that more important than maximizing discounted returns is minimizing the correct measure of risk. Investors are naturally risk-averse, which is Markowitz's motivation.

Let $(\Omega, \mathcal{F}, P)$ be a probability space with sample space defined by $\Omega$, event space defined by the $\sigma$-algebra $\mathcal{F}$, and probability measure $P$ on $\mathcal{F}$. Let $X$ be a continuous random variable on $(\Omega, \mathcal{F}, P)$ with probability density function $f(x)$ and cumulative density function $F(x)$. A risk measure is a function $\rho : X \mapsto \mathbb{R} \bigcup \{-\infty, \infty\}$. A risk measure allows the expression of the riskiness of an entire motion plan with a single real number. Using a proper risk measure means that a riskier plan yields a higher measure of that risk. The idea of a proper risk measure was unclear until Artzner defined the concept of coherent risk measures [22]. A coherent risk measure has the following properties, where we highlight investment intuition behind them:

- Normalized: $\rho(0) = 0$, *the risk of holding no assets is zero.*
- Translational invariance: $\rho(X + c) = \rho(X) - c$, *adding a certain gain decreases risk by the same amount.*
- Sub-additivity: $\rho(X_1 + X_2) \leq \rho(X_1) + \rho(X_2)$, *diversification decreases*

*risk.*

- Monotonicity: $X_1 \leq X_2 \implies \rho(X_1) \leq \rho(X_2)$, *higher losses mean higher risk.*

- Positive Homogeneity: $\rho(\lambda X) = \lambda\rho(X),\ \lambda \geq 0$, *risk of a position is proportional to its size.*

Like many authors in this field, Artzner was an investor. His goal with the definition of coherent measures of risk was to categorize a family of risk measures that guarantee the investors' best interest. In this dissertation, we concern ourselves with risk measures that guarantee the planner's best interest. Luckily for us, there is a wide intersection between the two, which is why many roboticists seek coherent risk measures such as CVaR to achieve risk-free motion planning. Value-at-Risk (VaR), is not a coherent risk measure, but the (arguably) most popular one. We define it as

$$\mathrm{VaR}_{1-\alpha}(X) = \inf_{t\in\mathbb{R}}\{t : \mathbb{P}(X \leq t) \geq 1-\alpha\}.$$

Intuitively, VaR is the smallest event of $X$ that occurs with probability $1-\alpha$. It provides a clear and concise understanding of its use and function in an algorithm. Although VaR lacks sub-additivity, it is usually not problematic for use in robotics and motion planning. The original idea behind sub-additivity is to guarantee proper risk assessment during diversification moves such as mergers and hedging. For motion planning purposes this is not a concern. However, VaR has one significant downfall when it comes to motion planning: it does not quantify losses beyond the $1-\alpha$ percentile. One can craft a distribution with an arbitrarily high loss event and arbitrarily low probability. Although it might seem like a stretch, safety-critical robotic systems cannot suffer a catastrophic failure that endangers human lives.

The non-coherent aspect of VaR prompted economists to create new measures of risk. A total of four years passed before risk professionals found a coherent alternative to VaR since Artzner introduced it in 1997. One such measure aimed to answer the question of *"how bad is bad?"* by taking the expectation of all values of $X$ that are higher than $\mathrm{VaR}(X)$ for some quantile:

$$\mathrm{CVaR}_{1-\alpha}(X) = \frac{1}{\alpha}\int_0^{\alpha} \mathrm{VaR}_{1-t}(X)\,\mathrm{d}t.$$

In this dissertation, we focus on CVaR. We wish to quantify the cost of motion planning should a human cause a worst-case scenario, and guarantee that we meet some threshold of safety. One differential for using risk measures for motion planning is that computational resources are limited, and many modern planners rely on gradient information, which may be unavailable. Economists do not suffer from such downfalls because their optimization problems can be run on large-scale computers and with no hard time limits. Should the solution be unfeasible investors might suffer financial loss, but no physical harm occurs. In our case, these are critical problems that we address in multiple ways. The rest of this chapter outlines some of the problems as it relates to CVaR-based optimization, and provide specific details for our approach in the following chapters.

### 2.2.1 Risk-Averse Optimization

In this section we outline optimization problems involving notions of risk, we follow a similar line of reasoning to [31, 32]. Suppose that a stochastic system with output variable $Z$ is a real-valued random variable. For some decision variable $x \in \mathbb{R}^n$ let

$$Z(\omega) = f(x, \omega),\ \omega \in \Omega,$$

where $f : \mathbb{R}^n \times \Omega \mapsto \mathbb{R}$. Note that in the context of motion planning and control we manipulate $x$ through some control input. In this scenario, $Z$ is an adverse quantity we wish to minimize. We can pose this as an optimization problem:

$$\begin{aligned} \min_x \quad & \mathbb{E}_P\left[f(x, \omega)\right] \\ \text{s.t.} \quad & x \in \mathcal{X} \end{aligned}$$

where $\mathcal{X}$ is the set of admissible values of $x$ and $P$ is a distribution on $(\Omega, \mathcal{F}, P)$. One cannot simply write a program like this into computer language. Before any attempt of a numerical solution, one either studies the propagation of $\omega$ through $f$ to obtain an analytical expression, or approximations and assumptions are made to relax the problem. This is due to a flawed assumption that $P$ is known, and that it is impossible to directly optimize the expectation of an arbitrary random variable. The traditional way of solving the issue of

arbitrary distributions is to turn the problem into a min max formulation. Suppose there is some class $\mathcal{P}$ of admissible distributions for $P$, then an equivalent formulation is

$$\begin{aligned} \min_x \quad & \sup_{P\in\mathcal{P}} \mathbb{E}_P\left[f(x,\omega)\right] \\ \text{s.t.} \quad & x \in \mathcal{X}. \end{aligned}$$

One can immediately draw parallels to the rationale of MPC professionals in Section 2.1. In many cases researchers consider a dynamical system with additive Gaussian noise, which we use as an example of a poor approach:

$$x_{k+1} = f(x_k, u_k) + \omega_k,\ \omega \sim \mathcal{N}(\mu, \sigma).$$

Under these conditions (and often many others) it is sufficient to take statistics from $\omega$ as additives to the system, yielding a far simpler OCP. For example, consider a constraint of the form $x_k \in \mathcal{X}$ with $f(x_k, u_k) = x_k + u_k$, $\omega \sim \mathcal{N}(0,1)$, and $x_k \geq 0$. We can approximate the constraint with $1-\sigma$ certainty to $x_k - 1 \in \mathcal{X}$. As previously discussed, however, this approach does not translate naturally to risk-aversion instincts.

Consider now that we wish to use VaR as a constraint on the system. That is, we wish that $x$ remain below some value with some probability, in other words, we wish to impose the following constraint on the system:

$$\mathbb{P}_P[Z \leq \bar{z}] \leq 1 - \bar{p}.$$

These constraints are usually referred to chance constraints. It is also common to classify constraints on CVaR as chance constraints, although we oppose that classification in this dissertation. In general, one can choose to minimize risk with

$$\begin{aligned} \min_x \quad & \rho\left[f(x,\omega)\right] \\ \text{s.t.} \quad & x \in \mathcal{X}, \end{aligned}$$

or to constrain risk:

$$\begin{aligned} \min_x \quad & \mathbb{E}_P\left[f(x,\omega)\right] \\ \text{s.t.} \quad & x \in \mathcal{X} \\ & \rho(f(x,\omega)) \leq c. \end{aligned}$$

A combination of both is possible but ill-advised. If using the same risk measure as an objective function and a constraint, one will have more luck utilizing slack variables [26]. If using different measures, likely, the units do not agree, and the designer will find themselves with a meaningless constraint and/or objective.

We have yet to answer how to solve these problems. Risk-averse optimization is usually solved similarly to generic stochastic optimization problems. We must either rely on approximations and assumptions, such as Sample Average Approximation [33]:

$$\mathbb{E}\left[f(x,\omega)\right] \approx \frac{1}{N}\sum_{i=1}^{N} f(x,\omega_i). \tag{2.6}$$

In this dissertation, we use approximations in Chapter 4, and sampling-based methods in Chapters 5 and 6 with similar methods to the one described in Subsection 2.1.2. Sampling based approaches are trivial in a way. We get around the fact that we do not have knowledge of the distribution of interest by sampling and propagating, then selecting the best outcomes. If we do have knowledge of the distribution we can directly optimize certain risk measures. For example, consider $\mathcal{U}(a,b)$ as the uniform distribution over the range $(a,b)$, then

$$\mathrm{VaR}_{1-\alpha}(X) = a + (1-\alpha)(b-a),$$

and

$$\mathrm{CVaR}_{1-\alpha}(X) = a + \frac{2-\alpha}{2}(b-a).$$

A closed form solution also exists for the Gaussian distribution $\mathcal{N}(\mu,\sigma^2)$ with

$$\mathrm{VaR}_{1-\alpha}(X) = \mu + z_\alpha\sigma,$$

and

$$\mathrm{CVaR}_{1-\alpha}(X) = \mu + \frac{\phi(z_\alpha)}{\alpha}\sigma,$$

where $\phi(\cdot)$ is the density function of the standard normal distribution, and $z_\alpha$ its upper $\alpha$-percentile.

One can quickly craft fast optimization problems with these closed forms. We use the example in [34] for portfolio optimization using CVaR. Consider the allocation vector $x \in \mathbb{R}^n$ across $n$ assets, $\beta = \text{CVaR}_{1-\eta}$ for some $\eta$, a loss threshold $u \in \mathbb{R}^+$, and a vector of returns $p \sim \mathcal{N}(\bar{p}, \Sigma)$. We maximize returns and minimize CVaR with the following optimization problem:

$$\begin{aligned} \min_{x,\beta} \quad & \mathbb{E}\left[-p^\intercal x\right] \\ \text{s.t.} \quad & \beta + \tfrac{1}{1-\eta}\mathbb{E}\left[-p^\intercal x - \beta\right]_+ \leq u \\ & \mathbb{I}^\intercal x = 1 \\ & x \succeq 0, \end{aligned}$$

where expectation is taken with (2.6) and $z_+ = \max\{0, z\}$. This linear optimization problem is easily implementable in a number of modern toolboxes, including MATLAB, CVX [35], and JuMP [36], the latter which we use throughout this thesis.

### 2.2.2 Risk-based Probabilistic Reasoning

In this subsection, we provide a high-level discussion of reasoning (or automating) processes that use these measures. We begin with a material example. Suppose we are presented with a bet represented by the random process $f$. We start with $f_0 = 100$ currency units and a fair coin that will be flipped being represented by the random variable $c$. If it is heads, we win 125; otherwise, we lose 100 and have no money left. A quick computation of expected value yields

$$\mathbb{E}[f_1] = 125 \cdot \mathbb{P}[c \text{ is heads}] - f_0 \cdot \mathbb{P}[c \text{ is tails}] = 12.50.$$

Presented with this value it is immediate for a pragmatic mathematician to take the bet, expecting that they will profit 12.50. In actuality very few people will take it; one can estimate around 20% to 40% depending on what form this question takes. We subconsciously reach this conclusion because humans are naturally risk-averse. We reason that the loss of 100 is more devastating than the profit of 125. We can model the diminishing marginal rate of substitution aspect of this bet in risk management terms by taking the utility function [21]

$$u(f) = \sqrt{f},$$

so that our initial utility is $u_0 = \sqrt{f_0} = 10$. Then we can compute the expected utility of this bet:

$$\mathbb{E}[u(f_1)] = u_1 = \sqrt{125 + u_0} \cdot \mathbb{P}[c \text{ is heads}] - \sqrt{f_0} \cdot \mathbb{P}[c \text{ is tails}] = 7.50 < u_0.$$

This result means that if we don't weigh cost linearly and instead use a properly constructed risk measure, we come to realize that the risk of taking this bet is higher (or the weight of probable loss is high) than what we are comfortable with. This is exactly how we reason in a motion planning context using risk-averse costs.

In motion planning, and specifically robots operating around humans, risk is a combination of failing the goal and of doing so while harming a human. The next question is the choice. Traditionally in MPC applications, there are no discrete decisions. That is, there is not a 'yes' or 'no' question that the solver must answer. The MPC will try to minimize the risk (or maximize utility) but the designer is responsible for building heuristics that work with the available information. This is perhaps the unfortunate aspect of choosing rigor over black-box approaches often found in artificial intelligence. Not only we must spend time and resources building these heuristics, but the risk thresholds are met only under the condition that the heuristics are accurate. This is a condition that no algorithm will ever be able to inform and is left for humans to ascertain. As far as this dissertation is concerned, we chose to use simple threshold-based heuristics (if risk is higher than some number, do not proceed); this way our only concern is how risky we are willing to be. The downside is that we must interpret the meaning of risk and choose this constant appropriately.

# CHAPTER 3

# ROBUST AERIAL MANIPULATION

In this chapter, we present all aspects of our efforts into aerial manipulation for drop-off and pickup of packages. First, we discuss mechanical design, where we point out ways in which one needs to optimize to guarantee a safe and robust flight. Using that mechanical design we expose Lagrangian equations used in motion control. Finally, we discuss the control architecture comprised of baseline, feedforward, and adaptive controllers. We finalize the chapter with optimal trajectory generation for such devices and results for the approach.

## 3.1 Manipulator Design

Manipulators, in general, are engineered on a task-specific basis, since each family of robotic arms provides the designer with a different set of advantages over the others. Traditional approaches utilizing serial manipulators aim to explore the better reach, ease of dynamics modeling, and use of readily available literature. In this parallel design, we consider the real needs an aerial manipulation system has along with the known ground-based results to elevate what is currently expected from this family of systems.

The prototype in [9] is a serial manipulator with 2-DoF; however, having only 2-DoF requires the UAS to act as the third actuator for complete 3-DoF reachability and proper positioning of the end-effector. Since it is logically desirable for the end-effector to have a higher bandwidth than the vehicle in all of its axes of motion, a 3-DoF manipulator is needed. This immediately causes design conflicts for a serial type as at least one of the actuators would contribute negatively to the stability of the system by having to be placed in a moving part of the arm. The parallel-delta manipulator (often referred to as Clavel's Manipulator [37]) however, manages to place all three actuators at its base and maintain three degrees of freedom. Since the actuators are often heavy, designs with static manipulators need less torque for trajectory

tracking, in general, which in this application reduces the torques induced on the UAS. The move from serial to parallel changes several aspects of the design, shown in Table 3.1, where improvements of interest to this work over the serial type are highlighted in green, while declines in red.

| **Feature** | **Serial** | **Parallel** |
|---|---|---|
| Workspace | Large | Average and complex |
| Workspace/robot ratio | High | Low |
| Forward kinematics | Easy | Very difficult |
| Inverse kinematics | Difficult | Easy |
| Position error | Accumulates | Averages |
| Force error | Averages | Accumulates |
| Maximum force | Single actuator | Sum of actuators |
| Stiffness | Low | High |
| Dynamics characteristics | Poor | Excellent |
| Solving Dynamics | Simple | Complex |
| Inertia | Large | Small |
| Payload/weight ratio | Low | High |
| Speed and acceleration | Low | High |
| Accuracy | Low | High |
| Component uniformity | Low | High |
| Calibration | Simple | Complex |

Table 3.1: Serial and Parallel Topology Comparison

This approach provides us several advantages such as improved stiffness, better accuracy, better manufacturing error rejection, and others at the cost of a reduced workspace and complicated dynamics. The two major downsides are the small workspace and challenging dynamics since calibration can be done mid-flight utilizing modern positioning systems and a pseudo-inverse-Jacobian method in an outer loop. To increase the workspace, exceptional stiffness is explored. The stiffness of these devices is so abundant that any load heavy enough to cause deflection in the links would be far too heavy for the UAS to lift. By curving the ends of the secondary links with Bézier curves to guarantee continuous lines (reducing mechanical stress), the design can achieve nearly 180 degrees of motion while maintaining enough stiffness to lift the maximum allowable payload. Figure 3.1 exemplifies the folding capabilities of these joints, and Fig. 3.2 compares it to the industry-standard approach for ground-based delta-type parallel manipulators. The actuators of choice are the Dynamixel RX-24F [38], which communicates via an RS-485

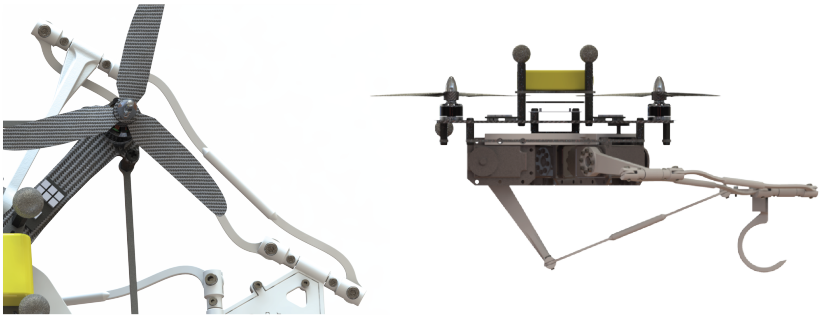
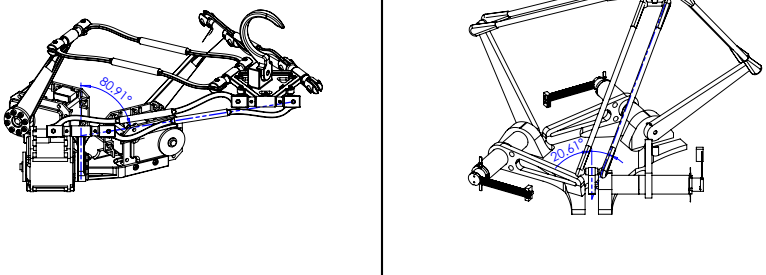


Figure 3.1: Folding property of the arm, top and right views.




(a) Bézier joint design

(b) Spherical joint design.

Figure 3.2: Comparison of joint design angular travel: (a) Bézier design attains $\sim \pm 81°$ and (b) spherical design attains $\sim \pm 21°$.

bus directly to the Crazyflie 2.0 board.

By utilizing this novel design the device nearly attains the reach of the serial manipulator variant while increasing accuracy, precision, stiffness, bandwidth, and adding one extra degree of freedom. The only true downside is the increased weight, although future prototypes are expected to utilize compact, better-optimized actuators and links which will undoubtedly improve both load capacity and flight performance. An additional property to bear in mind is that the end-effector is now always parallel to the UAS, which translates to a loss in the ability to orient the gripper as desired. This, however, is often unnecessary. Having a 4-DoF (three from the manipulator plus one from the UAS's yaw) platform eliminates the need for a more articulate end-effector, and in the rare occasions where it is required, the modular tool-base allows for interchangeable, mission-specific grippers to be installed.

## 3.2 Manipulator Dynamics

In robotic manipulator control literature, Computed Torque Control relies on calculating the expected torque generated by the dynamics of the manipulator, and feeding it forward to the actuators via a torque controller, augmenting the position PID controller. Since the Delta-type robot is widely used in the industry, these equations are well known. For this application, however, some changes have to be made so that the result reflects the torque generated at the actuators fixture to the base, instead of on its rotor. We mainly follow results of [39] which are modified as follows: For a given end-effector trajectory $\mathbf{p} = (p, \dot{p}, \ddot{p})$ and actuator topology $\phi = (\phi_1, \phi_2, \phi_3) = (-\pi/2, \pi/6, 5\pi/6)$ the three torques are computed as

$$\begin{aligned}
\tau_1 &= (I_1 + m_2 l_1^2)\ddot{\theta}_1 - (m_1 l_{1c} + m_2 l_1) g_c \cos(\theta_1) - \\
&\quad 2 l_1 \lambda_1 [(p_1 \cos(\phi_1) + p_2 \sin(\phi_1) + b - a) \sin(\theta_1) - \\
&\quad p_3 \cos(\theta_1)], \\
\tau_2 &= (I_1 + m_2 l_1^2)\ddot{\theta}_2 - (m_1 l_{1c} + m_2 l_1) g_c \cos(\theta_2) - \\
&\quad 2 l_1 \lambda_2 [(p_1 \cos(\phi_2) + p_2 \sin(\phi_2) + b - a) \sin(\theta_2) - \\
&\quad p_3 \cos(\theta_2)], \\
\tau_3 &= (I_1 + m_2 l_1^2)\ddot{\theta}_3 - (m_1 l_{1c} + m_2 l_1) g_c \cos(\theta_3) - \\
&\quad 2 l_1 \lambda_3 [(p_1 \cos(\phi_3) + p_2 \sin(\phi_3) + b - a) \sin(\theta_3) - \\
&\quad p_3 \cos(\theta_3)],
\end{aligned} \tag{3.1}$$

where the Lagrangian multipliers $\lambda_i$ are obtained by solving the system

$$\begin{aligned}
&2 \sum_{i=1}^{3} \lambda_i (p_1 + b \cos(\phi_i) - a \cos(\phi_i) \\
&\qquad - l_1 \cos(\phi_i) \cos(\theta_i)) = (m_p + 3m_2)\ddot{p}_1, \\
&2 \sum_{i=1}^{3} \lambda_i (p_2 + b \sin(\phi_i) - a \sin(\phi_i) \\
&\qquad - l_1 \sin(\phi_i) \cos(\theta_i)) = (m_p + 3m_2)\ddot{p}_2, \\
&2 \sum_{i=1}^{3} \lambda_i (p_3 - l_1 \sin(\theta_i) = (m_p + 3m_2)(\ddot{p}_3 - g_c)
\end{aligned} \tag{3.2}$$

The explicit symbolic solution to the system above exists and is well-defined but also extraordinarily long and, thus, omitted. The actuator topology $\phi$ represents the position of each actuator and the joint-space trajectory $(\theta_i, \ddot{\theta}_i)$ is obtained by applying inverse kinematics and passing it through the high fidelity actuator model. The remaining parameters are described in Table 3.2.

| | |
|---|---|
| $a$ | Radius of the fixed base (center of the base to actuator) |
| $b$ | Radius of the moving platform (center of the platform to joint) |
| $l_i$ | Length of link $i$ |
| $m_1$ | Mass of link 1 |
| $m_2$ | Half of mass of link 2 |
| $m_p$ | Mass of the moving platform |
| $l_{1c}$ | Length of the center of mass of link 1 |
| $I_1$ | Moment of inertia of link 1 |
| $g_c$ | Gravity component orthogonal to the moving platform |

Table 3.2: Kinematic parameters and mass properties of the Delta manipulator.

The transformation to the $x-y$ plane of the UAS is then done by projecting each actuator torque into the $x-z$ and $y-z$ planes following the topology $\phi$. The final torque mapping is

$$\hat{\tau} = \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix} = \begin{bmatrix} \sum_1^3 \tau_{i_x} \\ \sum_1^3 \tau_{i_y} \end{bmatrix} \tag{3.3}$$

where the $\hat{\cdot}$ operator denotes estimation and $\hat{\tau}$ is the estimated torque vector in the $x-y$ plane. No torque exists on the $z$ axis by construction.

## 3.3 Controller Design

In this section, the design of control laws is discussed. This process is using a realistic model of the system, where all signals flow as they would on the UAS. Throughout the rest of this dissertation, the generalized coordinates of the system are $Q(t) = [\varphi_1(t), \varphi_2(t), \theta_1(t), \theta_2(t), \theta_3(t)]$, where $[\varphi_1(t), \varphi_2(t)] = \varphi(t)$ are the pitch and roll angles respectively and $[\theta_1(t), \theta_2(t), \theta_3(t)] = \theta(t)$ are the manipulator actuators angular position. The subscript $c$ indicates a command

(i.e. $\theta_{1_c}$ indicates the angular position command for the first manipulator actuator). We first state the problem formulation:

**Problem Formulation 3.1** *(Stabilizing controller for an aerial manipulator)*
*Given a UAS equipped with a Delta-type parallel manipulator and an end-effector trajectory* $\mathbf{p}$*, find a control signal* $\mathbf{u}$ *that stabilizes the vehicle.*

*Assumptions:*

1. *The trajectory* $\mathbf{p}$ *remains strictly inside the workspace.*
2. *Throughout the trajectory* $\mathbf{p}$ *the payload never exceeds the allowable maximum.*
3. *All manipulator actuators don't saturate in terms of torque, speed or acceleration.*

The assumptions here are mainly in place to not allow non-feasible trajectories. Assumption 1 prevents the manipulator from reaching singularity positions; Assumption 2 guarantees that at every point in the trajectory stable flight is attainable; Assumption 3 restricts the trajectories to the ones that the manipulator can perform.

## 3.4 General Control Scheme

Fig. 3.3 shows the general control scheme of the system. The commanded joint-space trajectory $\theta_c$ and commanded pitch and roll attitudes $\varphi_c$ are fed to the system. The manipulator dynamics generate the disturbance $\tau_m$ while the Computed Torque Feedforward (CTF) algorithm estimates this disturbance and generates a control signal consistent with the vehicle dynamics. The baseline controller is a cascaded PI/PID (rate PI and attitude PID) controller which is augmented with the $\mathcal{L}_1$ adaptive controller.

### 3.4.1 Baseline Controller

The baseline controller follows a cascaded PI/PID attitude structure as shown below:

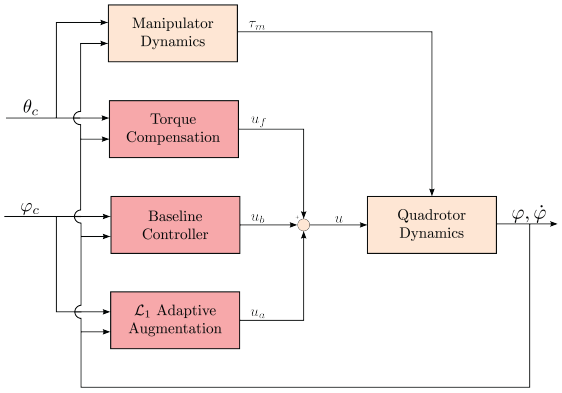



Figure 3.3: General control scheme

$$u_{b_i}(t) = k_{P1}\left(\dot{\varphi}_{i_c}(t) - \dot{\varphi}_i(t)\right) + k_{I1}\int_0^t \left(\dot{\varphi}_{i_c}(t) - \dot{\varphi}_i(t)\right) d\tau$$

where

$$\dot{\varphi}_{i_c}(t) = k_{P2}\left(\varphi_{i_c}(t) - \varphi_i(t)\right) + k_{I2}\int_0^t \left(\varphi_{i_c}(t) - \varphi_i(t)\right) d\tau + k_D\dot{\varphi}_i(t)$$

and $k_{P1}$, $k_{I1}$, $k_{P2}$, $k_{I2}$, $k_D \in \mathbb{R}$ are manually tuned control gains. Here, $u_b = [u_{b_1}\ u_{b_2}]^\top$ is the vector of the baseline pitch and roll control signals respectively.

### 3.4.2 $\mathcal{L}_1$ Adaptive Control Augmentation

The aerial manipulator can pick up a large variety of objects, varying in mass, inertia, density, shape, etc. With a flight controller augmented only with the feedforward torque compensation of the manipulator, there is no way to account for the induced torques and forces from these uncertain payloads. To reject uncertainties introduced by unknown payloads, the $\mathcal{L}_1$ adaptive control structure is chosen as a robust augmentation.

An $\mathcal{L}_1$ adaptive control augmentation with piecewise constant adaptation law from [40] is considered. Here we implement this structure on the pitch and roll axis separately; this is possible due to the decoupling between the $x$ and $y$ torque axis of the manipulator. Let $x_{I1}(t)$ and $x_{I2}(t)$ denote the states of the

integrators in the rate and attitude loop of the baseline controller respectively. Then, the rotational equation of motion of the UAS with the $\mathcal{L}_1$ augmentation can be expressed as

$$\begin{aligned} \dot{x}(t) &= A_m x(t) + B_r r(t) + B_m \left(u_a(t) + f_1(t,x)\right) + B_{um} f_2(t,x), \\ y(t) &= C_m x(t), \\ x(0) &= x_0, \end{aligned} \tag{3.4}$$

where $x(t) = [\varphi_1(t), \dot{\varphi}_1(t), x_{I1}(t), x_{I2}(t)]^\top$ is the vector of the system states, $r(t) = \varphi_{i_c}(t)$ is the reference attitude command, $f_1(t,x) : \mathbb{R} \times \mathbb{R}^4 \to \mathbb{R}$ is a nonlinear function containing information on the residual of the feedforward torque compensation for the disturbance torque from the manipulator and the matched uncertainty, $f_2(t,x) : \mathbb{R} \times \mathbb{R}^4 \to \mathbb{R}^3$ is a nonlinear function representing additional modeling uncertainty, $A_m \in \mathbb{R}^{4\times4}$ is a known Hurwitz matrix defining the desired system dynamics, $B_r \in \mathbb{R}^{4\times1}$ is the known command matrix, $B_m \in \mathbb{R}^{4\times1}$ is the known control matrix, $C_m \in \mathbb{R}^{1\times4}$ is a known full-rank constant matrix, and $B_{um} \in \mathbb{R}^{4\times3}$ is a matrix such that $B_m^\top B_{um} = 0$ and $[B_m \ B_{um}]$ has full rank. The product $B_{um} f_2(t,x)$ represents the unmatched uncertainty. The matrices $A_m$, $B_r$, and $B_m$ can be written as

$$A_m = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_{P1}k_{P2}}{J_0} & \frac{k_{P1}k_D}{J_0} & \frac{k_{I1}}{J_0} & \frac{k_{P1}k_{I2}}{J_0} \\ -k_{P2} & k_D - 1 & 0 & k_{I2} \\ -1 & 0 & 0 & 0 \end{bmatrix},$$

$$B_r = \begin{bmatrix} 0 \\ \frac{k_{P1}k_{P2}}{J_0} \\ k_{P2} \\ 1 \end{bmatrix}, \quad B_m = \begin{bmatrix} 0 \\ \frac{1}{J_0} \\ 0 \\ 0 \end{bmatrix}.$$

For the system given in (3.4), the elements of the $\mathcal{L}_1$ adaptive controller are given below.

#### 3.4.2.1 State Predictor

Taking the same structure as the system in (3.4), the state predictor is given by

$$\begin{aligned}\dot{\hat{x}}(t) &= A_m\hat{x}(t) + B_r r(t) + B_m\left(u_a(t) + \hat{\sigma}_1(t)\right) + B_{um}\hat{\sigma}_2(t),\\ \hat{x}(0) &= x_0,\end{aligned}$$

where $\hat{x}(t)$ is the predictor state, $\hat{\sigma}_1(t) \in \mathbb{R}$ and $\hat{\sigma}_2(t) \in \mathbb{R}^3$ are the estimates of the nonlinear functions $f_1(\cdot)$ and $f_2(\cdot)$ respectively.

#### 3.4.2.2 Adaptation Law

Given an adaptation rate $T_s > 0$, the estimates $\hat{\sigma}_1(t)$ and $\hat{\sigma}_2(t)$ are updated according to the following piecewise constant adaptation law:

$$\begin{aligned}\begin{bmatrix}\hat{\sigma}_1(t)\\ \hat{\sigma}_2(t)\end{bmatrix} &= \begin{bmatrix}\hat{\sigma}_1(iT_s)\\ \hat{\sigma}_2(iT_s)\end{bmatrix}, \quad t \in [iT_s, (i+1)T_s)\\ \begin{bmatrix}\hat{\sigma}_1(iT_s)\\ \hat{\sigma}_2(iT_s)\end{bmatrix} &= -\begin{bmatrix}1 & 0\\ 0 & \mathbb{I}_3\end{bmatrix}\left[B_m\ B_{um}\right]^{-1}\Phi^{-1}(T_s)e^{A_m T_s}\widetilde{x}(iT_s),\\ i &= 0, 1, 2, 3, ...,\end{aligned} \tag{3.5}$$

where

$$\Phi^{-1}(T_s) = A_m^{-1}\left(e^{A_m T_s} - \mathbb{I}_4\right)$$

and $\widetilde{x}(t) = \hat{x}(t) - x(t)$ is the state prediction error.

#### 3.4.2.3 Control Law

The control law is generated as the output of the following system:

$$u_a(s) = -k_a D(s)\hat{\eta}(s), \tag{3.6}$$

where $\hat{\eta}(s)$ is the Laplace transform of the signal

$$\hat{\eta}(t) \triangleq u_a(t) + \hat{\eta}_1(t) + \hat{\eta}_2(t)$$

with $\hat{\eta}_1(t) = \hat{\sigma}_1(t)$ and $\hat{\eta}_2(s) = H_1^{-1}(s)H_2(s)\hat{\sigma}_2(s)$ and

$$H_1(s) = C_m(s\mathbb{I} - A_m)^{-1}B_m,$$
$$H_2(s) = C_m(s\mathbb{I} - A_m)^{-1}B_{um}.$$

Here $k_a$ is a feedback gain and $D(s)$ is a strictly proper transfer function, which lead to a strictly proper stable

$$C(s) \triangleq \frac{k_a D(s)}{1 + k_a D(s)},$$

where $C(s)$ is a low pass filter with DC gain $C(0) = 1$.

## 3.5 Optimal Joint-Space Trajectory Generation

This section discusses a novel approach to task planning for these kinds of UAS based on real-world proven control techniques of redundant manipulators. The idea revolves around exploring the additional DoF's provided by the UAS and transforming the system into a redundant manipulator, that is, there are infinitely many joint-space trajectories for every workspace trajectory. The implication is that traditional workspace trajectory generation techniques fail. In this case, we propose a modification to optimal rate control methods so that these can be used on a UAS. First, we state the problem formulation:

**Problem Formulation 3.2** *(Optimal Trajectory Generation for a Redundant Manipulation UAS) Find a minimum energy joint-space* ***trajectory*** $\dot{q}$ *that reaches a desired end-effector position and satisfies* ***mission-specific*** *constraints. This is equivalent to the minimization problem*

$$\begin{aligned} \min_{\dot{q}} \quad & \|\dot{q}\|_2^2 \\ s.t. \quad & \textit{Mission constraints.} \end{aligned} \tag{3.7}$$

*Assumptions:*

1. *A stabilizing controller exists that tracks the solution of* (3.7).
2. *The yaw angle always remains aligned with the global frame.*

The assumptions are realistic given the results of the preceding section. The most traditional constraints are spatial (desired end-effector location), mechan-

ical (maximum joint travel) and energy-weighted (control cost of some joints over the others). The careful reader will note that this is a least-squares minimization problem. As another remark, we note that $\|\dot{q}\|_2^2$ is not exactly energy, but only proportional to it. Later in this section, there will be mathematical relations developed to better translate this problem into *real* minimum energy. Before continuing it is also important to note what is being optimized. The objective of the solution of (3.7) is to generate the *next* joint-space waypoint from the current location. The choice of this point takes into account the mission and, most notably, how expensive it is to move each joint. The optimality of the solution exists in the sense that from the infinite set of joint-space trajectories (from the redundancy property) to get to the next waypoint we are choosing the one which minimizes the norm of the joint-space velocity, based on user-defined weights. We can now show how to bridge the gap between redundant manipulator trajectory planning theory and *any* aerial redundant manipulator system.

### 3.5.1 Realization as a Unified Manipulation System

To proceed with the concept we first need to realize the UAS as a single manipulation system. The idea is relatively natural: any UAS has four DoF which can be attained in steady-state: a position in $SE(3)$ and the yaw angle, which we denote as the generalized coordinates $(x_q, y_q, z_q, \psi)$. Combining those with the degrees-of-freedom of a Euclidean position-oriented manipulator we will have a redundant system. Since the yaw angle does not overlap with the manipulator coordinates it can be commanded externally and, thus, won't be a part of the unified system. Let $s = (x_q, y_q, z_q)$ be the UAS position and $p$ be position of the manipulator's moving platform. We have

$$\dot{p} = J(q)\dot{q}$$

as the mapping from joint rates to position rates. Let $\mathbf{x} \in SE(3)$ be the global end-effector position and $\mathbf{q} = (s, q)$ be the joints of the unified system. Then

$$\dot{\mathbf{x}} = J_u(\mathbf{q})\dot{\mathbf{q}},$$

where $J_u(\mathbf{q})$ is the Jacobian of the unified system:

$$J_u(\mathbf{q}) = \begin{bmatrix} I_{n\times n} & J(q) \end{bmatrix} \tag{3.8}$$

and $n$ is the dimension of the workspace (usually 2 or 3). This realization opens many doors for trajectory generation and tracking, which are explored throughout the rest of this section. Note that the non-squareness of $J_u$ makes a kinematics-based approach to trajectory generation impossible.

### 3.5.2 Exact Unconstrained Solution

This solution (together with many others discussed in this section) is presented in [41]. For a given desired end-effector velocity we have that the exact solution is

$$\dot{\mathbf{q}} = J_u^{\dagger}\dot{\mathbf{x}}, \tag{3.9}$$

where $J_u^{\dagger}$ is the Moore-Penrose pseudoinverse and we omit its dependency on $\mathbf{q}$ for the benefit of compactness. In addition, (3.9) is a solution to the unconstrained version of (3.7). If $J_u$ is non-singular, then $J_u^{\dagger} = J_u^T(J_u J_u^T)^{-1}$. This compact result reflects the efficiency one can attain by moving away from position-based methods. It is still required to account for singularity positions and mission constraints, as shown in the following sections.

### 3.5.3 Spatial Constraints

A basic constraint is a final end-effector position. This constraint is formulated as a running cost by setting $\dot{\mathbf{x}}$ to be proportional to the position error with a saturation function:

$$\dot{\mathbf{x}} = \text{sat}(k_x(\mathbf{x_d} - \mathbf{x_e}), \delta), \tag{3.10}$$

where $\delta$ is a column vector containing the maximum speed of each joint, $\mathbf{x_d}$ and $\mathbf{x_e}$ are the desired and current end-effector positions, respectively, and $k_x$ is manually tuned proportional gain. The saturation function convention

throughout this dissertation is

$$\text{sat}(\alpha, \beta) = \begin{cases} \beta, & \alpha > \beta \\ \alpha, & -\beta \geq \alpha \geq \beta \\ -\beta, & \alpha < -\beta \end{cases},$$

where in the case of the arguments being vectors the operations are evaluated element-wise. Substituting (3.10) into any of the solutions presented in this section will generate a trajectory which asymptotically converges to the desired position.

### 3.5.4 Singularity Avoidance

As the system approaches a singularity (which may happen for a variety of reasons), the norm of the joint velocities becomes arbitrarily large [42]. To avoid this undesirable behavior a manually tuned damping factor $\lambda$ is introduced. The damped solution is

$$\dot{\mathbf{q}} = \left(J_u^T J_u + \lambda^2 I\right)^{-1} J_u^T \dot{\mathbf{x}}. \tag{3.11}$$

From (3.11) it becomes clear that the constant diagonal matrix will always make the bracketed term invertible, seeing as $J_u^T J_u \succ 0$ is symmetric.

### 3.5.5 Mechanical Constraints

In an aerial manipulation system it is often required to satisfy two forms of mechanical constraints: (i) the arm cannot attain a configuration that will impede the proper function of the propellers, and (ii) the UAS cannot get too close to certain objects while the end-effector can. The formulation (3.8) allows us to satisfy these constraints in one step. First, we write the augmented form of (3.11):

$$\dot{\mathbf{q}} = \left(J_u^T J_u + W_{LA}(\mathbf{q}) + \lambda^2 I\right)^{-1} J_u^T \dot{\mathbf{x}}, \tag{3.12}$$

where an example weight matrix for the UAS in this dissertation is

$$W_{LA}(\mathbf{q}) = \begin{bmatrix} L_1(q_1) & 0 & 0 & 0 & 0 & 0 \\ 0 & L_2(q_2) & 0 & 0 & 0 & 0 \\ 0 & 0 & L_3(q_3) & 0 & 0 & 0 \\ 0 & 0 & 0 & L_4(q_4) & 0 & 0 \\ 0 & 0 & 0 & 0 & L_5(q_5) & 0 \\ 0 & 0 & 0 & 0 & 0 & L_6(q_6) \end{bmatrix} \tag{3.13}$$

and $L_i(q_i)$ represents a potential function for the $i$-th joint. The potential function is manually tuned according to the following guidelines:

- A zero value allows the joint to move freely.
- A high value is equivalent to introducing a high damping factor and, thus, tends to stop the joint.
- A small negative value repels the joint in the opposite direction to the unconstrained optimal.
- A large negative value is equivalent to a high value.

The reasoning behind these rules is obvious. The potential function should be smooth and it suffices to have an exponential form.

### 3.5.6 Energy-Weighted Solution

This section aims to allow the mission planner to increase the cost of moving one joint over another. Not only is this a way of better approximating the minimization problem to *real* energy but also enables a high-level controller to decide when it is beneficial or necessary to move the manipulator, as any such movement introduces undesirable dynamics.

A simple modification to (3.12) will achieve the desired results:

$$\dot{\mathbf{q}} = \left(J_u^T J_u + W_{LA}(\mathbf{q}) + \lambda^2 W_A\right)^{-1} J_u^T \dot{\mathbf{x}}, \tag{3.14}$$

where an example weight matrix for the UAS in this dissertation is

$$W_A = \begin{bmatrix} A_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & A_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & A_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & A_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & A_6 \end{bmatrix}. \tag{3.15}$$

Here, $A_i$ represents the actuation cost for the $i$-th joint. If the mission planner wants an exact minimum energy solution the weights will be the moving mass or inertia (depending on type) of each link. Similarly, a high weight will be equivalent to aggressively damping the joint, automatically reducing its speed. It is worth to state the remark that excessive damping applied to too many joints will cause the tracking error per step to increase, slowing the rate of convergence to the desired end-effector position.

### 3.5.7 Additional Task Constraints

The goal here is to modify the cost function in order to reach a desired manipulator configuration goal. This is equivalent to writing

$$\dot{\mathbf{q}} = \left(J_u^T J_u + W_{LA}(\mathbf{q}) + J_T^T W_T J_T + \lambda^2 W_A\right)^{-1} \left(J_u^T \dot{\mathbf{x}} + J_T^T W_T \dot{\mathbf{q}}_T\right), \tag{3.16}$$

where an example weight matrix for the UAS in this dissertation is

$$W_T = \begin{bmatrix} T_1 & 0 & 0 \\ 0 & T_2 & 0 \\ 0 & 0 & T_3 \end{bmatrix} \tag{3.17}$$

$$J_T = \begin{bmatrix} 0_{3\times 3} & I_{3\times 3} \end{bmatrix}, \tag{3.18}$$

and $T_i$ represents the weight given for each joint's additional task. For example: if it is paramount that $\theta_1$ reaches a desired goal, then $T_1 >> T_i,\ i = 2, 3$. Similar to (3.10) we define the additional task's desired velocity as

$$\dot{\mathbf{q}}_T = \text{sat}(k_q(\mathbf{q}_{a_d} - \mathbf{q}_a), \zeta), \tag{3.19}$$

where $\zeta$ is the column vector containing the manipulator's joints maximum speeds, $\mathbf{q_{a_d}}$ is the desired final configuration, $\mathbf{q_a}$ is the current configuration, and $k_q$ is a manually tuned proportional gain.

## 3.6 Results

The simulation scenario is a simple movement that pulls the end-effector from 80mm to a position aligned with the $z$-axis of the center of mass of the UAS. The choice of scenario is motivated to compare this prototype with the one developed in [9]. The simulation scenario is depicted in Fig. 3.4. The simulation environment is a realistic simulator that includes the battery, motor, atmosphere, sensor, and radio models, among other details, which add realistic uncertainties and noise to the system.

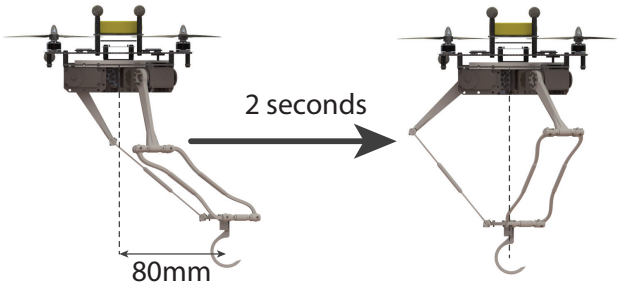



Figure 3.4: Simulation scenario.

### 3.6.1 Computed Torque Feedforward (CTF)

Here we first simulate the system without any payload. As we can see from Fig. 3.5, in the absence of a payload the uncertainties are minimal and, thus, we attain satisfactory performance. This does not reflect reality, however. Figure 3.6 shows how the addition of a payload increases the model uncertainty dramatically and, thus, the performance degrades.

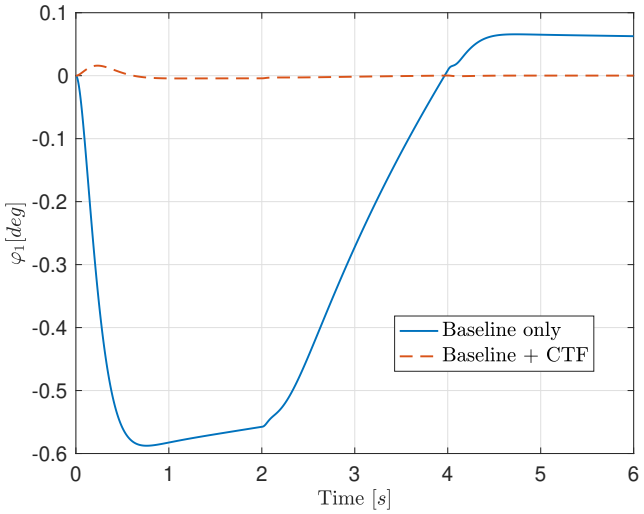



Figure 3.5: Simulation: baseline versus CTF, no payload.

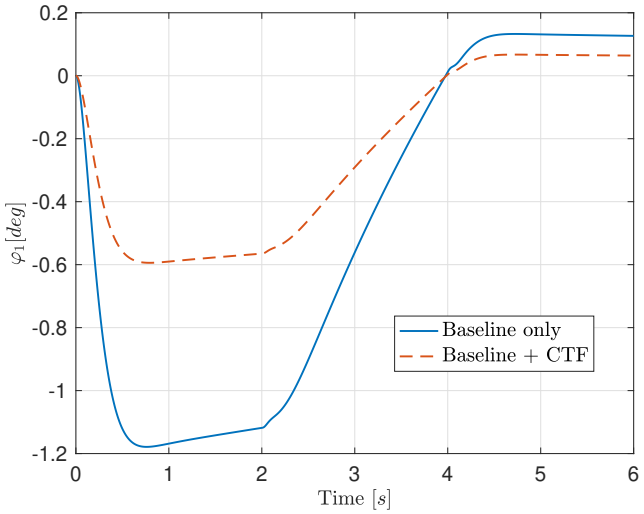



Figure 3.6: Simulation: baseline versus CTF, 25g payload.

### 3.6.2 $\mathcal{L}_1$ Adaptive Controller

The addition of the $\mathcal{L}_1$ adaptive controller proves to be necessary to account for model uncertainties. Figure 3.7 compares the different levels of compensation. Figure 3.8 includes sensor noise.

### 3.6.3 Comparison against serial-type

One main result of this dissertation is how the novel UAS design fares against the traditional serial-type approach. Figure 3.9 shows the time evolution of $\tau_m$. The torques on the serial-type change much faster than the ones in the

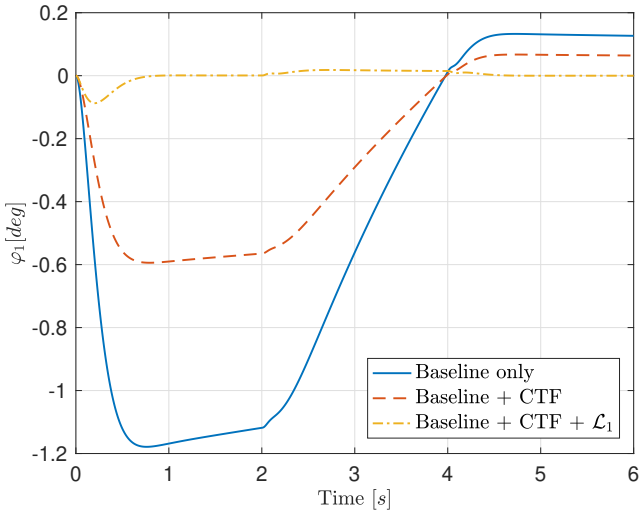



Figure 3.7: Simulation: baseline versus CTF versus $\mathcal{L}_1$-augmentation, 25g payload.

parallel case. At four seconds the parallel robot changes from 0 to $-5$mNm, while the serial-type changes from 0 to $-30$mNm. This behavior is detrimental to tracking performance as the UAS is unable to handle the fast dynamics adequately. The pitch response for both systems is depicted in Figure 3.10. At two seconds, the parallel-type jumps to 0.005° and the serial-type jumps to 0.03°, which is a direct impact of the fast dynamics. For this comparison, the same actuator is used on both models to better isolate the contribution of the mechanical topology to the performance. Both systems are using the same actuators and are attached to the same UAS; we can conclude from the smaller tracking error that the parallel robot offers better performance. One might argue that the improvement is not significant, but it is important to note that the true benefit lies in the addition of the extra DoF as well as other advantages discussed in Section 3.1. In the context of aerial logistics shown in Figure 1.1, the benefits lie in precise control of the package location with little to no cost on flight performance.

### 3.6.4 Multi-axis stabilization

So far we have only shown movements in the $y - z$ plane which only directly affect the pitch attitude. To showcase simultaneous pitch and roll stabilization we use the trajectory in Fig. 3.11, which utilizes all of the available DoF's.

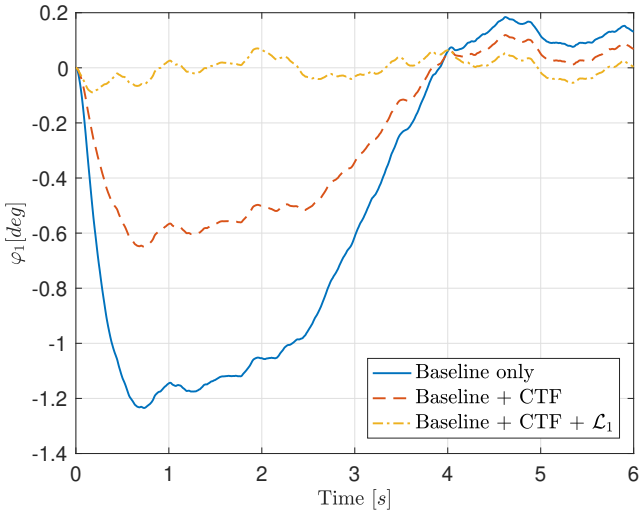



Figure 3.8: Simulation with sensor noise: baseline versus CTF versus $\mathcal{L}_1$-augmentation, 25g payload.

### 3.6.5 Optimal Trajectory Generation Algorithm

For the benefit of simplicity, we will use a 2D example that illustrates the optimal trajectory generation algorithm functionality. The example is a UAS moving in the $y-z$ plane equipped with a 2-link rotational serial manipulator aligned with the same plane. Geometrically we have a $m = 2$ dimensional workspace and $n = 4$ DoF's, so the system is indeed redundant with $d = n - m = 2$ degrees of redundancy.

#### 3.6.5.1 Spatially constrained scenario

The first task is a simple movement with no further constraints. Fig. 3.13 shows the achieved behavior. The algorithm prefers to move the arm when no weights are given. In Fig. 3.14 weights are added to reflect the dynamic cost of moving the manipulator, and the end-effector and UAS's trajectories become nearly identical meaning there is minimal movement executed by the manipulator. In Figure 3.15 we depict how the optimum strategy greatly minimizes kinetic energy over the non-weighted approach and, thus, minimizes the torques acting on the UAS.

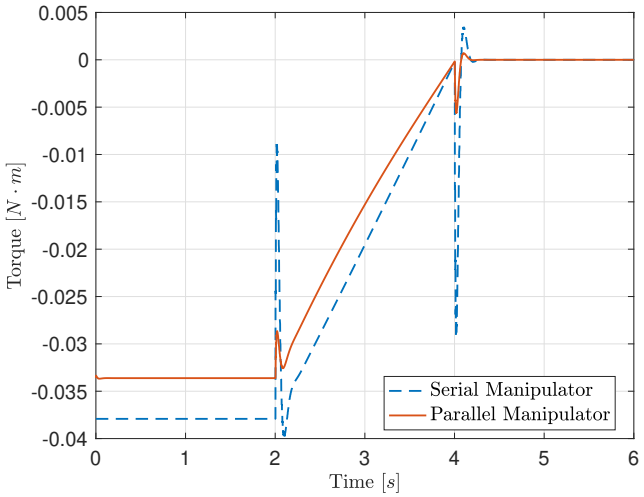



Figure 3.9: Torque profile: all controllers engaged, parallel versus serial-type, 25g payload.

#### 3.6.5.2 Mechanically constrained scenario with additional task

This scenario is similar to the previous one, but now there are two added constraints: collision avoidance and desired final configuration. In this scenario, we desire to reach the endpoint with the arm fully extended and with no collision between the UAS and the object (here depicted by a solid black sine wave). The resulting trajectory is depicted in Figure 3.16. The dashed line represents the region of influence, where the potential function is activated. We notice how the trajectory is adjusted as the UAS flies into the region of influence.

#### 3.6.5.3 Impossible task and relaxation

This is a modification of the previous scenario where the mission planner adds more constraints than the system can resolve. The endpoint resides in a position where the system would either collide or not reach the desired configuration. In Figures 3.17 and 3.18 the impossible and relaxed mission are shown respectively; in the former collision is avoided due to the higher weight given to that task, and in the latter, the mission is relaxed to be attainable.

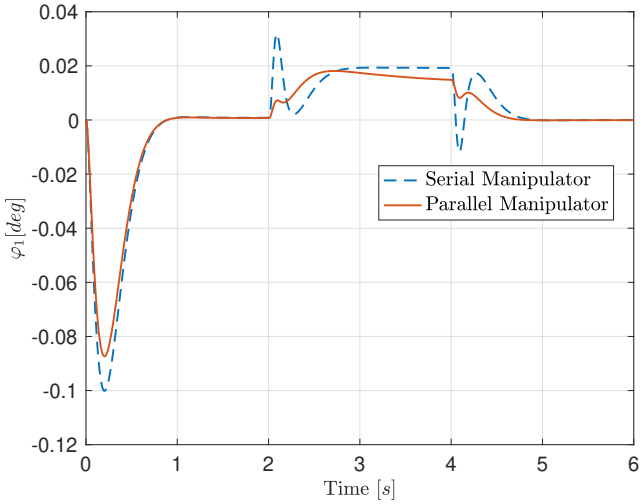



Figure 3.10: Simulation: all controllers engaged, parallel versus serial-type, 25g payload.

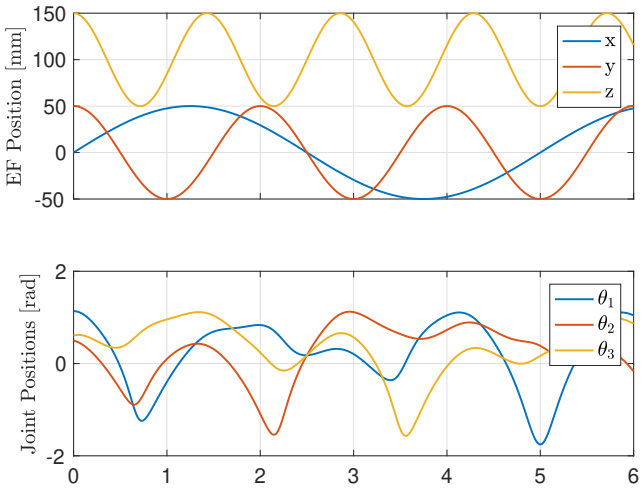



Figure 3.11: Simulation scenario: kinematic trajectories for multi-axis manipulation.

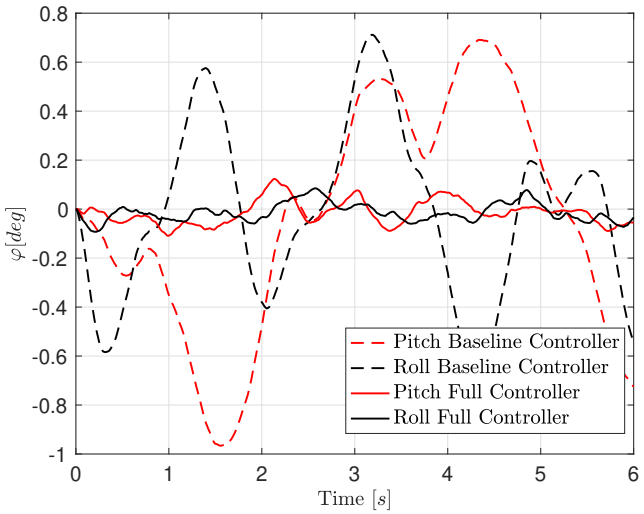



Figure 3.12: Simulation with sensor noise: all controllers engaged, multi-axis trajectory, 25g payload.

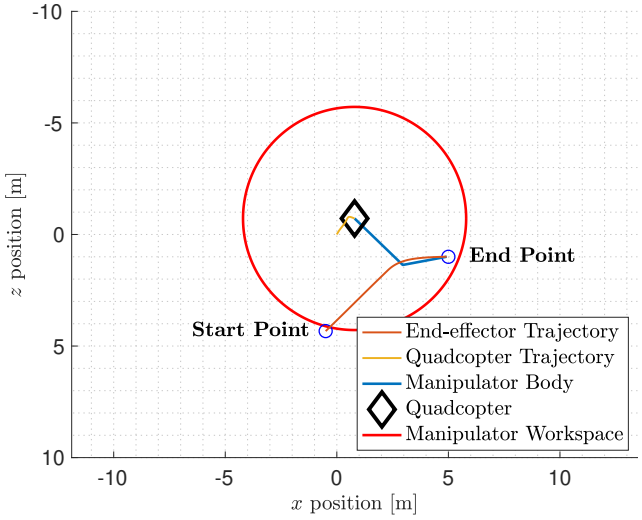



Figure 3.13: Mission: go from start to end.

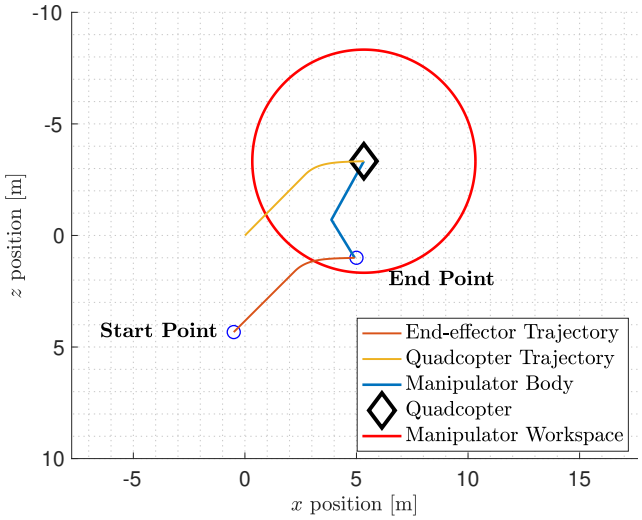



Figure 3.14: Mission: go from start to end, minimize kinetic energy.

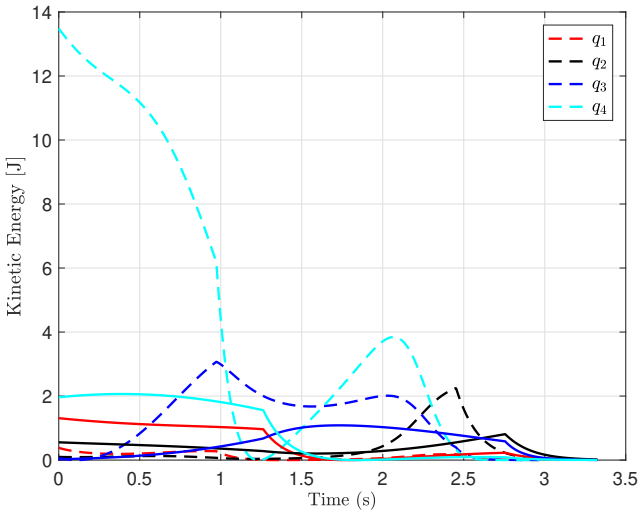



Figure 3.15: Kinetic energies for Fig. 3.13 (dashed) and Fig. 3.14 (solid), the optimum trajectory is much more efficient.

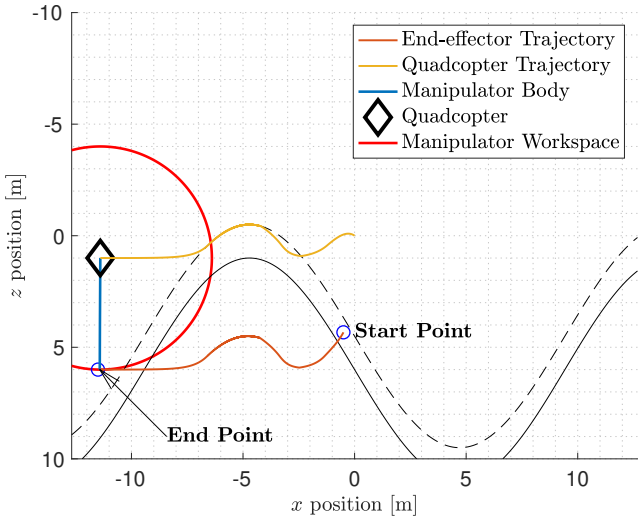



Figure 3.16: Mission: go from start to end with no collision and arrive with a desired configuration.

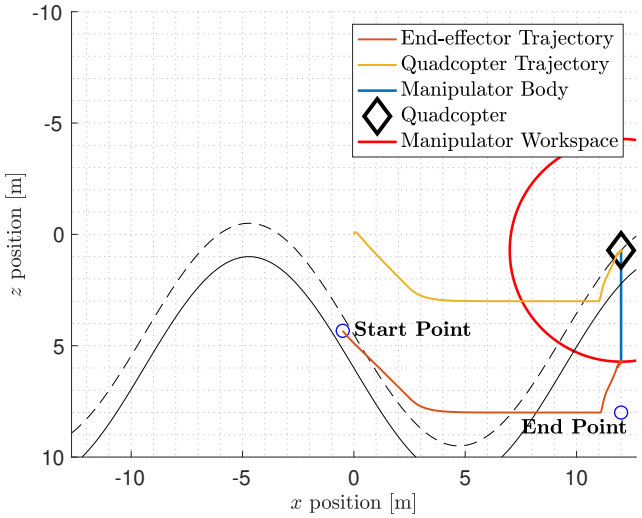



Figure 3.17: Impossible mission: go from start to end with no collision arriving with a desired configuration.

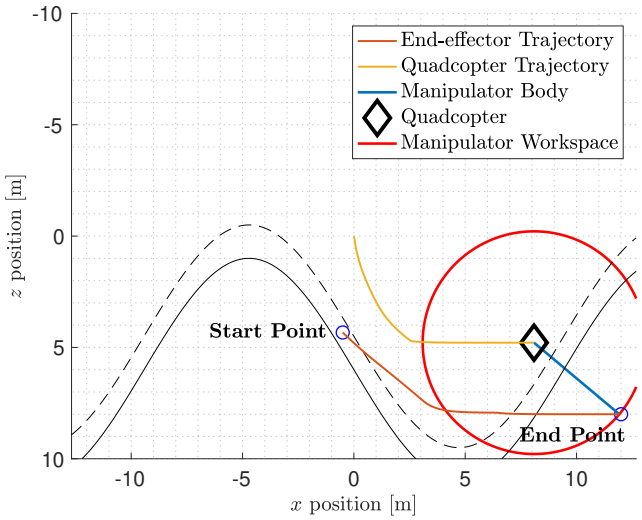



Figure 3.18: Relaxed mission: go from start to end with no collision.

# CHAPTER 4

# SINGLE PATH PLANNING

In this chapter, we focus on the high-level problem of trajectory planning of a UAV to reach a neighborhood of the ground vehicle and flying to the desired landing location, where we assume that successful take-off, rendezvous, and landing are always achievable by a local, low-level controller. We address tractability by formulating the problem with a concise risk measure directly related to mission success checked only once during the mission. Additionally, the mission is condensed into critical waypoints, which fully define the decision process and allow the heuristic layer to be designed with understandable parameters and crisp logic. Risk assessment is performed by applying Bayesian regression to vehicle behavior metrics, which provides computationally efficient equations and expressions to the MPC.

## 4.1 Statement of contributions

We present a hybrid algorithmic MPC framework to solve the running rendezvous problem in real-time under large uncertainties. We aim to condense a large-scale optimization problem into a few critical variables that minimize total time and energy consumption under the non-negligible probability of mission failure, which is handled by a robust and fast heuristic layer.

Structurally, a Bayesian learning component approximates driver behavior while enabling risk bounds to be efficiently computable. Parameterization of the path and velocities allows the data-driven Bayesian learner to remain fast, which, coupled with the MPC controller's low dimensionality, is shown to run in real-time even under highly nonlinear constraints. No approximation is made in the OCP itself; the solution is locally near-optimal up to the learned model quality.

We show that our method is flexible and robust, where a considerable portion of the computational complexity can be executed apriori. We provide a

scenario that illustrates the advantage of this approach, showing both successful and unsuccessful outcomes.

The rest of this chapter is structured as follows: in Section 4.2, we introduce and define the problem in algorithmic format. In Section 4.3, we present the two main components of this approach: the model learning and risk estimator, and the optimization problem statement. In section 4.4, we demonstrate two example scenarios showing the decision-making aspect of the algorithm. Finally, in Section 4.5, we provide concluding remarks and discuss the shortfalls of this approach.

## 4.2 Problem Formulation

We begin by defining the notion of persistent safety.

**Definition 4.1 (Persistent Safety)** *Let $x_{k+1} = f(x_k, u_k)$ be a system with state vector $x \in \mathbb{R}^n$ and control vector $u \in \mathbb{R}^m$. A safety set $S_k \subset (X, U)$ is a set, in which all states and inputs are considered safe by some measure $\rho(x) : \mathbb{R}^n \to \mathbb{R}$ at step $k$. We define a planning algorithm as persistently safe, if $S_k = \{x_k \in X,\ u_k \in U : f(x_k, u_k) \in S_{k+1}\}$ exists for all $k$ for a set of admissible states $X$ and control inputs $U$.*

The goal is to compute a persistently safe trajectory (sequence of states $x$ and inputs $u$) as defined in Definition 4.1 that satisfies a rendezvous condition. This is achieved by postponing a decision between aborting or continuing the mission for as long as possible. The additional time afforded by postponing this decision is used to improve uncertainty prediction and, consequently, reducing the risk of running out of battery or fuel.

For this problem, a parameterized path $p(\theta),\ p : \mathbb{R}^+ \to \mathbb{R}^2,\ \theta \in \mathbb{R}^+$, and historical velocity data along the path $\dot{\theta}_h(t),\ \dot{\theta}_h : \mathbb{R}^+ \to \mathbb{R}$ obtained from the traffic data are provided apriori. A stream of noisy position $\theta_d(t),\ \theta_d : \mathbb{R}^+ \to \mathbb{R}^+$, and velocity $\dot{\theta}_d(t)$ measurements from a driver moving along the path are obtained in real-time via on-board sensors. We wish to find a rendezvous point $\theta_d(t_R)$ that brings both vehicles together at a rendezvous time $t_R \in \mathbb{R}^+$. Due to sensor noise and uncertain driver behavior, we aim to estimate the distribution of $\theta_d(t_R)$ and plan on it. Along with the rendezvous point, we also determine a Point-of-No-Return (PNR) between the UAS and $\theta_d(t_R)$, from which a separate path navigates the UAS to a safe landing location in case

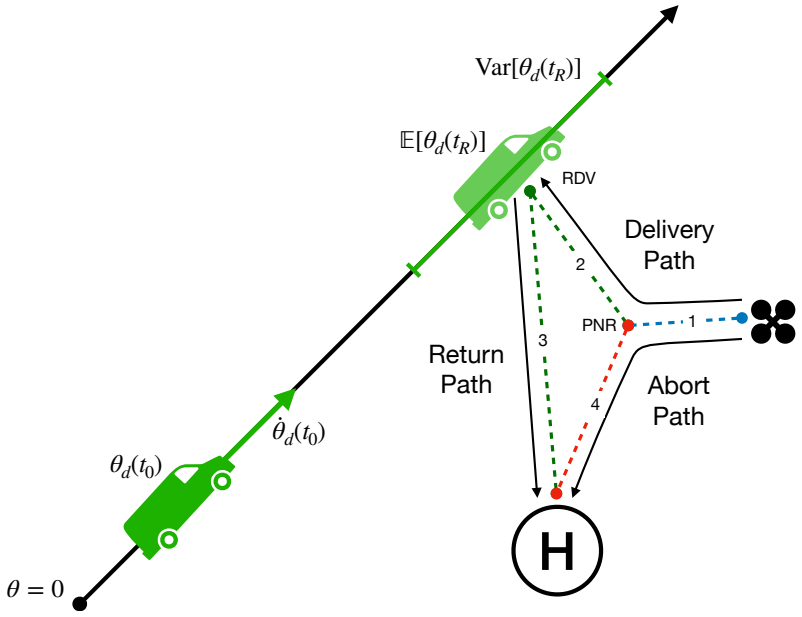



Figure 4.1: Overview of the problem setup at time instance $t_0$. Two separate paths are computed in parallel; one to rendezvous with the ground vehicle and return, and another to abort. $\mathbb{E}[\theta_d(t_R)]$ and $\text{Var}[\theta_d(t_R)]$ indicate the randomness associated with $\theta_d(t_R)$; uncertainty in driver behavior and inclusion of sensor noise imply that this prediction is a random variable.

the risk of rendezvous failure $\rho(\theta_d(t_R))$ that maps the distribution of $\theta_d(t_R)$ to $\mathbb{R}^+$ [22] is too great. We model the UAS with single integrator dynamics in this context. We define safety (and, thus, its associated risks) as a function of the probability of running out of remaining battery or fuel $E_r$. Figure 4.1 illustrates the setup.

Algorithmically, we maximize the amount of data gathered before the PNR is reached and check if that is sufficient to guarantee a low-risk rendezvous. The OCP constraints ensure that the decision time will tend to zero as $\rho$ increases, and time progresses. When this occurs, the heuristics evaluate a risk measure and commit to a route via predetermined thresholds. With this structure, the MPC is agnostic to the risk measure.

The MPC minimizes the risk associated with the rendezvous location itself ($\rho_R$), as well as the rendezvous time $t_r$. As the process is initiated, the lack of data will lead to excessively large risks; thus, it is beneficial to prolong the mission and maximize data gathering via a decision time $t_1$ between the UAS and the PNR. The stop condition is $t_1 \leq \varepsilon$. This condition indicates the moment where the decision time is at its allowable minimum, which eventually

occurs given the problem's spatial constraints. Once this threshold is met, we check $\rho \leq E_{\text{risk}}^{\max}$, where $E_{\text{risk}}^{\max}$ is defined as the maximum energy needed to complete the rendezvous trajectory inside the confidence bound $\gamma_{\max}$ of $\theta_d(t_r)$.

As the ground vehicle travels along $p(\theta)$, we collect data and append it to a dataset $\mathcal{D}$ containing the driver's velocity $\dot{\theta}_d$ and expected velocity $\dot{\theta}_h$. This dataset is then used to produce mean $\mu_w$ and variance $\Sigma_w$ functions of the driver's position in the future, a process described in Section 4.3.1. The mission algorithm does this process iteratively, sampling new data to improve the driver model and at the same time running an MPC loop to plan the routes $(\mathbf{v}, \mathbf{x}, \mathbf{t})$, defined in Section 4.3.2. If either $\rho_A$ grows above a threshold $\gamma_A$ or the decision time $t_1$ falls to $\varepsilon$, the risk measures are evaluated and a commitment is made. This process is described in Algorithm 2. Notice that traditional MPC approaches involve a moving finite time horizon, which is updated at every time step. Here the time horizon is fixed, but we update how that time is divided at each time step. Because the mission is re-planned the same way, we classify this controller as an MPC-type controller.

**Algorithm 2:** Mission Algorithm

$\mathcal{D} \leftarrow$ Initial Data
**while** $t_1 > \varepsilon$ **do**
  $\mu_w,\ \Sigma_w \leftarrow \text{Regress}(\mathcal{D})$
  $\mathbf{v}, \mathbf{x}, \mathbf{t} \leftarrow \text{MPC}(\mu_w, \Sigma_w, \mathbf{x})$
  Send Control Input $\mathbf{v}$ to UAS
  $\mathcal{D} \leftarrow \text{Append}(\text{New Data}, \mathcal{D})$
**end**
**if** $\rho(\mu_w, \Sigma_w, \mathbf{x}) \leq E_{\text{risk}}^{\max}$ **then**
  Proceed with rendezvous
**else**
  Abort and return
**end**

## 4.3 Methods

In this section we discuss the components of Algorithm 2. First, we define the learning component based on Bayesian linear regression. This regression approximates driver behavior and provides mean and variance functions for the vehicle's future location. Next, we outline the solver layer in the form

of an MPC controller, which plans a rendezvous location given the driver behavior estimate. Finally, we briefly discuss risk measures and dangers when implementing them in the proposed framework.

### 4.3.1 Bayesian Linear Regression and Risk Assessment

In this section we discuss the Bayesian learning component introduced in Section 4.2 and represented in Algorithm 2 as the regression function. One of the major challenges for the proposed problem is that each driver behaves differently. While one driver may drive at a conservative speed limit, another might drive relatively faster. Therefore, learning a driver's 'behavior' will be beneficial to the rendezvous problem. We now set up this learning problem. Consider the parameterized path $p(\theta)$, $\theta \in \mathbb{R}^+$. We assume that we have access to the driver's position $\theta_{d,i} = \theta(t_i)$, where $t_i$ is the time instance, at which the measurement is obtained. Furthermore, we have measurements of the driver's velocity denoted by $\dot{\theta}_{d,i} = \dot{\theta}_d(t_i)$. All measurements are considered to have additive noise. We also assume that we have access to historical velocity profile given by $\dot{\theta}_{h,i} = \dot{\theta}_h(t_i)$. Such a historical velocity profile can be generated by collecting measurements of vehicles traversing the path $p(\theta)$ and fitting a distribution over it using methods similar to those in [43, 44]. In our case, we assume the historical velocity profiles are in the form of a Gaussian distribution (explained later). To summarize, given the driver's position $\theta_i$, we have access to a measurement of the driver's velocity $\dot{\theta}_{d,i}$ and the corresponding probabilistic historical velocity $\dot{\theta}_{h,i}$. A comparison of $\dot{\theta}_{d,i}$ and $\dot{\theta}_{h,i}$ thus represents a measure of the driver's behavior. In particular, we wish to learn $\dot{\theta}_d(\dot{\theta}_h) : \mathbb{R} \to \mathbb{R}$.

The traditional approach would be to directly learn the vehicle's position function $\theta_d(t)$; however, this would cause the uncertainty propagation to expand too quickly and force an abort decision too often [19]. Instead, we explore both the fact that the vehicle is constrained to a known path and that the velocity along the path has a strong prior (the historical velocity $\dot{\theta}_h(\cdot)$). A disadvantage of this approach is that an integration procedure must be carried out to estimate $\theta_d(t)$. In a regular OCP formulation, this function would be forward-Euler integrated inside the solver in the form of dynamic model constraints [45]. However, due to the coarse discretization considered in this dissertation, such implementation would not be feasible. Instead, we use a

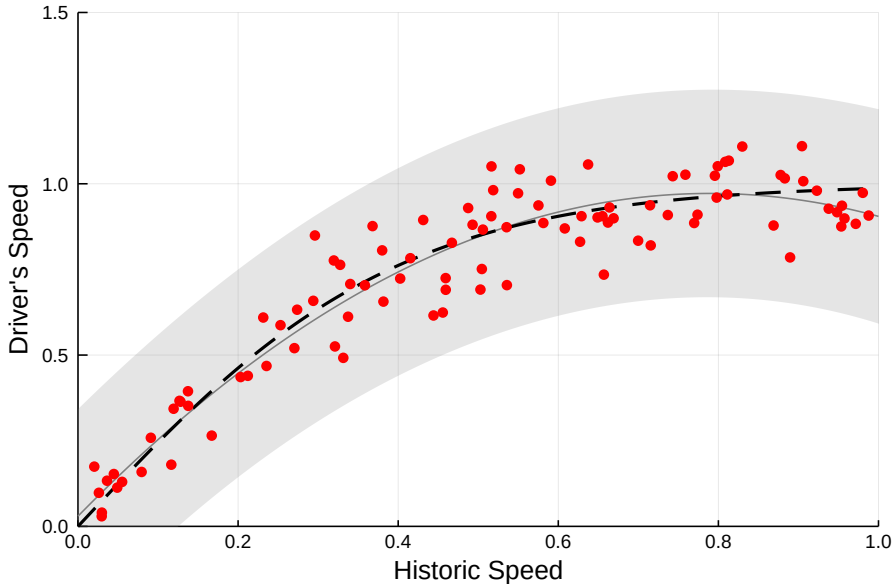



Figure 4.2: Example regression: a nonlinear curve we wish to learn (dashed) maps historical data to driver velocities, by regressing on the measurements we get: mean (gray line) and variance functions (grey shaded area).

finite basis model that enables analytical integration to be performed offline.

We assume that the mapping $\dot{\theta}_d(\dot{\theta}_h)$ admits a linear model with a finite number of basis functions as $\dot{\theta}_d(\dot{\theta}_h) = w^\top \phi(\dot{\theta}_h),\ w \in \mathbb{R}^m$, where $\phi : \mathbb{R} \to \mathbb{R}^m$ is the vector of known basis functions and $w \in \mathbb{R}^m$ are the weights to be learned. We place the following prior on the weight vector $w$ as

$$w \sim \mathcal{N}(0_m, \Sigma_m), \tag{4.1}$$

where $\Sigma_m \in \mathbb{S}^m$ is the covariance obtained using historical data. Note that we assume a zero prior mean without loss of generality. As we will see, since we consider noisy measurements corrupted by zero-mean Gaussian noise, non-zero prior mean can be easily incorporated [46, Sec. 2.7]. We assume a noisy data stream of the form

$$y_i = \dot{\theta}_d(\dot{\theta}_{h,i}) + \zeta, \quad \zeta \sim \mathcal{N}(0, \sigma^2), \quad i \in \{1, \ldots, N\}. \tag{4.2}$$

Considering the fact that by construction we have a finite-basis linear model and Gaussian measurements, we can construct the posterior distribution by conditioning the prior in (4.1) on the measurements in (4.2). This is known as Bayesian Linear Regression [46, Sec. 2.1] [47, Sec. 3.3]. For the case of a lin-

ear model with finite-number of basis functions, Bayesian Linear Regression (BLR) is equivalent to Gaussian Process Regression (GPR) with the kernel function induced by the basis functions $\phi(\cdot)$ [46, Sec. 2.1]. Then the natural question arises regarding the use of BLR and not GPR to accomplish the desired goals since the apriori choice of a finite number of basis limits the models' expressive flexibility. A straightforward argument is that BLR is computationally cheap compared to GPR, especially as a function of available data. This is crucial considering the online nature of the proposed method. Furthermore, one can always approximate the well-known stationary kernels like the Squared-Exponential (SE) or Matérn kernels using a finite number of random Fourier features [48]. Finally, we would like to highlight the fact that any estimation method which provides a notion of uncertainty can be used since the algorithm is agnostic to the risk estimation layer as it will be shown in subsection 4.3.2.

At any given time, the algorithm can sample a position $\theta_d$ and velocity $\dot{\theta}_d$ of the vehicle. Then using the prototypical velocity profile $\dot{\theta}_h$, we can generate the data in $\mathcal{D}$. We write the data in compact form as $\mathcal{D} = \{D, H\}$, where $D,\ H \in \mathbb{R}^N$ are defined as

$$D = \begin{bmatrix} \dot{\theta}_{d,1} & \cdots & \dot{\theta}_{d,N} \end{bmatrix}^\top, \quad H = \begin{bmatrix} \dot{\theta}_{h,1} & \cdots & \dot{\theta}_{h,N} \end{bmatrix}^\top.$$

Given the measurements (4.2) and the prior (4.1), we obtain the posterior distribution of the parameter vector $w$ as

$$w \in \mathcal{N}(\mu_w, \Sigma_w), \tag{4.3}$$

where $\mu_w = \frac{1}{\sigma^2} A^{-1} \Phi(H) D,\ \Sigma_w = A^{-1}$, and $A = \sigma^{-2} \Phi(H) \Phi(H)^\top + \Sigma_m^{-1}$.

With the mean and variance functions fitted, the next necessary step is to forward propagate these functions with respect to the historical data. The challenge is that the model represents a mapping between velocities, with no spatial information otherwise. Because we used parameterized velocities instead of the Euclidean representation, it is possible to integrate along the path using the known velocity profile from historic data and the path information itself. Suppose that at some time instance $t_0$ the rendezvous vehicle is at $\theta_{d,0} = \theta_d(t_0)$, and we wish to estimate the vehicle's position at some instant $t_f > t_0$. Given the integrable temporal prototypical velocity profile $\dot{\theta}_h(t)$, the

predictive distribution of $\theta_d(t_f)$ can be computed via

$$\theta_d(t_f) = \theta_{d,0} + \int_{t_0}^{t_f} (\dot{\theta}_d(\dot{\theta}_h(\tau))\,\mathrm{d}\tau = \theta_{d,0} + w^\top \psi(t_f), \tag{4.4}$$

where $\psi(t_f) = \int_{t_0}^{t_f} \phi(\dot{\theta}_h(\tau))\,\mathrm{d}\tau$. This integral can be computed, as a function of $t_f$, apriori. For example, for polynomial kernels or the random Fourier feature approximation of SE or Matern kernels, using the posterior distribution of $w$ in (4.3), and the fact that $\theta_d(t_f)$ is a linear transformation of the random variable $w \in \mathbb{R}^m$, we get the following posterior distribution

$$\mathbb{E}[\theta_d(t_f)] = \theta_{d,0} + \mu_w^\top \psi(t_f), \tag{4.5}$$

$$\mathrm{Var}[\theta_d(t_f)] = \psi(t_f)^\top A^{-1} \psi(t_f). \tag{4.6}$$

As the order of the basis increases, and depending on the structure of the historical velocity profile, the explicit form can become cumbersome. However, it is all done apriori and automated, and because the range of velocities we expect to encounter is small, we generally do not need a large number of basis.

### 4.3.2 MPC formulation

In this section, we discuss the structure and particulars of the MPC component introduced in Algorithm 2. A primary challenge of the rendezvous problem is presented by the trajectories under strict and numerous constraints, of which many are non-convex. By exploring two special features of the problem formulation we reduce dimensionality and attain tractability. We now outline the Optimal Control Problem (OCP) associated with the rendezvous problem. As mentioned previously in Sec. 4.2, the solver is tasked with finding two critical points: (Point-of-No-Return) PNR and the rendezvous location $p(\theta_d(T_R))$. To fully define the problem and gain temporal constraint management we expand the control from velocities to also include a time "input". The nature of this problem requires the UAS to coincide with the vehicle both in space and time. By introducing time as a manipulated variable in the OCP, we allow for the solver to directly decide on the optimal time and place for the rendezvous maneuver to occur. This time input works by assuming a piece-wise constant control law along each of the four segments (PNR, rendezvous, landing location, and abort location), which is possible due to our assumption on the UAS

integrator dynamics described below:

$$x_k = x_{k-1} + v_k T_s \tag{4.7}$$

$$E_{r,k} = E_{r,k-1} - \left(\frac{mv^2}{2} + \alpha m\right) T_s, \tag{4.8}$$

where $T_s$ is the sampling time, $m$ the scalar vehicle mass, and $\alpha$ the scalar hovering energy consumption constant. We represent each of the segments using the state vector $(\mathbf{x}, \mathbf{v}, \mathbf{t}) \equiv (x_i, v_i, t_i)$, $i \in \{1, \ldots, 4\}$. Here, $t_i$ represents the time to be spent at a constant velocity $v_i$ to reach one waypoint from another. Furthermore, $x_i \in \mathbb{R}^2$ represents each of the defined physical waypoints in Euclidean coordinates and $v_i \in \mathbb{R}^2$ represents velocity inputs in Euclidean coordinates. The waypoints are, in this order, the Point-of-no-Return (PNR), the rendezvous location (RDV), the landing location, and the abort location, as shown in Figure 4.1. The designed Optimal Control Problem (OCP) is given by:

$$\min_{U, T_R} \quad \rho_R(T_R, \Sigma, \mathbf{x}) + \left(\sum_{i=2}^{3} t_i - t_1\right) \tag{4.9a}$$

$$\text{s.t.} \quad x_i = x_{i-1} + v_i t_i,\ x_4 = x_1 + v_4 t_4, \tag{4.9b}$$

$$|v_i| \leq v_{\max}, \tag{4.9c}$$

$$x_3 = \mathbb{E}[\theta_d(T_R)],\ x_4 = S_L,\ x_5 = S_A, \tag{4.9d}$$

$$\sum_{i=1}^{3} t_i \leq t_{\max},\ t_1 + t_4 \leq t_{\max},\ t_c \leq t_i \tag{4.9e}$$

$$E_1 + E_2 + E_3 \leq E_r,\ E_1 + E_4 \leq E_r, \tag{4.9f}$$

where $T_R \equiv t_1 + t_2$, $S_L$ and $S_A$ are the landing and abort destinations, $E_r$ the remaining energy $E_i$, $t_c$ a dwell time for the low level controller to switch tracked segments, $\Sigma$ is a variance function associated with the mission state given by (4.3), and $\rho_R$ any risk measure we wish to minimize. The dwell time is necessary to stop the solver from placing waypoints arbitrarily close to each other and creating undesirable sharp turns, which are problematic for our single integrator dynamics assumption. Moreover, $U \equiv \{t_i, v_i\},\ i \in (1, ..., 4)$. Formulating this problem with risk constraints instead of cost would cause the solver to potentially deny dangerous solutions instead of postponing a decision and waiting for new data that can eventually yield a feasible solution.

Since risk constraints still need to exist, their satisfaction is relegated to the heuristics discussed in Algorithm 2.

All of these constraints are natural because every single one is predetermined at the design stage. For example, constraint (4.9f) is directly produced from the battery used in the UAS, and constraint (4.9d) is given from the map where the mission takes place. The only task left for the designer is to choose the risk measures. Fortunately, because the proposed method is agnostic to risk measures, the designer can choose with no concern over tractability or internal conflicts in the solver.

Quantifying risk is the effort of determining a measure $\rho$ that maps a set of random variables to a real number [22, 21]. With this definition, the random variables are the states of the UAS (due to process and measurement noises) and, more importantly, the position of the ground vehicle due to the driver's uncertain behavior. It is crucial to choose measures that reflect meaningful quantities in the problem formulation. In this framework, the risk is directly related to the uncertainty regarding the vehicle's location in the future and the limitations that the path imposes on planning. If the driver is erratic, or the path only allows the rendezvous to happen in unfavorable locations, we consider that the mission has elevated risk. Several risk measures are popular; some examples are Expectation-Variance [23], (Conditional, Tail) Value-at-Risk [21, Sec. 3.3], and Downside Variance [21, Sec. 3.2.7]. These measures can introduce nonlinearity and preclude gradient information, endangering tractability. A popular approach uses gradient-free methods, which sample these measures and choose inputs corresponding to minimum risk [49]. In this dissertation, we tackle tractability by exploring the problem geometry to its fullest and reducing the number of variables to an absolute minimum, as shown above, and in Section 4.3.1. In this dissertation, we consider two risk measures. Measure $\rho$ is the Downside Potential [21] of energy expenditure. It computes the worst outcome energy-wise as the maximum energy necessary to complete the mission within the driver position confidence bounds. Furthermore, $\rho_R$ is defined as the variance of driver position at time $t_R$. These measures provide guarantees that the UAS will return and favor locations on the road where uncertainty is minimal, respectively. Figure 4.3 depicts this setup, where one tail of the distribution is a downside tail, and the other an upside tail.

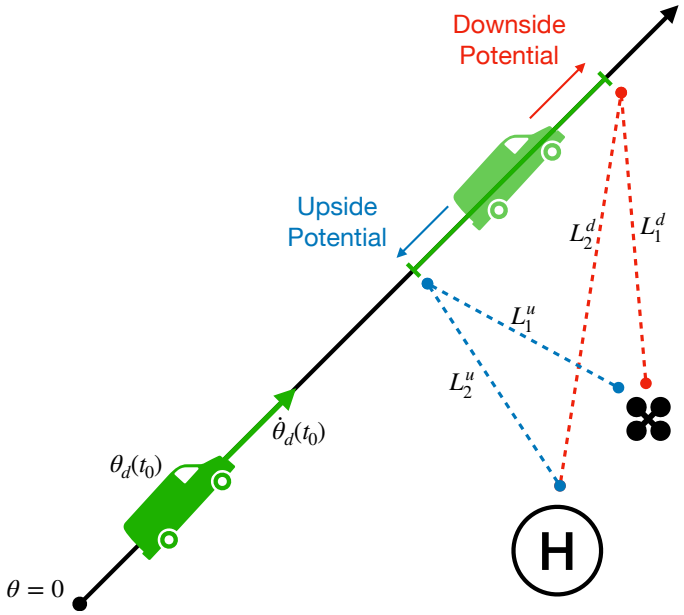



Figure 4.3: Downside Risk measure: the two outcomes within some confidence bound at time $t_R$. The downside potential consumes more power because $L_1^d + L_2^d > L_1^u + L_2^u$. The Downside Potential Energy is the extra energy necessary should the red outcome occur.

## 4.4 Results

This section presents two scenarios with distinct outcomes. In the first scenario, the mission is successful, since when the decision time $t_1$ reaches $\varepsilon = 5$ seconds, the Downside Energy Potential $\rho$ is below its threshold of 200J. In the second scenario, we increase noise and make the driver erratic, which increases uncertainty makes the UAS abort a risky rendezvous in favor of returning home. To simulate the dynamics, we use an Euler integration scheme with a discretization step of one second. The OCP solver reaches a solution in a median time of 75.604ms on a 2012 3.4 GHz Quad-Core Intel Core i7 implemented in Julia with the Ipopt solver. At this rate, the algorithm runs faster than the discretization time. The source code can be found at `https://github.com/gbarsih/Safe-Optimal-Rendezvous`. The mission takes place on a 1km$^2$ area, with a stretch of road going from (0,0)m to (1000,1000)m, as shown in Figure 4.1. We set the abort, landing, and take-off locations ($S_A$, $S_L$, and $x_0$, respectively) to the same position at $(500, 0)$m, the path to $p(\theta) = (\theta, \theta)$m, and driver behavior functions to $\dot{\theta}_d(\dot{\theta}_h) = a\dot{\theta}_h(t)$, so that $a > 1$ makes a driver proportionally faster than the average driver. On

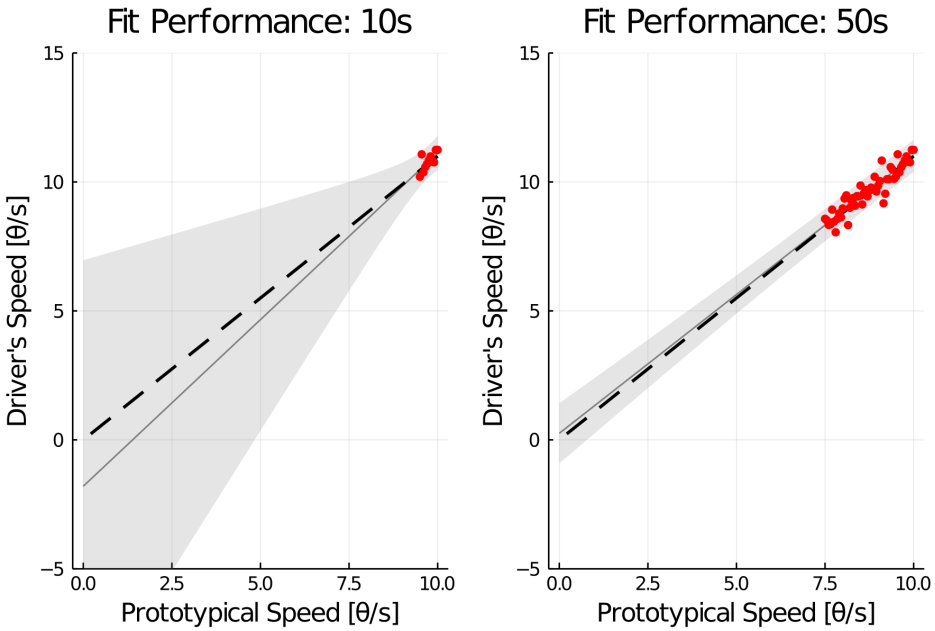



Figure 4.4: Bayesian fit performance at two points in time: more data completes the driver's behavior profile. Waiting for more data means noise is averaged, and we explore more speeds.

that same path we set $\dot{\theta}_h(t) = 10(1 - t/200)$ and $\rho_R(\theta_d) = \text{Var}[\theta_d(t_R)]$.

### 4.4.1 Maximizing Decision Time

One of the algorithm's core functionalities relies on postponing a decision by maximizing $t_1$ to gather more data and attempt to find a safe trajectory. This maximization goes against the mission objective of minimizing time but is necessary to guarantee safety and feasibility. Figure 4.4 shows exactly that, for $a = 1.1$, where more data at 50 seconds reduces uncertainty in unknown speeds.

### 4.4.2 Low-Risk Mission

In this example, we set $\sigma = 3$ and $a = 1.1$. In 20 seconds, the algorithm terminates and is forced to make a decision. We notice that the abort energy is always maximized up to the available energy and that the downside potential falls below the threshold despite being prohibitively high in the early stages. Figure 4.5 shows the energy profile of each trajectory segment, available energy,

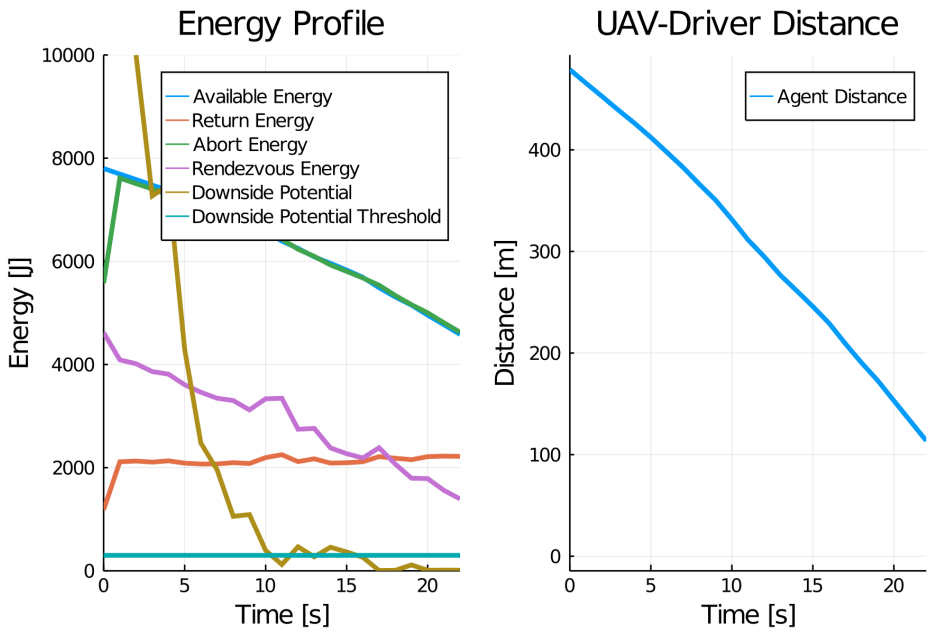



Figure 4.5: Outcome of a low-risk decision: in 20 seconds available to make a decision, enough data is collected to assert high confidence of success.

and downside potential energy, as well as the distance between the car and the UAS for the low-risk scenario.

### 4.4.3 High-Risk Mission

In this example, we set $\sigma = 6$ and $a = 1.3$, meaning that the sensor has more noise and the driver is more erratic. In 18 seconds, the algorithm terminates and is forced to make a decision. Because of the heightened uncertainty, the downside potential is high, and the UAS has a high probability of running out of battery. In this case, the algorithm will trigger the abort decision, which is feasible because that path was planned. Figure 4.6 shows the energy profile of each trajectory segment, available energy, and downside potential energy, as well as the distance between the car and the UAS for the high-risk scenario.

## 4.5 Conclusions

We presented an algorithm capable of planning a long-distance rendezvous between an autonomous aerial vehicle and a ground vehicle traversing a path. The large uncertainties associated with driver behavior combined with finite

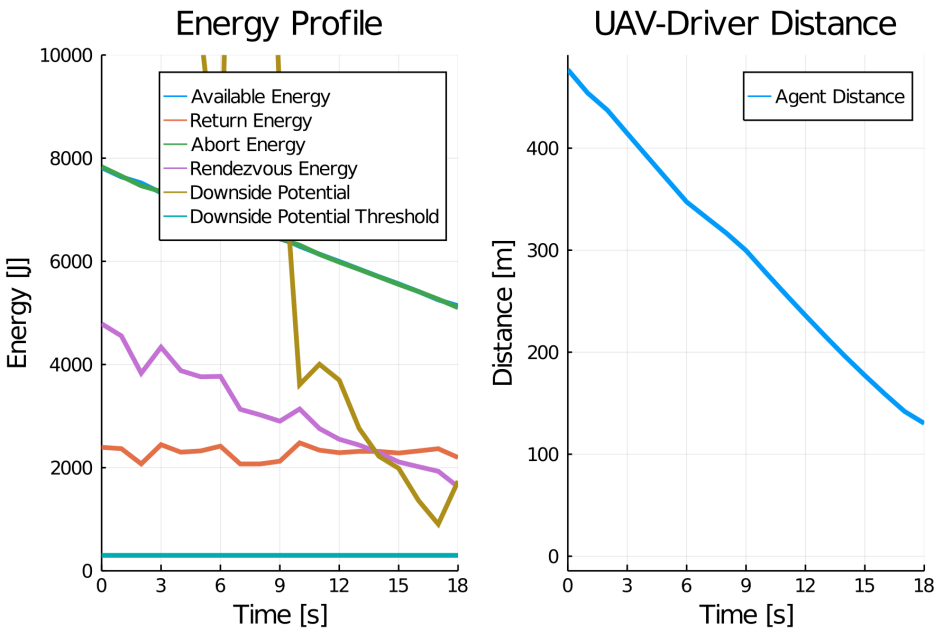



Figure 4.6: Outcome of a high-risk decision: in 18 seconds, not enough data was collected, or it was, and risk is too high. At decision time, the downside potential is above the threshold, and a decision is made to abort the mission.

energy to complete the mission require risk management, a learning component, and an MPC-like controller to work in unison and guarantee safety at all times, even if safety translates to aborting the mission. The algorithm is shown to be persistently safe due to the continuous planning of an abort path at all times. For future work, we intend to expand this work in two ways. First, a modification of this algorithm to accommodate pruning paths. Second, a parcel delivery framework like the one described here will have multiple vehicles in the logistic matrix. A third decision path can be added by leveraging this, which waits for a new ground vehicle to enter the matrix and then deliver to that vehicle.

# CHAPTER 5

# MULTIPLE PATHS PLANNING

In Chapter 4 we consider the task planning problem of guiding a UAS to the neighborhood of a human-operated vehicle traveling along a known path. Uncertain driver behavior and the large distances the UAS needs to cover required the parallel planning of one risk-enabled path that rendezvous with the driver and one deterministic return path. The motivation is that under the conditions where a rendezvous is only safe if there is a low probability of running out of battery, we can increase the potential range of the mission if we have high assurance that the UAS can meet the driver, and hence have less payload on the way back. Should we commit to such a plan with a high risk of not meeting the driver, there is a high probability that the package will not be delivered, and the extra mass will cause the UAS to deplete its battery prematurely and crash on the way back.

In this chapter, we allow the driver not to be constrained to a single path; instead, the driver can choose among $N$ different parametrized paths. We propose a sampling-based method similar to that in [15]. After selecting the best sample as a rendezvous location candidate, an MPC-like controller generates inputs for two trajectories. One trajectory rendezvous with the car in future time and another returns to a safe landing location. The decision between which trajectory to follow is made by probabilistic heuristics that monitor risk measures based on the sampling statistics. In Section 5.2 we formalize the problem setup.

## 5.1 Related work

Sampling-based motion planning has grown in popularity with the increased computational power afforded by modern CPUs and Graphics Processing Units (GPUs). The core idea behind these methods is to cleverly sample inputs from some distributions and use these samples' quality to update the distribution

and improve the next batch of samples. Ultimately, the goal is to converge to a narrow distribution centered around the optimal input (or set of inputs) to the system. In this dissertation, we sample rendezvous times and their associated rendezvous location. Two approaches are directly related to this dissertation. In [15] the authors present a novel method to find trajectories for mobile robots in cluttered environments. In [50], the authors present a sampling-based MPC that integrates risk management using Conditional Value-at-Risk (CVaR) [21, Sec. 3.3]. Both motion planning and usage of CVaR are relevant to this dissertation, both of which are described in Section 5.3.

### 5.1.1 Statement of contributions

We present a hybrid algorithm that is capable of attempting to rendezvous with a human-operated ground vehicle and does not crash with guaranteed safety bounds. The algorithm handles three significant challenges: driver behavior, multiple possible routes, and non-differentiable paths. To safely attempt a rendezvous, three main components are needed.

A Gaussian Process Regression learning module collects sensor data from the ground vehicle and builds a driver model. Unique to this dissertation is the way we pose this learning problem. Instead of modeling future driver position, we leverage historical traffic data from the area to learn how the driver deviates from an average virtual driver. This way, we significantly reduce the problems associated with uncertainty propagation. This benefit is only possible by the fact that we know all reachable roads a priori.

To address pathing complexities, we use a modified version of [15] that accepts a specialized risk measure and integrates with the learning and MPC layers instead of finding optimal trajectories directly. This novel way of sampling trajectories relies on fast Gauss-Kronrod Quadratures to estimate sampling quality, an approach that can benefit from parallel computing hardware such as GPUs.

The last component is an MPC-like controller. Unlike traditional MPC, our formulation uses the time horizon as an input, allowing a compact set of variables capable of timing control. The temporal component is crucial because an optimal rendezvous has the UAS reaching the ground vehicle precisely both in space and time. To achieve this, we use the fact that the mission's spatial scale is large enough that single-integrator dynamics are a satisfactory

approximation.

These three modules share critical information backward and forwards between each other and provide a robust, efficient, and concise set of hyperparameters. The rest of this dissertation is structured as follows: in Section 5.2, we introduce and define the problem in algorithmic format. In Section 5.3, we present the three main components of this approach: the model learning, the importance sampling scheme, and the MPC controller. In section 5.4, we demonstrate results for individual modules and two full planning stack examples. Finally, in Section 5.5, we provide concluding remarks and discuss the shortfalls of this approach and future directions to address them, respectively.

## 5.2 Problem Formulation

The goal is to compute a persistently safe trajectory (sequence of states $x$ and inputs $u$) as defined in Definition 4.1 that satisfies a rendezvous condition. This computation is achieved by postponing a decision between aborting or continuing the mission for as long as possible. The additional time afforded by postponing this decision is used to improve uncertainty prediction and, consequently, reduce the risk of running out of battery or fuel.

Consider a set $\mathcal{P}$ of $N$ indexed parametrized paths $\mathcal{P} = \{p_j(\theta),\ j = 1, \ldots, N\} \subset \mathcal{R}$, $p : \mathbb{R}^+ \to \mathbb{R}^2,\ \theta \in \mathbb{R}^+$ on a planar Euclidean region $\mathcal{R} \in \mathbb{R}^2$, and historical velocity data along each path $\dot{\theta}_j^h(t),\ \dot{\theta}_j^h : \mathbb{R}^+ \to \mathbb{R}$ obtained from the traffic data that are provided apriori. A stream of noisy position $\theta^d(t)$, $\theta^d : \mathbb{R}^+ \to \mathbb{R}^+$, and velocity $\dot{\theta}_j^d(t)$ measurements from a driver moving along any of the paths are obtained in real-time via on-board sensors. Let $\varkappa$ be set of all permutations of $\{1, \ldots, N\}$. Let a path intersection set be defined as $\mathcal{I} = \{\{\theta, p\} \in \mathbb{R}^2 | p_i(\theta) = p_j(\theta), \{i, j\} \in \varkappa\}$. The intersection set $\mathcal{I}$ contains all path segments that intersect each other before terminally pruning. The purpose of this set is to identify values of $\theta$ for which we are uncertain of what path the driver will choose. We ignore paths that the driver may no longer choose, i.e. a path which the driver passed and chose not to turn into. Obviously this also makes it so that, within $\mathcal{I}$, the historical data is the same across intersecting paths. We wish to find a rendezvous point $\theta_j^d(t_{R,j})$ that brings both vehicles together at a rendezvous time $t_{R,j} \in \mathbb{R}^+$, for some path $j$.

Due to sensor noise and uncertain driver behavior, we aim to learn the

distribution of $\theta_j^d(t_{R,j})$ and plan on it. In single path problems the distribution $g(\theta_d|t_R)$ is distributed along the path. For this problem, however, each path will have its own distribution $g_j(\theta_j^d|t_{R,j})$. We approximate driver behavior by learning a deviation mean function $d(\theta_j^h(t)) : \mathbb{R} \mapsto \mathbb{R}$ and variance function $\Sigma_d(\theta_j^h(t)) : \mathbb{R} \mapsto \mathbb{R}$. The deviation function is such that $\theta_j^d(t) = \theta_j^h(t) + d(\theta_j^h(t))$ if learned exactly. Section 5.3.1 explains this learning process.

The next step is to use the driver model to find quality rendezvous candidates comprised of time and location pairs. This search is done through stochastic optimization, explained in Section 5.3.3. Assume each path has an optimal rendezvous location local to the path. The random nature of the problem makes the rendezvous locations stochastic, with an $N$-dimensional distribution $\mathcal{A}^\star(\mu_\mathcal{A}^\star, \Sigma_\mathcal{A}^\star)$. Since we do not have knowledge of $\mathcal{A}^\star$ we aim to approximate it by manipulating the parameters of an ancillary distribution $\mathcal{A}(\mu_\mathcal{A}, \Sigma_\mathcal{A})$ with equal dimension. To estimate $\mathcal{A}$ we require a driver model ($d$ and $\Sigma_d$) and some way of optimizing the parameters of $\mathcal{A}$ such that $\mathcal{A} \to \mathcal{A}^\star$ as $t \to \infty$. If we are successful, we can find the optimal path to rendezvous with defined as $p_{\text{tgt}}$, such that $p^\star = p_{\text{tgt}}(\theta_{\text{tgt}}^d(t_{R,\text{tgt}}))$ is the optimal rendezvous location.

With knowledge of $p^\star$, the next step is to find a trajectory that guides the UAS to that point in space and time. We wish to have guarantees that the UAS will not run out of battery. To achieve this, the trajectory planner plans two options; one that rendezvous with the ground vehicle and another that returns to a safe landing location. Because the latter (abort path) does not depend on any uncertainties, we are guaranteed to land safely by choosing that option. The cost is that we do not rendezvous and render the system sub-optimal. To mitigate this problem, we find the two paths by planning a Point-of-No-Return (PNR) between the UAS and $p^\star$, from which a separate path navigates the UAS to a safe landing location in case the risk of rendezvous failure $\rho_d(p^\star)$ is too great. The risk measure $\rho_d$ maps the distribution of $p^\star$ and system states to $\mathbb{R}^+$. We model the UAS as a system capable of tracking single integrator dynamics in this context. We define safety (and, thus, its associated risks) as a function of the probability of running out of remaining battery or fuel $E_r$. Figure 4.1 illustrates the setup for a single path. In Section 5.3.4 we present the multi-path setup, which is illustrated in Figure 5.1.

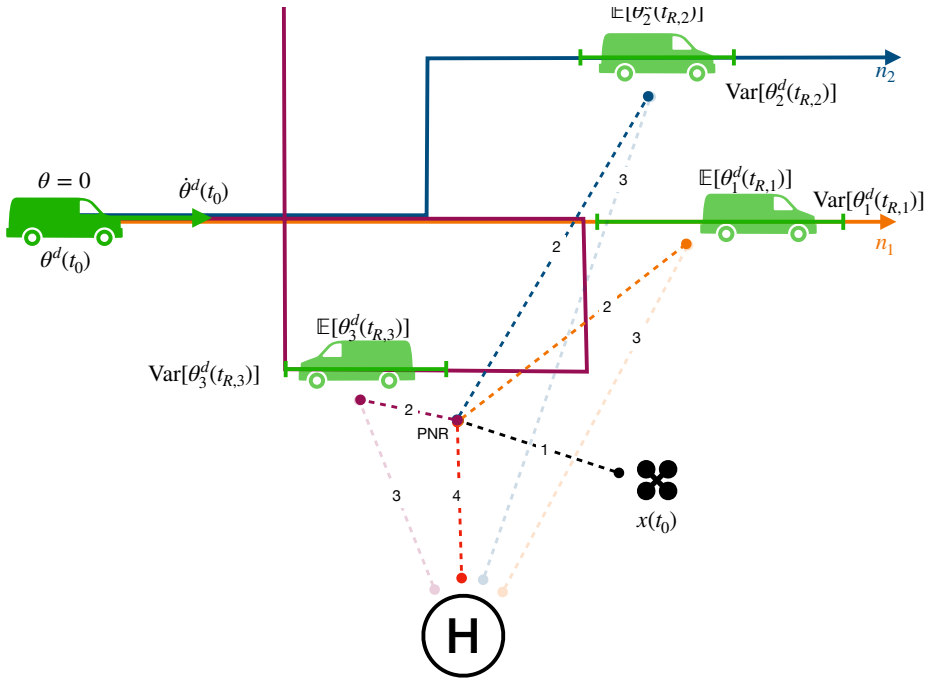



Figure 5.1: Overview of the problem setup at time instance $t_0$ for multiple paths. Additional uncertainty in path choice: each path has it's own individual driver position uncertainty. Planning for all outcomes is intractable.

The discretized single integrator dynamics of the UAS are

$$x_k = x_{k-1} + v_k T_s, \tag{5.1}$$

where $x_k \in \mathbb{R}^2$ is the Euclidean UAS position, $v_k \in \mathbb{R}^2$ is the Euclidean velocity input, and $T_s$ the sampling time. Additionally, the remaining energy has the following dynamics:

$$E_{r,k} = E_{r,k-1} - \left(\frac{mv^2}{2} + \alpha m\right) T_s, \tag{5.2}$$

where $E_{r,k}$ is the remaining energy, $m \in [m_a, m_b]$, $m_a > m_b$ is the mass of the UAS (for a package drop-off mission, the mass decreases after the package is dropped on the ground vehicle), and $\alpha$ is the scalar hovering energy consumption constant.

Note that $m$ will decrease from $m_0$ to $m_1$ after the package is dropped off on the ground vehicle, decreasing the energy consumption rate and increasing the range. This is the detail that makes it beneficial to commit to a plan so that we can reach further and increase efficiency. Additionally, given this

problem's large scale, the single integrator assumption is not strong. At this scale, virtually any controllable robot will be able to track these dynamics without difficulties. Adaptive control techniques such as [51, 52] can formalise this assumption. We can now present the problem formulation.

**Problem Formulation 5.1 (Risk-Averse Multi-Path Rendezvous)** *Given a map of $N$ paths, historical velocity data along each path $\dot{\theta}_i^h(t)$, and a stream of noisy position $\theta^d(t)$ and velocity $\dot{\theta}^d(t)$ measurements from a driver traversing an intersection of any subset of the paths, find a persistently safe sequence of control inputs and a future time $t_R$ such that the UAS reaches a neighbourhood of the driver at time $t_R$ and flies to a safe pre-determined landing location. This trajectory is the solution to the following optimization problem*

$$\begin{aligned} \min_{U, t_R, t_L} \quad & L(x_k, u_k, \mathcal{D}) - t_D \\ s.t. \quad & x_k = x_{k-1} + v_k T_s \\ & ||x(t_{R,\text{tgt}}) - p^\star|| \leq \varepsilon,\ x(t_L) = S_L \\ & E_{r,k} = E_{r,k-1} - \left( \frac{mv^2}{2} + \alpha m \right) T_s \\ & E_r(t_{R^\star}) \geq 0, \end{aligned}$$

*where $L(\cdot)$ is a cost function that minimizes risk, input costs, and delivery time, $t_D$ is the decision time between the current state $x(t_0)$ and Point-of-No-Return (PNR), $t_{R,\text{tgt}}$ is a rendezvous time for path* tgt*, $p^\star$ is a rendezvous location, $\varepsilon$ is a small positive number, $t_L$ is a landing time for a landing location $S_L \in \mathbb{R}^2$, $x(t)$ and $E_r(t)$ are continuous time realizations of $x_k$ and $E_{r,k}$ respectively, and $\mathcal{D} = \{D, H\}$ is a data set containing measurements from the ground vehicle, where $D,\ H \in \mathbb{R}^M$ are defined as*

$$D = \begin{bmatrix} \dot{\theta}^{d,1} & \dots & \dot{\theta}^{d,M} \end{bmatrix}^\top, \quad H = \begin{bmatrix} \dot{\theta}^{h,1} & \dots & \dot{\theta}^{h,M} \end{bmatrix}^\top,$$

*and $M$ denotes the number of measurements collected, $\dot{\theta}^{d,j}$ are the driver samples, and $\dot{\theta}^{h,j}$ is the expected velocity at the GPS-collected point $\theta^{d,j}$ obtained from historical data.*

In Section 5.3 we present the tools that solve Problem 5.1 by altering its different components. Although we never solve the Optimal Control Prob-

lem shown above explicitly, we reach an equivalent solution through multiple planning stages.

## 5.3 Methods

In this section, we outline the different methods utilized in solving Problem 5.1.

### 5.3.1 Driver Model Learning

This section discusses the learning component introduced in Section 5.2. One of the major challenges for the proposed problem is that each driver behaves differently. While one driver may drive at a conservative speed limit, another might drive relatively faster, slower, or erratically. Therefore, learning a driver's 'behavior' will be beneficial to the rendezvous problem. Later on, we use this model in the approximation algorithm that estimates future driver position. We now set up this learning problem. We assume that we have access to the driver's position $\theta^{d,i} = \theta^d(t^i)$, where $t^i$ is the time instance at which the measurement is obtained. Furthermore, we have measurements of the driver's velocity denoted by $\dot{\theta}^{d,i} = \dot{\theta}^d(t^i)$. Note that there is no dependency on which path the driver is on because we assume that $\theta^{d,i} \in \mathcal{I}$, and $\mathcal{I}$ ignores non-reachable paths. All measurements are considered to have additive normally distributed noise. We also assume that we have access to historical velocity profile given by $\dot{\theta}^{h,i} = \dot{\theta}^h(t_i)$. Such a historical velocity profile can be generated by collecting measurements of vehicles traversing the path and fitting a distribution over it using methods similar to those in [43, 44]. In our case, we assume the historical velocity profiles are in the form of time-parametrized mean functions. To summarize, given the driver's position $\theta^{d,i}$, we have access to a measurement of the driver's velocity $\dot{\theta}^{d,i}$ and the corresponding probabilistic historical velocity $\dot{\theta}^{h,i}$. A comparison of $\dot{\theta}^{d,i}$ and $\dot{\theta}^{h,i}$ thus represents a measure of the driver's behavior. In particular, we wish to learn $\dot{\theta}^d(\dot{\theta}^h) : \mathbb{R} \to \mathbb{R}$. An equivalent problem is to learn a deviation function $d(\dot{\theta}^h) : \mathbb{R} \to \mathbb{R}$ such that $\dot{\theta}^d(t) = \dot{\theta}^h(t) + d(\dot{\theta}^h(t))$. Throughout this dissertation, we learn the deviation function.

The traditional approach would be to directly learn the vehicle's position

function $\theta^d(t)$; however, this would cause the uncertainty propagation to expand too quickly and force an abort decision too often [19]. Instead, we explore both the fact that the vehicle is constrained to a known path and that the velocity along the path has a strong prior (the historical velocity $\dot{\theta}^h(\cdot)$). A disadvantage of this approach is that an integration procedure must be carried out to estimate $\theta^d(t)$. We leverage the sampling-based nature of this algorithm presented in Subsection 5.3.3 and modern numerical integration methods to provide a computationally efficient integration procedure.

As described by Williams and Rasmussen [46], a Gaussian Process (GP) is a generalization to functions of the Gaussian distribution. Assume that we have a stream of $M \in \mathbb{N}$ measurements of the form $y_i = d(x_i) + \zeta$, $\zeta \sim \mathcal{N}(0, \sigma_n^2)$, $i \in \{1, \ldots, N\}$, where $y_i = x_i - \dot{\theta}^{d,i}$, $\dot{\theta}^{d,i}$ is the actual sensor measurement, and $x_i = \dot{\theta}^{h,i}$ is known a priori. Note that this definition is equivalent as far as learning objectives to the one in Problem 5.1. Now let

$$\mathbf{Y} = \begin{bmatrix} y_1 & \ldots & y_M \end{bmatrix}^\intercal, \ \mathbf{X} = \begin{bmatrix} x_1 & \ldots & x_M \end{bmatrix}^\intercal,$$

and define the data set $\mathcal{D}_M = \{\mathbf{Y}, \mathbf{X}\}$. GPR (Gaussian Process Regression) assumes that

$$y_i \sim \mathcal{N}(d(x_i), \sigma_n^2), \ d \sim \mathcal{N}(0, K_d(x, x'))$$

for a kernel $K_d$. The choice of kernel functions depends on the particulars of the problem. In this dissertation, we use the Matérn Kernel [53]. We can then define posterior distributions at any point $x^\star \in \mathbb{R}$ given $\mathcal{D}_M$ as $d(x^\star)|\mathbf{Y} \sim \mathcal{N}(\mu_d(x^\star), \Sigma_d(x^\star))$. The mean and variance functions $\mu_d(x^\star)$, $\Sigma_d(x^\star)$ of the GP model are defined as

$$\begin{aligned} \mu_d(x^\star) &= \mathbf{K}^\star(x^\star)^\intercal(\mathbf{K} + \sigma_n^2)^{-1}\mathbf{Y}, \\ \Sigma_d(x^\star) &= \mathbf{K}^{\star\star}(x^\star) - \mathbf{K}^\star(x^\star)^\intercal(\mathbf{K} + \sigma_n^2)^{-1}\mathbf{K}^\star)(x^\star). \end{aligned}$$

The terms $\mathbf{K}^{\star\star}(x^\star)$, $\mathbf{K}^\star(x^\star)$ and $\mathbf{K}$ are defined based on the kernel $K_d$ of the GP model: $\mathbf{K}^{\star\star}(x^\star) = K_d(x^\star, x^\star) \in \mathbb{R}$, $\mathbf{K}^\star(x^\star) = K_d(\mathbf{X}, x^\star) \in \mathbb{R}^M$, $\mathbf{K} = K_d(\mathbf{X}, \mathbf{X}) \in \mathbb{R}^{M \times M}$. Extensive further reading on this topic can be found in [46, 47]. One of the main benefits towards the risk-averse efforts in this dissertation of using GPR is that estimates are computed in predictive distributions. The resulting distributions will provide tools for risk assessment

in Sections 5.3.3 and 5.3.4. For analysis purposes we also define

$$\mathcal{O} = \{\varpi \in \mathbb{R} : \min_i x_i \leq \varpi \leq \max_i x_i\}.$$

The set $\mathcal{O}$ is the observed set that tracks which points were measured in the domain of $d$.

One downside of using GPR is computational efficiency. In this dissertation, we mitigate this issue with Sparse Gaussian Processes [46, Sec. 8.4]. In particular, we use Deterministic Training Conditionals (DTC) [54, 55]. Although DTC is not state-of-the-art, in practice and for this problem, in particular, it is not outperformed by other methods while providing non-negligible speedup. Deterministic Training Conditionals work by selecting specific inducing points instead of regressing over the entire data set. There are many approaches for selecting the inducing points; we use equally-spaced data quantiles. Compared to other methods, DTC had the property of being conservative — an important characteristic for this framework. Figure 5.2 compares DTC to full GPR. Figure 5.3 shows a fitting performance comparison between the two methods.

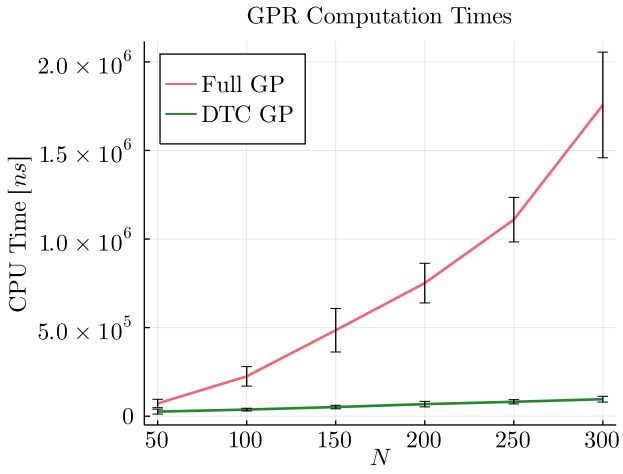



Figure 5.2: Median computation times for full GPR and DTC GPR. Bars represent standard deviation, $N$ indicates the amount of data points. At $N = 300$ a full GP regresses in a median time of 1.555ms, while DTC finishes in 88.827µs. All computations were executed on a single core of a 2012 Intel Core i7.

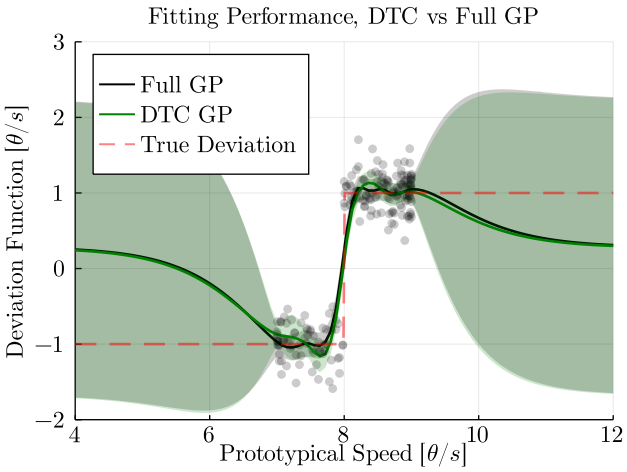



Figure 5.3: Fitting performance: DTC performs effectively the same as a full GP in this application. The goal is to approximate the true deviation function from observed data. Shaded area indicates 95% confidence bounds.

### 5.3.2 MPC formulation

In this section, we discuss the structure and particulars of the MPC component. A primary challenge of the rendezvous problem is presented by strict and numerous constraints, of which many are non-convex. By exploring two unique features of the problem formulation, we reduce dimensionality and attain tractability. We now outline the Optimal Control Problem (OCP) associated with the rendezvous problem. As mentioned previously in Sec. 5.2, the solver is tasked with computing the Point-of-No-Return (PNR) and control inputs that navigate the UAS between important waypoints (rendezvous location, landing location). To fully define the problem and gain temporal constraint management, we expand the control from velocities to include a time "input". The nature of this problem requires the UAS to coincide with the vehicle both in space and time. By introducing time as a manipulated variable in the OCP, we allow the solver to decide on the optimal time allotment before reaching PNR directly. This feature is critical because we rely on maximizing this time allotment (decision time) to increase the number of data points we can gather and subsequently improve the GPR model's quality. This time input works by assuming a piecewise constant control law along each of the four segments (PNR, rendezvous, landing location, and abort location), which is possible due to our assumption on the UAS integrator dynamics described in (5.1) and (5.2).

We represent each of the segments using the state vector $(\mathbf{x}, \mathbf{v}, \mathbf{t}) \equiv (x_i, v_i, t_i)$ with $i \in \{1, \ldots, 4\}$. Here, $t_i$ represents the time to be spent at a constant velocity $v_i$ to reach one waypoint from another. Furthermore, $x_i \in \mathbb{R}^2$ represents each of the defined physical waypoints in Euclidean coordinates and $v_i \in \mathbb{R}^2$ represents velocity inputs in Euclidean coordinates. The waypoints are, in this order, the Point-of-no-Return (PNR), the rendezvous location (RDV), the landing location ($S_L$), and the abort location ($S_A$), as shown in Figure 4.1. The designed Optimal Control Problem (OCP) is given by:

$$
\begin{aligned}
\min_{U} \quad & t_2 + t_3 + t_4 - t_1 \\
\text{s.t.} \quad & x_i = x_{i-1} + v_i t_i, \; x_4 = x_1 + v_4 t_4, \\
& E_{r,k} = E_{r,k-1} - \left( \frac{m_k v^2}{2} + \alpha m_k \right) t_k, \\
& E_{r,4} = E_{r,1} - \left( \frac{m_4 v^2}{2} + \alpha m_4 \right) t_4, \\
& |v_i| \leq v_{\max}, \; x_3 = p^\star, \; x_4 = S_L, \; x_5 = S_A, \\
& \sum_{i=1}^{3} t_i \leq t_{\max}, \; t_1 + t_4 \leq t_{\max}, \; t_c \leq t_i \\
& E_1 + E_2 + E_3 \leq E_{r,k}, \; E_1 + E_4 \leq E_{r,k},
\end{aligned}
$$

where $t_R \equiv t_1 + t_2$ is a provided rendezvous time, $m_1 = m_2 = m_4 = m_a$ is the UAS mass with the package, $m_3 = m_b$ is the UAS mass without the package, $p_{\text{tgt}} \in \mathcal{P}$ is target path we aim to rendezvous with, $p^\star$ is the optimal rendezvous location provided by the GPR model in Section 5.3.1 and elite samples from the importance sampling algorithm in Section 5.3.3, $S_L$ and $S_A$ are the landing and abort destinations, $E_{r,0}$ is the remaining energy, $E_{r,i}$, $i = 1, \ldots, 4$ is the energy associated with the segment, and $t_c$ is a dwell time for the low level controller to switch tracked segments. The dwell time is necessary to stop the solver from placing waypoints arbitrarily close to each other and creating undesirable sharp turns, which are problematic for our single integrator dynamics assumption. Moreover, $U \equiv \{t_i, v_i\}$, $i \in (1, ..., 4)$. Note that apart from risk-related design choices, all constraints and constants are given a priori from mission parameters.

### 5.3.3 Importance Sampling

In this section, we discuss the Sampling-based optimization algorithm. The goal is to approximate $\mathcal{A}^\star$, an $N$-dimensional distribution of optimal rendezvous times according to some criteria. In possession of a rendezvous time, we use the model found in Subsection 5.3.1 to estimate where the rendezvous location is. We perform this optimization problem using the cross-entropy (CE) algorithm [56]. We now set up this optimization problem. We present this algorithm in three parts; first, we discuss the ranking system that selects the best samples in a group; second, we add a risk-averse component; and finally, we briefly show how to update the sampling parameters.

#### 5.3.3.1 Sample Ranking

Let $\mathcal{A}(\mu_\mathcal{A}, \Sigma_\mathcal{A})$ be a $N$-dimensional Gaussian distribution of rendezvous times with mean vector $\mu_\mathcal{A}$ and diagonal variance matrix $\Sigma_\mathcal{A}$. Let $n_s$ and $n_e$ be positive integers such that $n_s >> n_e$. Let $\mathcal{S} \in \mathbb{R}^{N \times n_s}$ be a matrix of $n_s$ samples from $\mathcal{A}$. We say that $\mathcal{S}$ is the sample set, and $\mathcal{S}_e \in \mathbb{R}^{N \times n_e}$ is the elite sample set that contains the $n_e$ best row-wise samples from $\mathcal{S}$. In other words, $\mathcal{S}_e$ is a matrix where each row contains the $n_e$ best samples for the path associated with that row. To find the elite set, we partially rank the sample set according to a a cost function $l(n^{i,j})$ we wish to minimize, where $n^{i,j}$ is the element in the $i$-th row and $j$-th column of the (non-ordered) set $\mathcal{S}$. Each element $n^{i,j}$ is a time sample that produces a rendezvous location candidate. The first step is to compute the expected driver position for each sample as a rendezvous location. Consider a moment in time $t_0$ and a time sample $t^{i,j} \in \mathcal{S}$. At $t_0$ we have a GPR model of $d(\dot{\theta}_i^h(t))$ for every path $i \in \mathcal{P}$. Thus

$$\mathbb{E}[\theta_i^d(t)] = \theta^d(t_0) + \int_{t_0}^{t^{i,j}} \dot{\theta}_i^h(t) + d(\dot{\theta}_i^h(t))\,\mathrm{d}t, \tag{5.4}$$

where the integration procedure is done numerically using Gauss-Kronrod Quadrature. This integration completes in a median time of 124.4µs on a single core of a 2012 Intel Core i7. However, the nature of the algorithm permits this implementation to be largely computed in parallel on a GPU, although such implementation is not done in this dissertation. A parallel implementation would provide significant speedup for higher values of $n_s$ than

the ones used in this dissertation (we use $n_s = 5$, for reference). Using (5.4) we can compute the expected driver position $\mathbf{p}$ for each time sample $j$ and path $i$ with $p_i(\mathbb{E}[\theta_i^d(t_j)])$ in matrix form:

$$\mathbf{p} = \begin{bmatrix} p_1(\mathbb{E}[\theta_1^d(t_1)]) & \cdots & p_1(\mathbb{E}[\theta_1^d(t_{n_s})]) \\ \vdots & & \vdots \\ p_N(\mathbb{E}[\theta_N^d(t_1)]) & \cdots & p_N(\mathbb{E}[\theta_N^d(t_{n_s})]) \end{bmatrix}.$$

In possession of $\mathbf{p}$ we can compute the "quality" of each sample. Naturally, because our goal is to not run out of fuel or battery, we choose samples that minimize energy consumption. This is not equivalent to minimizing distance to the UAS for two reasons: (a) spatial points have a temporal constraint, and (b) the landing location and remaining fuel for landing depend on an external path planner (discussed in Subsection 5.3.2). Temporal constraints mean that two equally close rendezvous candidates have two different times for the UAS to reach that location. The energy dynamics (5.2) are such that the best rendezvous time is non-obvious in this case. Additionally, the landing location and remaining fuel after rendezvous need to be included in the quality criteria. Failure to do so would select a sample that minimizes the energy necessary to rendezvous, but might not minimize overall mission energy. To compute energy costs for each sample we use the energy dynamics (5.2). Let $\mathbf{E} \in \mathbb{R}^{N \times n_s}$ be a matrix containing the energy costs for each sample, and $x_0 = [x_0^1, x_0^2]$ be the euclidean position of the UAS at $t_0$. Then we compute each element of $\mathbf{E}$ as

$$\mathbf{E}^{i,j} = m_a(t_j - t_0)\left[\frac{1}{2}\left\|v_r^{i,j}\right\|_2^2 + \alpha\right] + m_b(t_l - t_j)\left[\frac{1}{2}\left\|v_l^{i,j}\right\|_2^2 + \alpha\right], \tag{5.5}$$

where $t_l = \sum_{k=1}^{3} t_k$ is provided by the MPC solution in Section 5.3.2 (in the absence of a solution in the first iteration we do not compute the second term of (5.5)), $\|\cdot\|_2$ denotes the 2-norm of a vector, and

$$v_r^{i,j} = \frac{|\mathbf{p}^{i,j} - x_0|}{t_j - t_0}, \quad v_l^{i,j} = \frac{|S_L - \mathbf{p}^{i,j}|}{t_l - t_j}, \tag{5.6}$$

where $S_L$ is the landing location as described in Section 5.3.2. For this dissertation, we set $l(n^{i,j}) = \mathbf{E}^{i,j}$. In Subsection 5.3.3.2 we add a risk-averse component to the calculation of $\mathbf{E}$. These equations reflect the (generic) en-

ergy dynamics we consider in this dissertation, but the procedure is agnostic to the energy dynamics chosen by the designer. To find $\mathcal{S}_e$ we select the $n_e$ best samples for each path:

$$\mathcal{S}_e = \begin{bmatrix} \arg\min_j^{n_e} l(n^{1,j}) \\ \vdots \\ \arg\min_j^{n_e} l(n^{N,j}) \end{bmatrix},$$

where $\arg\min_x^k f(x, \dots)$ means we select the $k$-best arguments that minimize $f$ over $x$ that we output in the order from best to worst. This way the first column of $\mathcal{S}_e$ contains the best sample for each path and so on for the other columns. Using similar logic, we select the target optimal rendezvous location $p^\star = p_{\text{tgt}}(\theta^d_{\text{tgt}}(t_{R,\text{tgt}}))$ to be forwarded to the MPC planner. Let $S_e^{i,1}$ be the first column of $S_e$. Then for $\text{tgt} \in \{1, \dots, N\}$ let $\text{tgt} = \arg\min_i c\left(\mathcal{S}_e^{i,1}\right)$. This equation applies a cost function to the best samples from each path and selects the best path index based on that cost function. The design of $c$ is intricate, and an in-depth analysis of what constitutes a suitable cost function is left as future work. We present two naive designs in this dissertation.

The most natural function one could choose is based on the *Best First* logic. This logic selects the best global sample, and we consider the path associated with that sample to be the optimal one to rendezvous with. This is equivalent to simply setting $c(\mathcal{S}_e^{i,1}) = \mathcal{S}_e^{i,1}$, i.e. select the time sample of least energy. This strategy would be desirable if it is possible to control which path the driver will choose. In a different framework application where we would consider autonomous vehicles as the ground agent, such a cost function is highly desirable. In this dissertation, however, selecting the best global sample is overly ambitious. If the driver chooses any path other than $p_{\text{tgt}}$, there is a non-negligible probability that there will not be enough battery to alter the course since the MPC will be spending resources to maximize $t_1$. An alternative version of Best First applies weights to the cost function with $c(\mathcal{S}_e^{i,1}) = w_i \mathcal{S}_e^{i,1}$, where $w_i \in \mathbb{R}_+^N$ is a vector of user-defined weights. This variant is beneficial if the designer has prior knowledge of the driver's probability of choosing each path.

In this dissertation, we use the opposite strategy; *Worst First*. This strategy selects the worst-of-the-best time samples, i.e., from the best samples for each path, selects the worst path. The logic is simple: if Best First is an ambitious

strategy, Worst First is a conservative one. If we plan to have enough energy to reach the worst path for optimal rendezvous, all others require less energy and, thus, are reachable. This strategy assumes $c(\mathcal{S}_e^{i,1}) = -w_i\mathcal{S}_e^{i,1}$, where we can again weigh each entry according to some path choice distribution. For the rest of this dissertation, we assume a uniform path choice distribution such that $w_i = 1 \forall i$.On an implementation note, we use the partial quicksort algorithm in [57] to quickly partially sort the array of samples. Partial quicksort is more efficient than naive approaches since it only returns the top-$k$ elements of an array. The partial sorting procedure finishes on a median time of 2.182µs on a single core of a 2012 Intel Core i7. Since all entries are independent, this procedure is both vectorized and parallelized so that CPU time scaling is better than linear.

#### 5.3.3.2 Risk Assessment

Quantifying risk is the effort to determine a measure $\rho$ that maps a set of random variables to a real number representing the probability or expected value of an undesirable outcome [22, 21]. With this definition, the random variables are the states of the UAS (due to process and measurement noises) and, more importantly, the position of the ground vehicle due to the driver's uncertain behavior. It is crucial to choose measures that reflect meaningful quantities in the problem formulation. In this framework, risk directly relates to uncertainty regarding the vehicle's location in the future and the limitations that the path imposes on planning. If the driver is erratic, or the path only allows the rendezvous to happen in unfavorable locations, we consider that the mission has elevated risk. Several risk measures are popular; some examples are Expectation-Variance [23], (Conditional, Tail) Value-at-Risk [21, Sec. 3.3], and Downside Variance [21, Sec. 3.2.7]. These measures can introduce nonlinearity and preclude gradient information, endangering tractability. A popular approach uses gradient-free methods, which sample these measures and choose inputs corresponding to minimum risk [49]. In this dissertation, we use two risk measures; $\rho_r$ is the rendezvous risk measure used by the rating system to select samples of least risk, and $\rho_d$ is the decision risk measure used in Subsection 5.3.4 to decide on whether to abort the mission and safely return, or proceed with the rendezvous. In this subsection, we discuss the design and implementation of $\rho_r$. The main differentiator between the two is that the

rendezvous risk measure needs to be computationally efficient since we repeat its calculations for every sample, every time step. We embed risk into the cost by adjusting the distance between each sample and the UAS or landing area in the numerator of Eq. (5.6).

We start by computing the propagated uncertainty for each sample $n^{i,j}$ as

$$h^{i,j} = \gamma \int_{t_0}^{n^{i,j}} \Sigma_{d,i}(\dot{\theta}_i^h(t))\,\mathrm{d}t, \tag{5.7}$$

where $\gamma \in \mathbb{R}^+$ is a scaling factor. Equation (5.7) makes it so that $p_i(\mathbb{E}[\theta_i^d(t_j)]) \pm h^{i,j}$ represents a confidence interval in $\theta$. The goal now is to select which of these three numbers is furthest from the UAS or landing location, and use that distance when computing necessary velocities. Let $\Gamma(a,b) : \mathbb{R}^2 \times \mathbb{R}^2 \mapsto \mathbb{R}$ be the Euclidean distance between two points $a, b \in \mathcal{R}$, then for every sample $n^{i,j}$ let $r^{i,j} = \Gamma(\mathbf{p}^{i,j}, x_0)$ be the neutral range, $r_+^{i,j} = \Gamma(p_i(\mathbb{E}[\theta_i^d(t_j)]) + h^{i,j}, x_0)$ be the positive uncertainty range, and $r_-^{i,j} = \Gamma(p_i(\mathbb{E}[\theta_i^d(t_j)]) - h^{i,j}, x_0)$ be the negative uncertainty range. Then $\rho_r$ naturally follows: $\rho_r^{i,j}(\Sigma_d, \mathcal{S}, x_0) = \max(r^{i,j}, r_+^{i,j}, r_-^{i,j}) - r^{i,j}$.

Figure 5.4 shows a visual representation of the different ranges. This risk measure is an approximation of Conditional Value-at-Risk (CVaR). In its common form, $\mathrm{CVaR}_{1-\gamma}$ represents the expected value of the $\gamma$-percentile of a distribution that quantifies potential loss (downside potential). Here, instead of computing the energy distribution, we compare the energy required to reach the sample at its mean and at some $\sigma$-distance away from the mean. We then pick the worst outcome and say this is the potential loss for this sample. Finally, we can compute the risk-enabled velocities with

$$v_r^{i,j} = \frac{r^{i,j} + \rho_r^{i,j}(\Sigma_d, \mathcal{S}, x_0)}{t_j - t_0},$$
$$v_l^{i,j} = \frac{r^{i,j} + \rho_r^{i,j}(\Sigma_d, \mathcal{S}, S_L)}{t_l - t_j},$$

and compute and rank $\mathbf{E}$ the same way as before.

#### 5.3.3.3 Parameter Update

This section discusses the parameter update algorithm for a single path. Since all paths are independent, we repeat this process identically for every path.

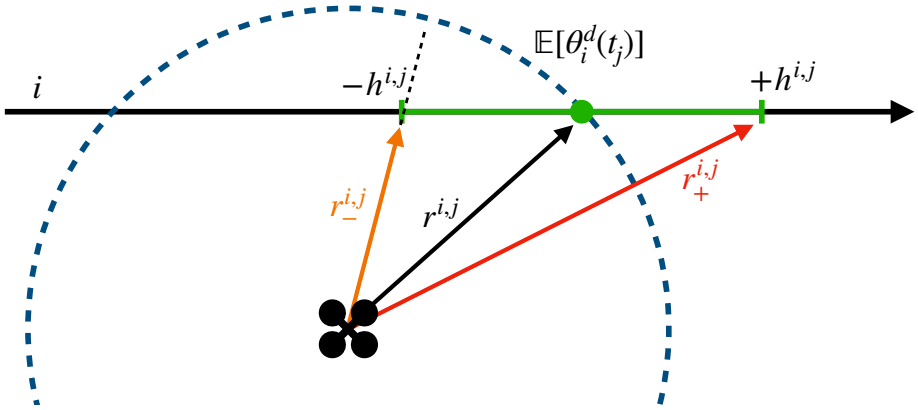



Figure 5.4: Downside Risk as potential required range gain. The red outcome forces the UAS to spend more energy to meet the car. The extra energy is the downside potential, used as risk measure.

We update $\mu$ and $\Sigma$ by taking mean and variance row-wise from $\mathcal{S}_e$ with $\mu_{\mathcal{A}} = \text{Mean}(\mathcal{S}_e)$ and $\Sigma_{\mathcal{A}} = \text{Var}(\mathcal{S}_e) + \lambda$, where $\lambda \in \mathbb{R}$ is a small positive number. The scalar parameter $\lambda$ serves as an exploration tool due to the time-varying nature of the algorithm and avoids convergence to a (traditionally desirable) static impulse-like distribution.

### 5.3.4 Heuristics

In this section, we discuss the architecture of the overarching algorithm that integrates all modules and commands a rendezvous/abort decision. In summa, this is shown in Algorithm 3. The functions in Algorithm 3 and their correlated method are shown in Table 5.1.

The overarching logic is the same as discussed in Section 5.2. While the decision time $t_1$ (time before reaching the PNR waypoint) is greater than some constant, we keep acquiring data, improving the model, and searching for a better rendezvous point. When a decision is necessary we perform a one-time risk analysis and comparison against the scalar constant $\kappa \in \mathbb{R}$ to decide between turning back or continuing. Note that following Definition 4.1, $\mathbf{x}_k \in \mathcal{S}_k \forall k$ because if this condition is violated, we switch to the plan with deterministic guaranteed safety.

As discussed in Section 5.3.3, for online optimization purposes we approximate $\text{CVaR}_{1-\gamma}$. When deciding whether to abort or attempt a rendezvous,

**Algorithm 3:** Mission Algorithm

$\mathcal{D} \leftarrow$ Initial Data
**while** $t_1 > \varepsilon$ **do**
  $d,\ \Sigma_d \leftarrow \text{Regress}(\mathcal{D})$
  $\mathcal{S} \leftarrow \text{Sample}(\mathcal{A}, N_s)$
  $p^\star,\ \mathcal{S}_e \leftarrow \text{Rank}(\mathcal{S}, \mathbf{x}, \mathbf{t}, d, \Sigma_d)$
  $\mu_{\mathcal{A}},\ \Sigma_{\mathcal{A}} \leftarrow \text{UpdateParameter}(\mathcal{S}_e)$
  $\mathbf{v}, \mathbf{x}, \mathbf{t} \leftarrow \text{MPC}(p^\star, \mathbf{x})$
  Send Control Input $\mathbf{v}$ to UAS
  $\mathcal{D} \leftarrow \text{Append}(\text{Sensor Data}, \mathcal{D})$
**end**
**if** $\rho_d(d, \Sigma_d, \mathbf{x}) \leq \kappa$ **then**
  Proceed with rendezvous and then to $S_L$
**else**
  Abort and return to $S_L$
**end**

| Function Name | Procedure |
|---|---|
| Regress | GPR in Sec. 5.3.1 |
| Sample | Returns $N_s$ samples from $\mathcal{A}$ |
| Rank | Ranking Procedure in Sec. 5.3.3 |
| UpdateParemeter | Updates parameters of $\mathcal{A}$ as in Sec. 5.3.3.3 |
| MPC | Computes MPC control inputs as in Sec. 5.3.2 |
| Append | Appends new sensor data to $\mathcal{D}$ |

Table 5.1: Correlation between Algorithm 3 and methods in Section 5.3.

however, we can afford an expensive one-time computation of $\text{CVaR}_{1-\gamma}$. We briefly define $\text{CVaR}_{1-\gamma}$ for completion. Let $\mathbf{Z}$ be a set of real-valued continuous random variables, $\rho : \mathbf{Z} \mapsto \mathbb{R}$ be a risk measure function, $\mathbf{X} \in \mathbf{Z}$ be a random variable, $x \in \Omega_{\mathbf{X}}$ be the domain of $\mathbf{X}$, $f_{\mathbf{X}}(x)$ be the probability density function of $\mathbf{X}$, $F_{\mathbf{X}}(x)$ be the cumulative density function of $\mathbf{X}$. Then we define the two quantities of interest: $\mathbf{VaR}_\gamma(\mathbf{X}) = \inf\{x \in \Omega_{\mathbf{X}} : F_{\mathbf{X}}(x) \geq \gamma\}$ and

$$\text{CVaR}_{1-\gamma}(\mathbf{X}) = \frac{1}{\gamma} \int_0^\gamma \mathbf{VaR}_\gamma(\mathbf{X})\, \mathrm{d}\gamma,$$

where $\gamma \in [0, 1]$ is a real-valued quantile. For the purposes of this dissertation, we consider the the distribution of extra fuel required for rendezvous ($\mathbf{X} = E_e = E_r - (E_1 + E_2 + E_3)$) as the random variable, with distribution derived from $\mu_d$ and $\Sigma_d$ for each of the paths' optimal rendezvous location.

## 5.4 Results

In this section, we present the results of all modules. Implementation code that generates all figures and animations of results is available at `https://github.com/gbarsih/Multi-Path-Safe-Rendezvous`.

### 5.4.1 Learning Performance

We start by presenting results on the learning algorithm in Section 5.3.1. The learning problem seeks to find a deviation function $d$ that captures the driver behavior. Figures 5.5 and 5.6 show two points in time. Figure 5.5 is taken after 10s of data gathering, and Figure 5.6 after 50s. The nature of $\dot{\theta}^h$ ensures that in Figure 5.5, $\mathcal{O}$ is limited to only under half of possible values $\dot{\theta}^h$ can take. Consequently, the GPR model has no information on that region and produces a high value for projected uncertainty. Conversely, in Figure 5.6 we explored the entire domain, and the GPR model can produce estimates with higher certainty. For the remaining results, we use the same functions shown here for all paths, where $\dot{\theta}^h(t) = 8+\sin(t/10)$ and $\dot{\theta}^d(t) = \dot{\theta}^h(t)+\text{sign}(\dot{\theta}^h(t)-8)$.

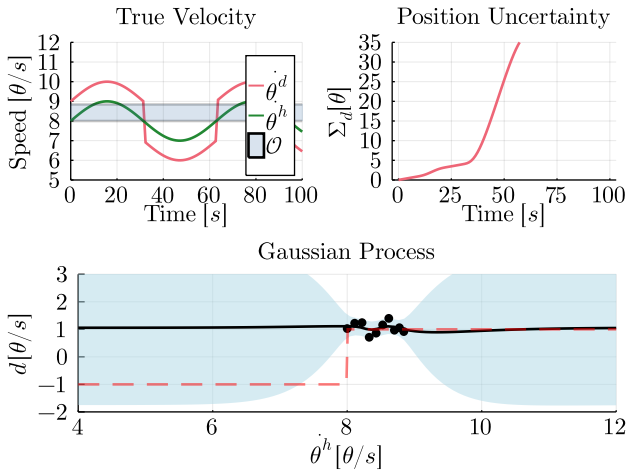



Figure 5.5: GPR learning process snapshot after 10s. Shaded area indicates 95% confidence bounds.

### 5.4.2 Importance Sampling

Here, we present results on the importance sampling algorithm in Section 5.3.3. Since an analytical form of the optimal rendezvous location does not exist, we

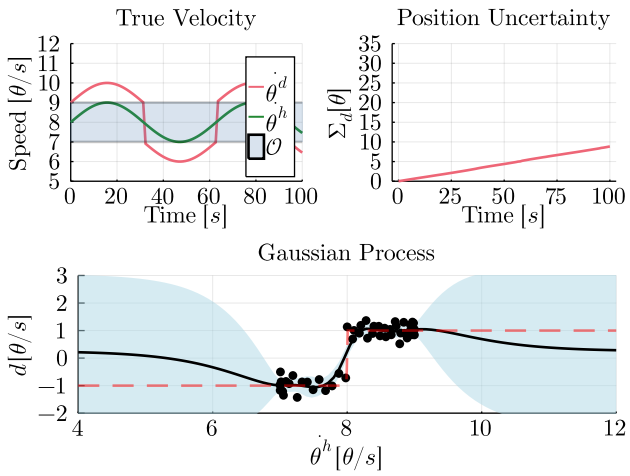



Figure 5.6: GPR learning process snapshot after 50s.

leave the performance quantification for the full planning stack results in the next subsection. To show the efficacy of this individual module, we analyze the convergence rate. Figure 5.7 shows 100 different runs of the same algorithm, with mission parameters randomly selected. We notice that it quickly converges (in about four iterations), and that when $\mathcal{O}$ begins to expand at $t = 10\pi$, the algorithm increases $\Sigma_{\mathcal{A}}$ to optimize over the new data.

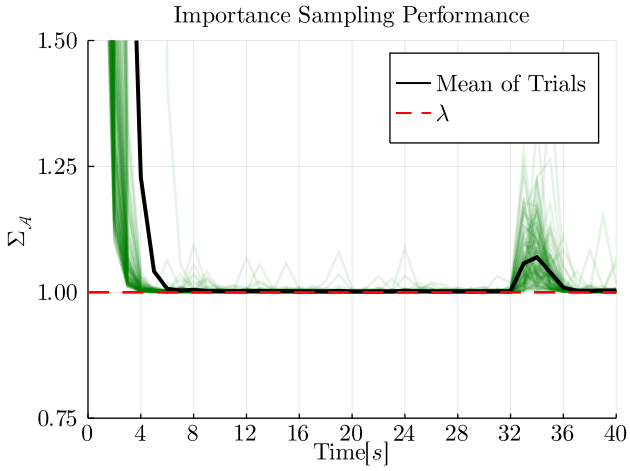



Figure 5.7: Average convergence rate (black) of $\Sigma_{\mathcal{A}}$ for 100 trials (green). At $t = 10\pi$s, $\mathcal{O}$ starts to cover new information. The shift in the learned driver behavior causes the sampling algorithm to react as indicated by the momentary increase in $\Sigma_{\mathcal{A}}$.

### 5.4.3 Full Planning Stack

Consider the map in Figure 5.8. With $m = [3, 1]$kg the maximum range assuming no drop-off is 335m, while the maximum range assuming a successful drop-off is 485m, with $\alpha = 20$ and $E_{r,0} = 1.6\text{E}4$. Under these conditions, the roads are only reachable with the algorithm presented in this dissertation. Figure 5.9 compares the risk associated with the secondary path (that is, the path that is not $p_{\text{tgt}}$) for the two importance sampling strategies. As expected, using Worst First yields reduced risk. The choice of $\kappa$ is entirely dependent on the mission parameters and how much risk the designer is willing to take; however, using the Worst First strategy will, in general, attempt a rendezvous more often. Note that which path the driver chooses after a decision is made, is irrelevant here since risk assessment is performed for all cases. The algorithm aims to ensure that risk is low for all possible outcomes. Finally, Figure 5.10 depicts the planned energies throughout the mission and the distance from the UAS to the driver. Both plans (abort and rendezvous) are maximized and feasible.

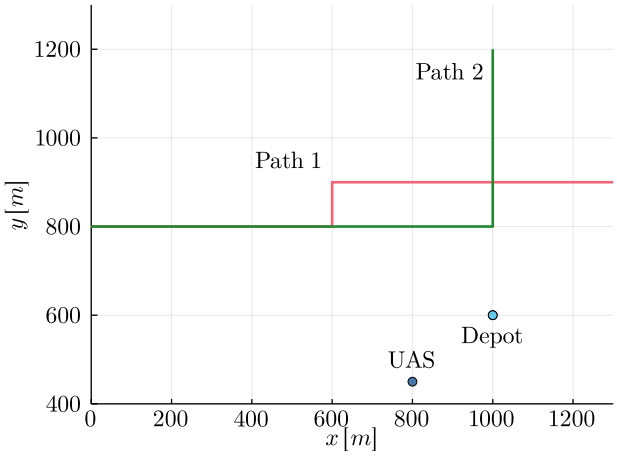



Figure 5.8: Mission map representing a section of an urban grid.

## 5.5 Conclusions

We presented an algorithm capable of planning and executing a rendezvous mission between an autonomous UAS and a human-operated ground vehicle. The planner can assess the risk associated with the human factor and make

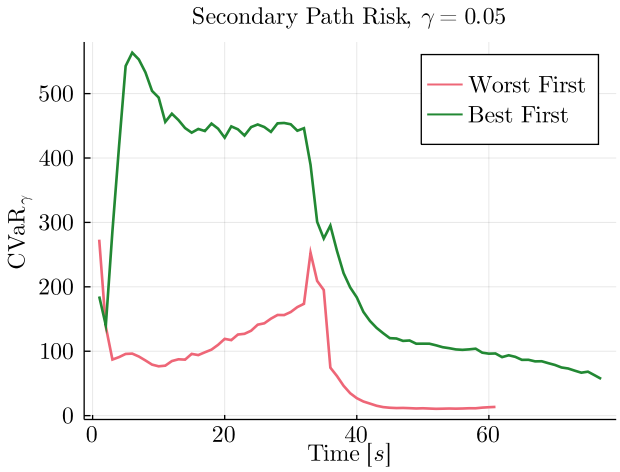



Figure 5.9: Same mission parameters, equally seeded, for the two strategies proposed in Section 5.3.3. Plot terminates when $t_1 < \varepsilon = 1$. Worst First finds trajectories with least risk should the driver choose a path $p \in \mathcal{P} \setminus p_{\text{tgt}}$.

informed decisions on either proceeding with the rendezvous or flying to a safe landing location. Such an arrangement is persistently safe because the abort plan is deterministic. We show numerically that the approach accomplishes its goals. For future work, we intend to address two deficiencies of this method. First, the algorithm needs to account for multiple drivers. There are untapped benefits of having multiple rendezvous options at any given time. Second, we wish to model the dynamics of drivers entering the network to preemptively start a mission and improve system efficiency.

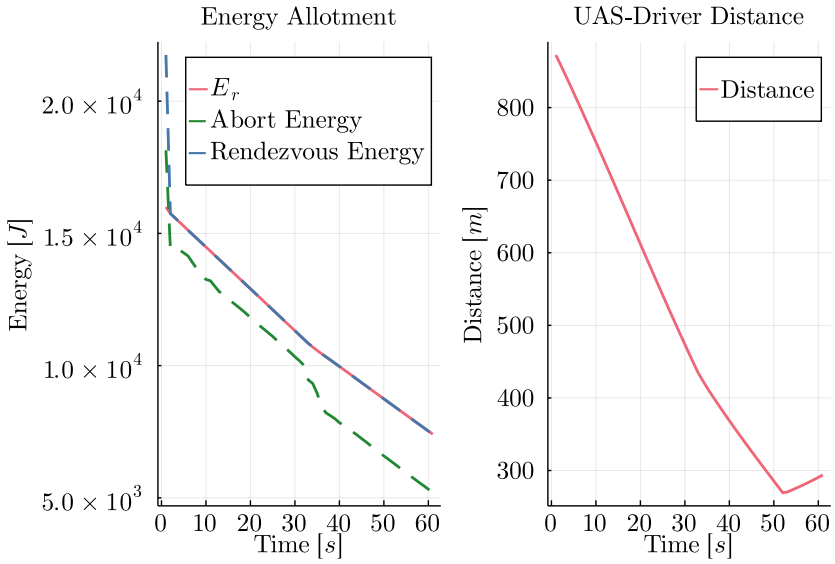



Figure 5.10: Results using Worst First strategy. The algorithm uses all available energy to try and minimize risk. Distance increase towards the end is due to path geometry.

# CHAPTER 6

# LEARNING SAFE PATHS AT LARGE SCALES

In this chapter, we aim to estimate risks associated with an urban GPS route as perceived by a rendezvous motion planning algorithm. In Chapters 4 and 5 we established that a novel motion planning algorithm can extend the useful range of UASs by estimating the risk of rendezvous failure and committing accordingly. We showed that given a certain driver it is possible to build a behavior model during flight and perform the rendezvous. The inherent risk of such a mission lies in uncertain driver behavior both in speed and path choice. A driver who makes a lot of mistakes or is too fast or slow when compared to a nominal trajectory is risky because there is a high probability of spending more battery than previously planned, which can cause a crash on the way back. In our contribution, we provide the algorithmic foundation to select this target driver.

In a dense urban scenario, we expect to have multiple drivers available to receive a package. Figure 6.1 shows the Downtown Chicago area and several vehicles with prescribed nominal routes. In this context, we are concerned with the epistemic uncertainty of coupling between human behavior and the street layout of the area. As discussed in [58, 59], drivers often deviate from the optimal route due to a variety of factors, including road conditions, familiarity, preferences, road closures, accidents, and mistakes (due to lack of attention or wrong directions). The authors also found that drivers who do not follow the route tend to make their trips longer, regardless of the perceived superiority of their choice.

From the perspective of motion planning, we focus on finding an estimate for spatial and temporal deviation from the prescribed route that we can map to risk. This estimate should also rely on limited information about these drivers; only their current position and destination. A good estimator will infer their route from this data; this also means any learned model will not be transferable to other regions or even other depots in that region. This model

can then be used to quickly sort through the drivers and perform UAS-driver assignments in real-time.

## 6.1 Preliminaries and Problem Formulation

We are given a directional graph $\mathcal{G} = (V, E, w)$ with node set $V$, edge set $E$, and edge-weight function $w : E \to \mathbb{R}_{\geq 0}$.This strongly-connected graph represents streets on an urban area, so every node has a unique label and *stores* a pair of coordinates in the Geographic Coordinate System (latitude and longitude). For a node $v$, its pair of coordinates is $c_v = (\text{lat}(v), \text{lon}(v))$. Let $\mathbf{v} = (v_1, \ldots, v_N),\ v_i \subset V$, be a sequence of nodes such that $\deg\text{sep}(v_i, v_{i+1}) = 1,\ i \in [1, N-1]$, where $\deg\text{sep}(\cdot, \cdot)$ indicates the degree of separation between two nodes. We say that the set $\Delta_i$ is the set of all neighboring nodes of node $i \in V$:

$$\Delta_i = \{j \in V : \deg\text{sep}(i, j) = 1\}.$$

Any sequence built this way is a valid route. Additionally, let $S$ be set of all subsequences of $\mathbf{v}$. To model particles (drivers) traversing this graph, we create a probability-weighted adjacency matrix, effectively building a discrete-time Markov chain (DTMC). First, we define the deterministic case where the driver never deviates from the prescribed route. Let $P_d$ be a transition matrix of a DTMC where each element $p_{ij}$ represents the transition probability from node $i$ to node $j$:

$$p_{ij} = \begin{cases} 1,\ (i,j) \in S,\ i = j = N,\ \text{or } i = j \notin \mathbf{v}, \\ 0,\ \text{otherwise.} \end{cases}$$

The deterministic version admits a stationary distribution $\pi$ if and only if the initial state is distributed along the states $\mathbf{v}$. This Markov chain is reducible with the reduced version of $P$ being transient for all states but $v_N$. Assume now that we want to allow for drivers to randomly jump to a node $j \notin \mathbf{v}$ from a node $i \in \mathbf{v}$ such that $\deg\text{sep}(i, j) = 1$. That is, we wish to have a non-zero probability of transitioning off the prescribed route to neighboring nodes (note that *nodes* is used when referring to $\mathcal{G}$, but *states* when referring to transition matrices).

We define the path-stochastic model with a DTMC transition matrix $P$.

We define $P$ with a shortest-path tree $T_g$ to a goal node $g \in V$. This tree is rooted at $g$ and spans to contain all nodes in the graph $G$ such that every node points to the next node along the shortest path to $g$, and $g$ points to itself. Since every node in $T_g$ only points to one other node, it is also a mapping $T_g : V \mapsto V$. In practice, a node $v_1$ is contained in $T_{v_N}$ and it would point to $v_2$, then $v_3$, and so on, constructing the same sequence $\mathbf{v}$, or $T_{v_N}(v_i) = v_{i+1}$, $i \in [1, N-1]$.

We can now define the path-stochastic version of $P$ for a goal node $g$:

$$p_{ij} = \begin{cases} 1-p, \text{ if } j = T_g(i), \ |\Delta_i| > 2, \\ \dfrac{p}{|\Delta_i| - 2}, \text{ if } j \in \Delta_i, \ |\Delta_i| > 2, \\ 1, \text{ otherwise.} \end{cases}$$

The above definition simply indicates that if a node is connected to more nodes than the next and previous node, then there is a probability $p \in [0, 1]$ to switch to an adjacent node. In this context, $p$ is relatively small. Studies [58, 59] have found that deviations occur in anywhere from 30% to 60% of routes, making the original shortest path between two nodes the most likely one to be followed. For now, we assume that this probability is uniform throughout the nodes and that adjacent nodes have an equal probability of being chosen. This assumption is unrealistic, however, it does not change the analysis. Figure 6.2 shows an example in Downtown Chicago with hypothetical errors. Notice how some errors are minor, while others might redefine the entire route.

Due to the stochastic nature of this model, only two quantities are certain: the initial and goal nodes. We now define the problem.

**Problem Formulation 6.1 (Risk Estimation for Markov Drivers)** *Given a weighted graph $\mathcal{G} = (V, E, w)$ and depot node $v_d$, find an approximation $\hat{\rho}$ of the risk measure $\rho : V \times V \mapsto \mathbb{R}^+$ that represents the risk associated with a rendezvous mission [60].*

Problem 6.1 will be solved with a learning algorithm and appropriate loss function. The main challenge ahead is data specification and the synthesis of the loss variable. Figure 6.3 depicts several drivers simulated with this model in the Downtown Chicago area for a single origin/destination pair. Notice that some deviations are minor and we expect these to not cause many problems, while others are drastic and their effect on a rendezvous mission is unknown.

Another important factor is the spatial structure of the graph; in Figure 6.3 all paths collapse on the main highway and do not escape for some time. This is expected; highways are often the quickest route, and there are not many opportunities to make a mistake or choose a different route while on one. The following sections will outline the learning scheme, first by defining the data inputs, then discussing neural network topology.

## 6.2 Methods

This section outlines methods related to the numerical nature of the problem and how it relates to data and the machine learning approach to approximating risk measures.

### 6.2.1 Brute Force Approach

This section describes procedural data generation for finding $\hat{\rho}$. The data necessary to solve Problem 6.1 is not readily available. The only public data we have access to are the street maps and traffic flows. In a production environment, the data we are about to describe would be captured from real drivers, a process we discuss later in this paper. Several approaches could be considered when finding a solution to Problem 6.1. Recall that we are given origin and destination nodes for many drivers, and we wish to use $\hat{\rho}$ to quickly sort through these drivers and find the safest ones. This is equivalent to having a lookup table containing all possible pair permutations of $V$ and their associated risk $\rho$. A naive solution would be to build this lookup table in its entirety. This is an NP-hard problem based solely on the necessity of computing risk for every single origin/destination pair permutation in $V$. However, describing the procedure is useful for later parts of this section. We begin by describing the brute-force approach to solving Problem 6.1 for a given graph $\mathcal{G}$.

Consider a list $\mathcal{L}$ of length $|V| \cdot (|V| - 1)$ containing all pair permutations of nodes $V$ in a graph $\mathcal{G} = (V, E, w)$. Additionally, consider an element $i \in \mathcal{L}$ consisting of nodes $u$ and $g$ (in this order, defining a trip from $u$ to $g$). A single agent traverses the graph through the transition matrix $P$; from this traversal we store the sequence of nodes visited and the respective time on which the visit occurred. The size and contents of this sequence are stochastic. The

traversal function $\mathcal{T}(u, g, p) : V \times V \times \mathcal{U} \mapsto V^N \times \mathbb{R}^N$ simulates a traversal from node $u$ to node $g$ with jump probability $p \in \mathcal{U} = [0, 1]$ and returns the aforementioned sequences.

For a given node sequence $\mathbf{v} = v_1, \ldots, v_N$, with associated time sequence $\mathbf{t} = t_1, \ldots, t_N$, we compute the effort necessary to reach each node in space and time with $\mathbf{E} = E_1, \ldots, E_N = f_E(\mathbf{v}, \mathbf{t}, v_d)$. This function can be arbitrary and represents how the UAS consumes resources (energy) to fly. Throughout the framework we use

$$f_E(\mathbf{v}, \mathbf{t}, v_d) = m \cdot \left( \frac{d_G(\mathbf{v}, v_d)^2}{\mathbf{t}} + \alpha \mathbf{t} \right), \tag{6.1}$$

which is quadratic with respect to velocity. Additionally, $m$ represents the mass of the UAS and payload combined, $\alpha \in \mathbb{R}^+$ is a hovering cost, and $d_G(\mathbf{v}, v_d)$ is the great-circle distance between the coordinates of each element in $\mathbf{v}$ and the coordinates of the depot in node $v_d$. Equation (6.1) is evaluated element-wise for every element in the sequence and returns an appropriately sized sequence of energies $\mathbf{E}$. It is then trivial to find the node of minimum energy $v^\star \in \mathbf{v}$ with a linear search:

$$v^\star = \arg\min_{\mathbf{v}} \mathbf{E},$$

with minimum

$$E^\star = f_E(v^\star, t^\star, v_d).$$

Note that since it is tied to $\mathbf{v}$ and for ease of notation we do not specify $\mathbf{t}$ as an optimization variable.

The process to approximate $\rho$ now becomes clear. In a Monte-Carlo fashion we simulate a large number $M$ of cars travelling from $u$ to $g$ using $\mathcal{T}(u, g, p)$ and collect $E^\star$ for each run. We select $p$ so that the expected number of deviations per trip is one. This does not match Samson's findings [59], but here we have a different goal: we wish to estimate the gravity of deviations along some route, not their rate of occurrence. We can then apply a plethora of known risk measures to the set of all optimal energies $\mathcal{E} = E_1^\star, \ldots, E_M^\star$ and find an estimate $\hat{\rho}(u, g)$. Unsurprisingly the quality and computational costs of this approximation grow with $M$. In particular, we use the Conditional Value-at-Risk $\mathrm{CVaR}_{1-\gamma}(X)$ for a percentile $1-\gamma$. More details on Conditional Value-at-Risk can be found in [21, Sec. 3.3]. Intuitively, $\mathrm{CVaR}_{1-\gamma}$ is the

expected value of the $1-\gamma$ percentile of worst outcomes of the random variable $X$, or the expectation of the $\gamma$-tail of $X$. We define $\text{CVaR}_{1-\gamma}$ as

$$\text{CVaR}_{1-\gamma}(X) = \frac{1}{\gamma}\int_0^{\gamma} \text{VaR}_{1-t}(X)\mathrm{d}t,$$

where VaR is the Value-at-Risk [21, 22]:

$$\text{VaR}_{1-\gamma}(X) = \inf_{t\in\mathbb{R}}\{t : \mathbb{P}(X \leq t) \geq 1-\gamma\}.$$

There are several reasons why CVaR is beneficial, and we point the reader to [21] for further reading but highlight the intuition behind the choice. By measuring the expectation of the *bad* tail of $\mathcal{E}$ we achieve two beneficial goals towards the learning problem. The first is robustness: since CVaR captures the expectation of outcomes, a highly unlikely scenario with extremely high cost accounted for. This is important because we saw in Figure 6.3 that a rare event could trigger a monumental detour from the expected route. Second, CVaR captures distributions with high tail asymmetry. Figure 6.4 highlights the importance of this feature, where we observe rare events with high resource requirements. These events are exactly the ones that can cause a failure, and CVaR can capture the magnitude of these events, whereas most of the popular risk measures do not.

The last step in building our lookup table is to repeat this process for every single origin/destination pair of $\mathcal{G}$. For the relatively small graph of Downtown Chicago ($|V| = 3588$), a Python script running on 64 threads would take up to 7 years to finish. Alternatively, we could run the Monte-Carlo approximation only on the origin/destination pairs of cars we are interested in at the moment. A conservative estimate of 20 available cars would still require around 7 minutes of CPU time on the same graph of Downtown Chicago. We argue that this is unacceptable, especially in the high-stakes business of logistics. In large graphs this computational cost grows exponentially, deepening the problem of scalability. To address this shortcoming we now propose a neural network approach to fill in gaps of a sparsely populated lookup table.

### 6.2.2 Deep Neural Network Design

In this section, we build on the previous. Instead of computing every entry of the risk lookup table, we sparsely sample origin/destination points on the graph and compute their risk; then a Neural Network (NN) is engineered to infer the gaps and provide a fast estimate whenever needed. We preface this section by discussing the properties of Problem 6.1 that are particularly challenging for a NN to solve it.

The first and most obvious challenge is the sparsity of data. The Downtown Chicago map shown in Figure 6.4 is optimally simplified using R-Tree reduction [2] and it still contains 3588 nodes, for a total of 12,870,156 possible origin/destination pairs. Any dataset (artificial or real) will not be large enough to cover a significant portion of the graph. This fact is usually not a major hurdle for Neural Networks, however, cities usually possess many features such as mountains, highways, and rivers. It will become clear as we develop this method that there is a major trade-off between identifying these features and proper generalization. Next is the nature of using a Monte-Carlo approach to building data. Picture two origin nodes next to each other and a single destination node. If the driver makes no mistakes, the route from either origin node towards the goal should be largely the same, however, when running each trial different errors might occur that make one of the nodes seem a lot riskier than its neighbor. This phenomenon can be dealt with by increasing $M$, but it comes at a steep cost of computational time when generating data. Instead, a proper NN design should understand the spatial proximity of these two nodes and infer that their estimated risk should be strongly correlated.

We aim to provide as input data the origin and destination coordinates, and output data the difference $\delta_E = \mathrm{CVaR}_{1-\gamma}(\mathcal{E}) - \min f_E(\mathbf{v}_n, \mathbf{t}_n, v_d)$ with $\mathbf{v}_n, \mathbf{t}_n = \mathcal{T}(o, g, 0)$ for some origin node $o$ and goal node $g$. Since the most likely route for any given driver to take is the deterministic shortest path between $o$ and $g$, we call that the *nominal* route $\mathbf{v}_n$ with time vector $\mathbf{t}_n$. The reason behind providing the delta between the deterministic energy and $\mathrm{CVaR}_{1-\gamma}(\mathcal{E})$ is part of the network design. Traditionally it is desirable to have training data be nicely distributed on a small range, usually the real unit range $\mathcal{U} = [0, 1]$. Since $\delta_E$ does not directly depend on the distance between the deterministic path and the depot it better reflects specific features of the path instead of its spatial location. Indeed, when comparing data it was observed that, for the Downtown Chicago area, we have $\mathrm{Var}(\mathcal{D}(\mathrm{CVaR}_{1-\gamma}(\mathcal{E})))/\mathrm{Var}(\mathcal{D}(\delta_E)) \approx 0.5$,

where $\mathcal{D}(x)$ is a collection of every available $x$ (the entries of the lookup table that were computed for training purposes). Figures 6.6-6.9 depict distributions for both quantities. When using the network for inference the computation of $\min f_E(\mathbf{v}, \mathbf{t}, v_d)$ is a fast linear search, which is added to the NN output.

As an initial effort, we provide a solution based on relatively simple deep perceptrons. The details of design choices will be discussed in the results section, where we highlight the downfalls of early attempts. Namely, the final architecture has $n_n$ neurons per dense layer, with $n_l$ layers. The depth and width of the network depend on the size of the graph and the sparsity of its associated dataset. Additionally, we use the scaled exponential linear unit (SELU) activation layer [61], with batch and dropout normalization. Figure 6.5 depicts the architecture, where we use mean square error (MSE) as the loss function.

## 6.3 Test Cases

To validate this research direction we chose four locations with distinct properties, outlined in Table 6.1. These locations are of increasing complexity. Champaign-Urbana is a small, flat town, with little to no geographical properties and a very structured grid-like street layout. Downtown Chicago (or the Chicago Loop) is roughly the same size and structure as Champaign-Urbana, but it has a coast and rivers flowing through the city. Chicago (the City of Chicago) is a larger version of Champaign-Urbana; at this scale, the coast and rivers lose significance. Lastly, Rio de Janeiro is a large, highly unstructured city with many geographical features. Figure 6.10 summarizes the structural properties of these locations, where the non-grid network of Rio de Janeiro shows as a spread-out plot. Figures 6.6-6.9 outline the data statistics for the different cities. We highlight a strong concentration at or below the 20th percentile for most datasets, and that there appears to have, for all cases, concentration lines around certain values. This indicates that some structure exists in the dataset, which we aim to find via DNNs. We hypothesize that these structures are popular roads, supported by the fact that they are more prevalent in the larger maps. Additionally, Figures 6.11-6.14 depict the street maps for all four regions. Table 6.1 shows statistics for all datasets, which are made publicly available at `https://github.com/gbarsih/Safe-Rendezvous-RL`.

| Location | Dataset Name | $\|V\|$ | $\|E\|$ | $\|\mathcal{D}\|$ |
|---|---|---|---|---|
| Champaign-Urbana | champaign | 3450 | 9975 | 217399 |
| Downtown Chicago | dtchicago | 3588 | 9351 | 223278 |
| Chicago | chicago | 28455 | 75921 | 213782 |
| Simplified Chicago | simplifiedchicago | 4118 | 10522 | 152057 |
| Rio de Janeiro | janeiro | 69561 | 171161 | 275919 |
| Simplified Rio | simplifiedjaneiro | 8217 | 14525 | 224628 |

Table 6.1: Locations chosen for testing, dataset name indicates a label used throughout the dissertation.

## 6.4 Results

In this section, we discuss results for all regions. In summa, we found that the simple perceptron model in Figure 6.5 is capable of satisfactory performance for medium-sized graphs such as Champaign-Urbana and Downtown Chicago.

To analyze results we introduce two performance metrics:

- Perf: Relative ranking performance, if 8 out of 10 drivers are ranked correctly, then Perf = 80%.
- $e_{\text{rank}}$: Mean relative ranking error, for the 2 incorrectly ranked drivers with errors of 10% and 20%, we obtain $e_{\text{rank}} = 100 \cdot ((0.1+0.2)/2) = 15\%$.

An ideal model would have high raking performance and low ranking error. We show in the following results that looking purely at how well a given DNN ranks drivers can be misleading. The intuition behind this metric is the same as the one for using Risk-Averse MPC over Stochastic or Robust MPC. High performance indicates near-optimal results but neglects tail events. By adding the equivalent of a robustness metric we encompass the risk-averse objectives of this algorithm. In practice the addition of $e_{\text{rank}}$ aims to inform us of the magnitude of error, colloquially it answers the question *"Should the safest driver be ranked incorrectly, what is the potential of an accident?"*. It will be shown in each of the following examples that high performance (compared to how machine learning usually performs) is relatively trivial to achieve, but that is not the case with robustness. In Section 6.4.3, we use Chicago to discuss the implications of these results and how to address this obvious deficiency, and in Section 7.1 we further discuss future directions for this research field.

### 6.4.1 Champaign-Urbana

In this section, we discuss the results for Champaign-Urbana (CU). Figure 6.11 depicts the street network of CU, where we notice that it presents unique features we can map. First is the unique shape, since CU is composed of two smaller towns (Champaign to the east and Urbana to the west) there is a natural narrowing between them. Additionally, with only one major highway cutting across the region, and few access points to that highway, we expect to find a significant structure for this graph.

Figure 6.16 shows a heatmap inference via a network with $n_n = 128$, $n_l = 3$, and $p_d = 0.2$. With this structure, we achieve a ranking performance of 97%, with $e_{\text{rank}} = 10\%$. In this figure, hotter colors relate to higher risks, with each quadrant depicting the same depot location (green) with different destinations (cyan). A risky (hotter) location on the heatmap indicates that a driver departing from that location is going through a riskier route than others departing from a safe location (colder).

As expected we captured structures that directly translate to main streets and specific regions of town. Additionally, we obtained an obvious but reassuring result: regions that are across from the destination through the depot are significantly safer than others. This is expected – these cold regions must pass close to the depot to reach the destination. This feature is a good indication that the DNN is providing useful results. Figure 6.17 depicts the same scenatio but projected onto the graph.

| Activation | $n_l$ | $n_n$ | $p_d$ | MSE | MAE | STD(MSE), $n = 5$ |
|---|---|---|---|---|---|---|
| ReLU | 3 | 512 | 0.2 | 1.07E-02 | 5.50E-02 | 7.97E-05 |
| SELU | 3 | 512 | 0.2 | 1.08E-02 | 5.85E-02 | 4.36E-05 |
| ReLU | 2 | 512 | 0.2 | 1.14E-02 | 5.86E-02 | 1.31E-04 |
| ReLU | 3 | 128 | 0.2 | 1.15E-02 | 6.05E-02 | 7.16E-05 |
| SELU | 3 | 128 | 0.2 | 1.23E-02 | 6.37E-02 | 6.35E-05 |
| ReLU | 2 | 128 | 0.2 | 1.30E-02 | 6.55E-02 | 2.24E-04 |
| ReLU | 3 | 512 | 0.8 | 1.33E-02 | 6.67E-02 | 4.74E-05 |
| SELU | 2 | 512 | 0.2 | 1.36E-02 | 6.84E-02 | 3.67E-05 |
| SELU | 3 | 512 | 0.8 | 1.40E-02 | 6.98E-02 | 3.53E-05 |
| SELU | 2 | 128 | 0.2 | 1.41E-02 | 7.03E-02 | 5.50E-05 |

Table 6.2: Champaign-Urbana top performing learning hyperparameters.

### 6.4.2 Downtown Chicago

In this section, we discuss the results of Downtown Chicago, a region we define as a 5000m bounding box from 208 E Randolph Street (Cloud Gate, the Chicago Bean). This is a scenario where we expected strong results. Downtown Chicago is an almost perfectly rectangular region with almost all streets oriented in a grid. Counter-intuitively, Downtown Chicago was revealed to be challenging. The grid structure combined with the square region yielded too few features for the DNN to capture. Figure 6.18 depicts results identically to Figure 6.16. Combined with the map projection on Figure 6.19 we notice that the DNN was able to correctly identify the east side of the river bank. This is expected, drivers starting on that side will most likely be routed through Lake Shore Drive, which runs along the shore of Lake Michigan. Meanwhile, drivers starting from the west bank will be routed through the highway, as shown in figure 6.3.

We reached a ranking performance of 83%, the ranking error rose to 17%. The DNN was still able to capture routed going through the depot with low risk. Attempts to capture more structure from this region cause the network to overfit and $e_{\text{rank}}$ to grow exponentially. In the following examples, we will see this behavior, where a small network (identical to Champaign-Urbana) was insufficient, but a larger network overfits on the data.

### 6.4.3 Chicago

Chicago proved to be a scaled-up problem of its downtown area. Results are found inf Figures 6.20 and 6.21. After extensive experimentation, we found that a larger network was necessary to achieve a satisfactory ranking performance. Unfortunately, such a large network suffered overfitting, causing the ranking error to increase significantly, to 88%. In our estimation, this performance leaves much to be desired. With a ranking performance of 92%, on average one in ten drivers will be incorrectly selected as the top choice, and that driver might have up to 88% error from the true risk value. Such a discrepancy can easily cause a crash. However, we do see silver linings. Just as before, routes traveling through the depot heading towards the destination have low risk. Additionally, the lower-left plot correctly identified the lakeshore as a feature, in this particular case drivers starting on the coast might be routed

through Lake Shore Drive if no mistakes are made, but might get rerouted through the highway system if they do make a mistake and head towards the grid system. The Network also seems to have identified a standalone, square, neighborhood on the lower-left portion of the map. This region seems to be risky in all scenarios but one, where it is the lowest risk. In places where risk is high, we postulate that this is due to the two possible ways of exiting that region, and in the scenario where it is a low risk we find that it is due to the diametrically opposed destination.

This scenario showed a flaw in the framework. We propose this flaw be fixed with one of the following approaches. We know that for smaller graphs we can obtain good performance. A solution is to segment the map into discrete regions. A hierarchical algorithm can then, on a low level, learn risks within these "districts" and a high-level algorithm finds the flow model between them. Figure 6.22 shows a hypothetical segmentation of Chicago. We highlight the Downtown area, where we obtained good performance. An alternative solution is to use a more sophisticated loss function. We use MSE in this work, however, it does not reflect the objective of the algorithm in its entirety. If the objective is to correctly rank drivers, an ideal loss function should measure raking performance and error. At the time of this writing, the author is unaware of any methods that could accomplish this without modifications, and the investigation of such an architecture is left as future work. Another approach is to somehow simplify these complex graphs. We did so by eliminating the most local roads and focusing on major access ways. The careful reader noticed that Table 6.1 shows two simplified versions of Chicago and Rio de Janeiro. Table 6.3 presents results for the simplified versions of these graphs. The question of whether or not this simplification is sufficient to represent a risk in the smaller scales that were eliminated is left as a future study. Undeniably (and expectedly) the simplification improved results dramatically. We still argue that Rio de Janeiro is not satisfactory, even when compared to the results presented next.

### 6.4.4 Rio de Janeiro

Rio de Janeiro, as expected, was revealed to be the most challenging scenario. Figures 6.23 and 6.24 depict the results. We achieved a ranking performance of 94% after increasing the dropout probability to 0.4. Unfortunately this came

at cost of $e_{\text{rank}} = 37\%$. We expect that the solutions proposed before will have identical efficiency here.

| Location | $\|V\|$ | $\|E\|$ | Perf | $e_{\text{rank}}$ |
|---|---|---|---|---|
| Chicago | 28455 | 75921 | 98% | 41% |
| Simplified Chicago | 4118 | 10522 | 99% | 8% |
| Rio de Janeiro | 69561 | 171161 | 95% | 35% |
| Simplified Rio | 8217 | 14525 | 99% | 24% |

Table 6.3: Performance and robustness results for simplified versions of Chicago and Rio de Janeiro.

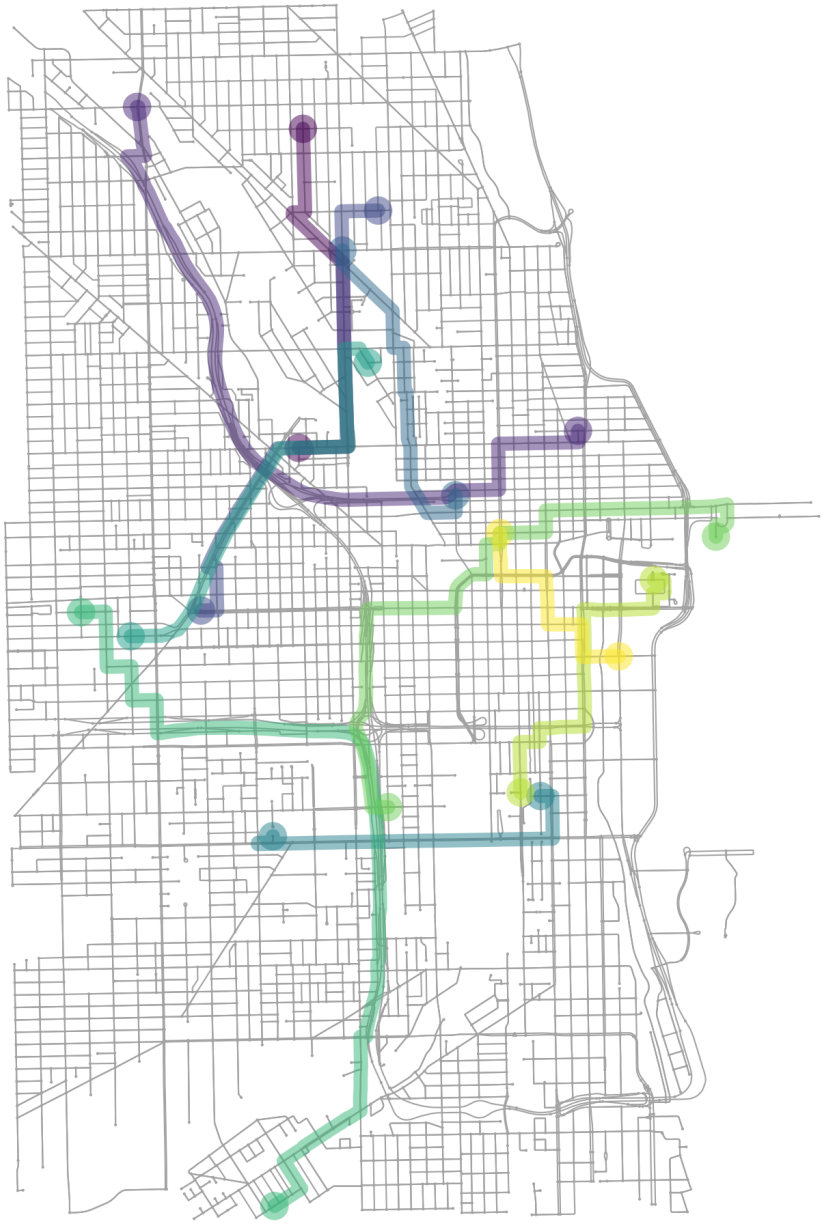


Figure 6.1: Several moving vehicles with respective routes in the Downtown Chicago area.

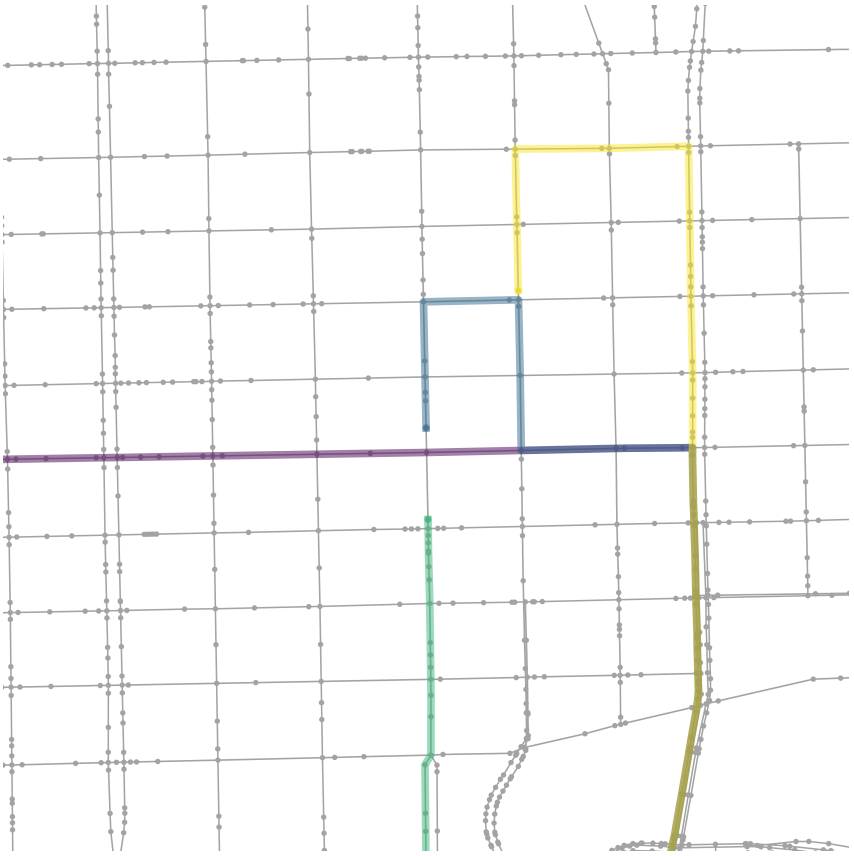


Figure 6.2: Consequences of making a mistake: some errors (yellow, blue) reroutes to the prescribed route (purple), others (green) causes an entirely new route to be calculated.

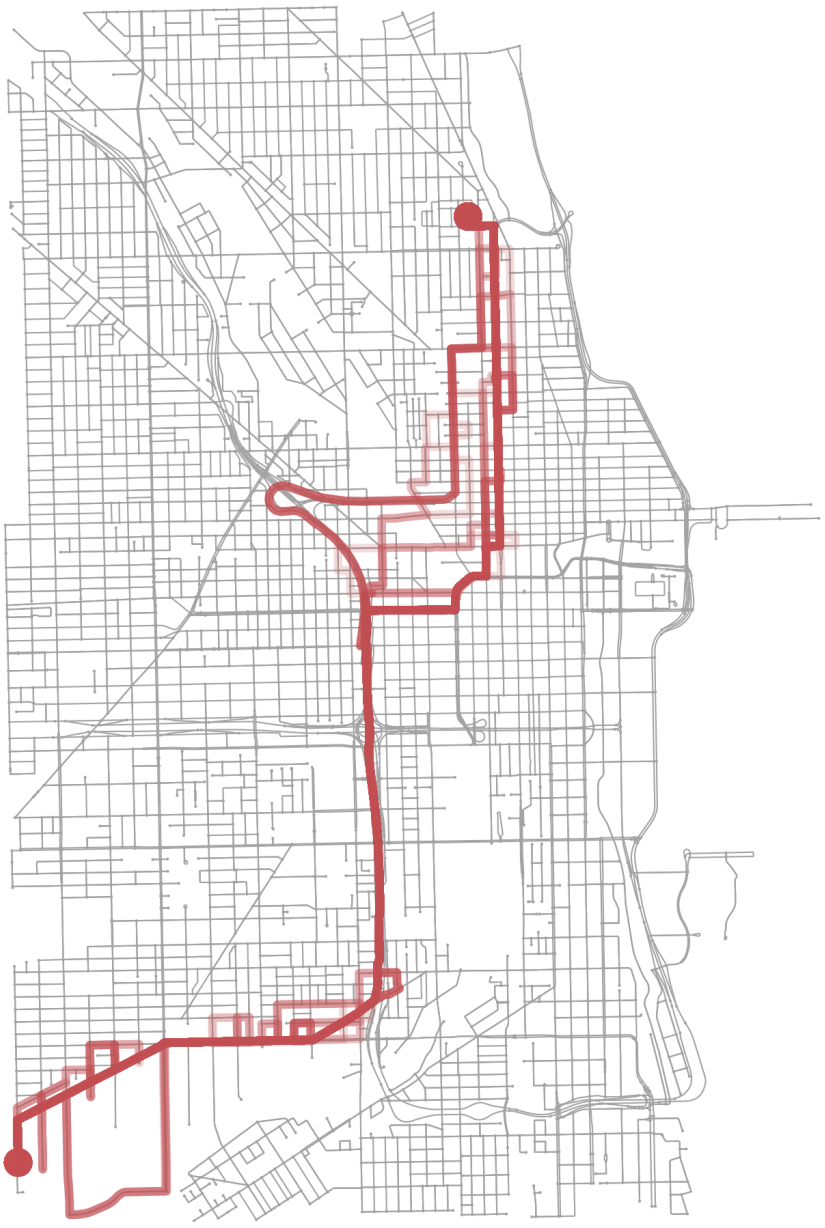


Figure 6.3: Multiple drivers simulated in the Downtown Chicago area driving between two points: minor mistakes cause little concern, but some drastically change the driver future position. Can we estimate which origin and destination pairs are less prone to these large deviations?

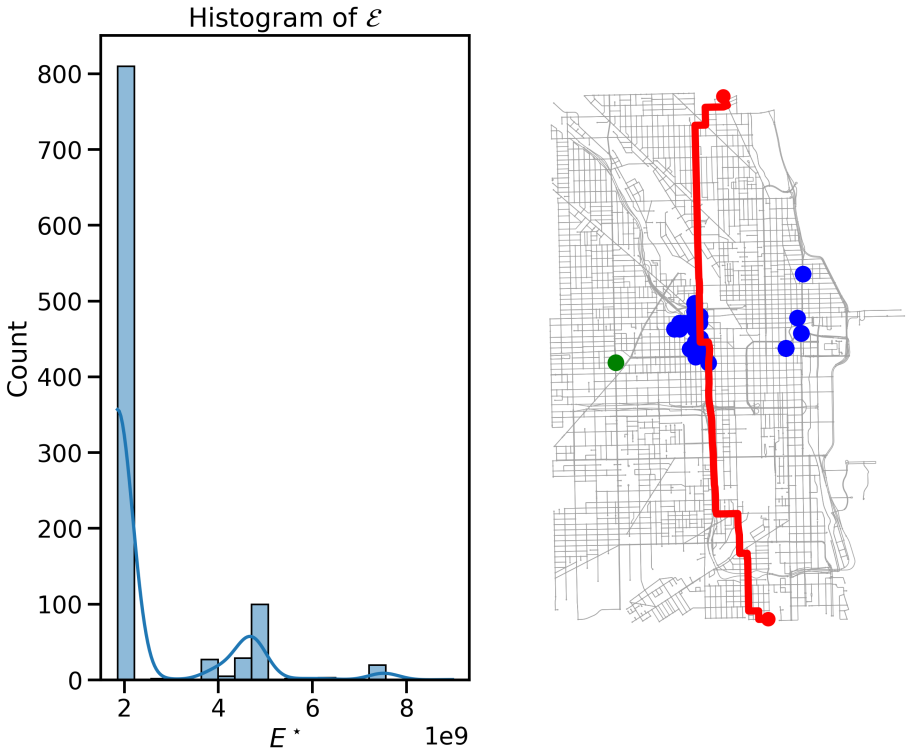



Figure 6.4: One Monte-Carlo run with $M = 1000$. Left: histogram of $\mathcal{E}$. Right: map of Downtown Chicago with optimal rendezvous locations in blue, depot location in green, deterministic route sequence $\mathcal{T}(u, g, 0)$ in red. Rare events in the histogram indicate possibly catastrophic resource requirements.

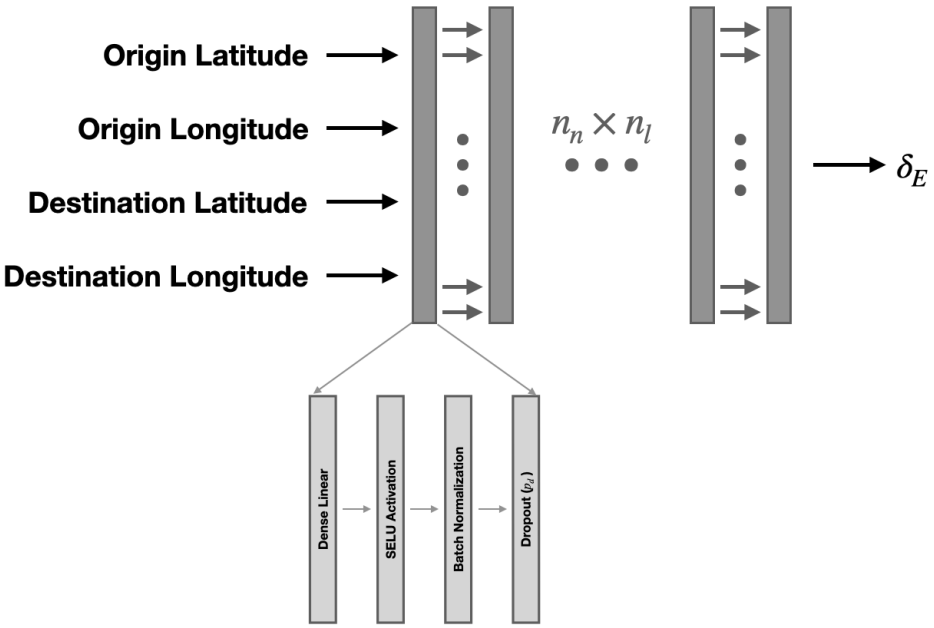



Figure 6.5: Neural Network architecture.

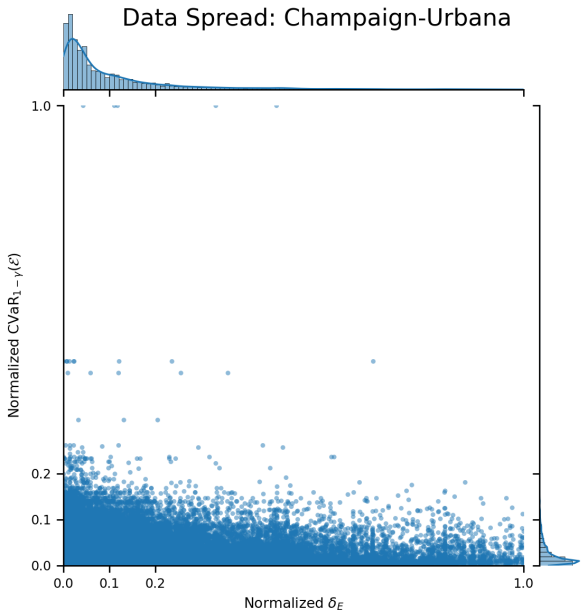



Figure 6.6: Data spread for Champaign-Urbana: Utilizing $\delta_E$ as training target variable provides better input distribution.

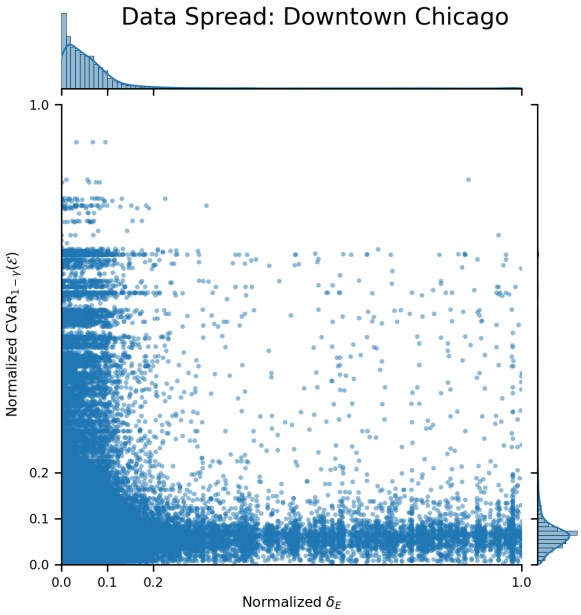



Figure 6.7: Data spread for Downtown Chicago.

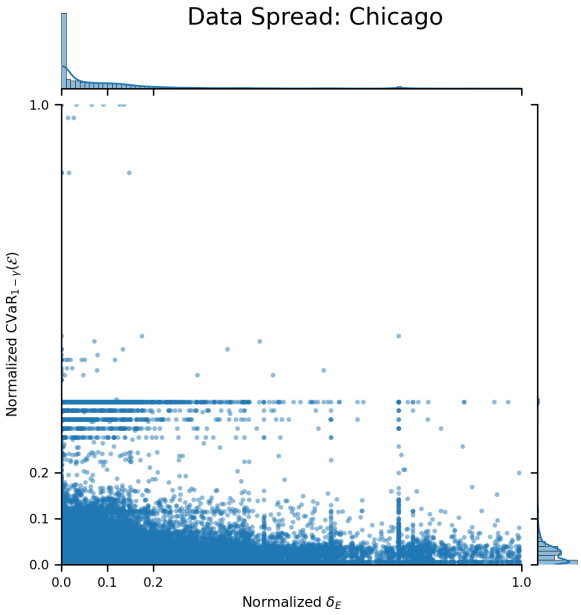



Figure 6.8: Data spread for Chicago.

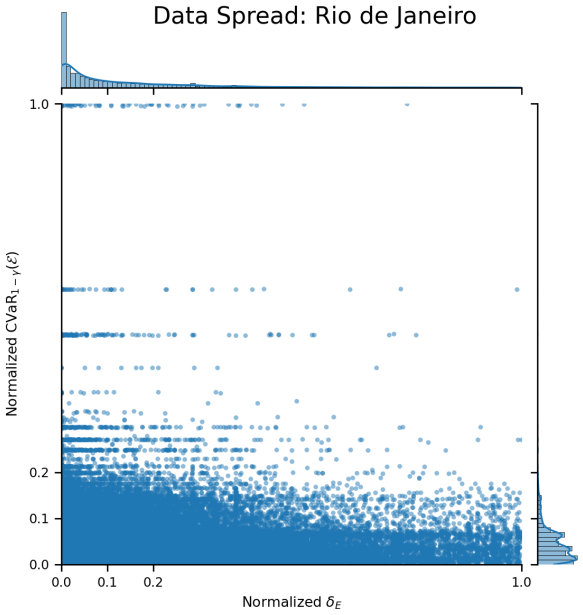



Figure 6.9: Data spread for Rio de Janeiro.

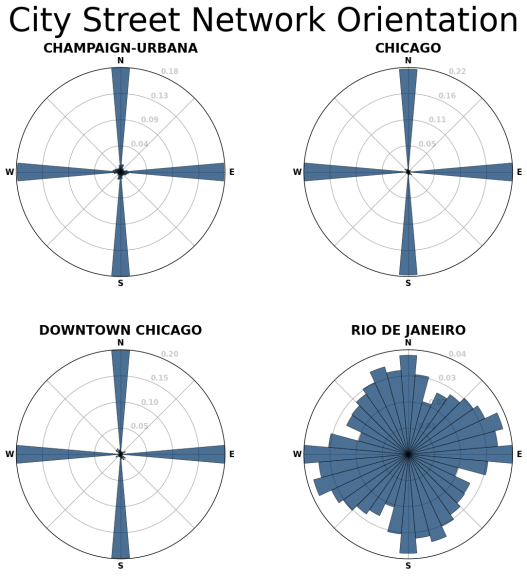



Figure 6.10: Street network orientation for test locations [2].

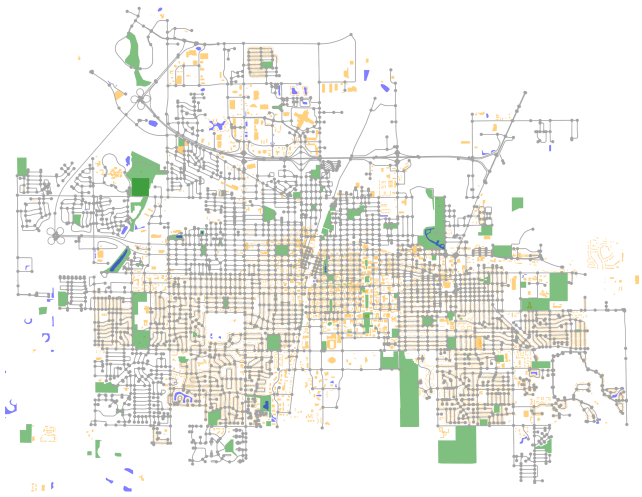


Figure 6.11: Map of Champaign-Urbana with natural features and building depicted.

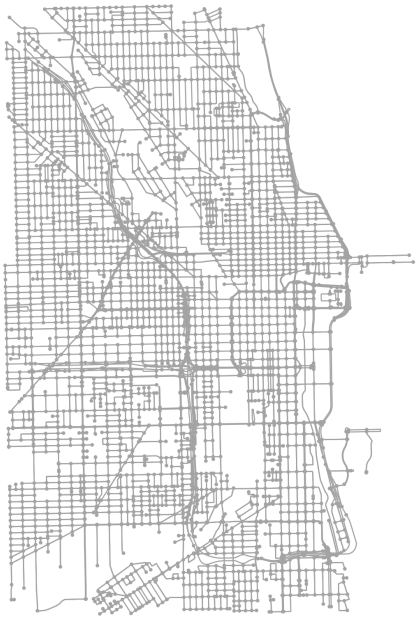


Figure 6.12: Map of Downtown Chicago.

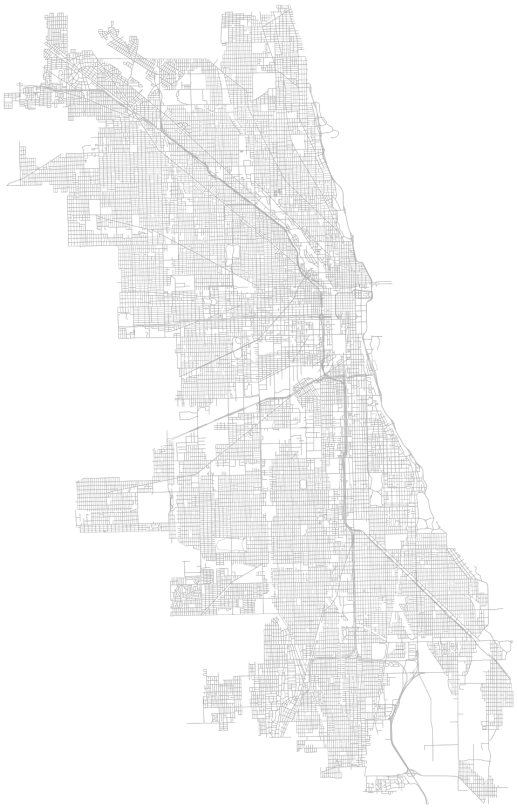


Figure 6.13: Map of Chicago.

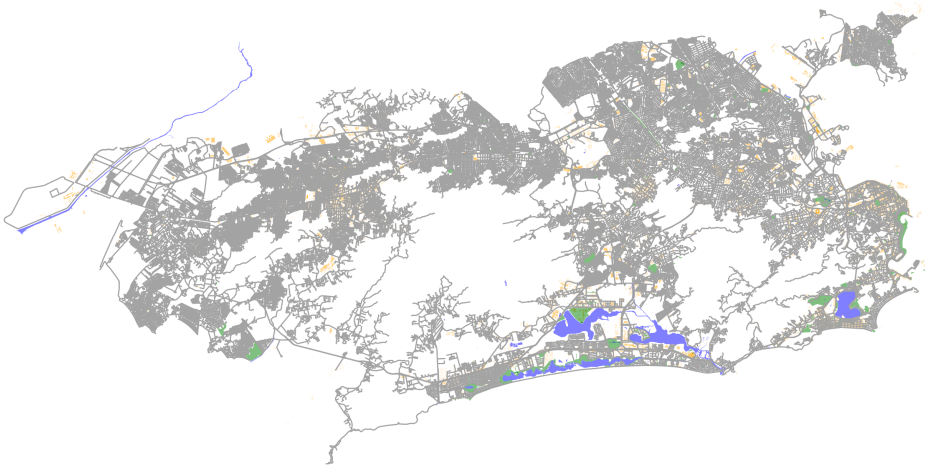


Figure 6.14: Map of Rio de Janeiro with natural features and building depicted.

Figure 6.15: Map of a neighborhood in Rio de Janeiro, unstructured street orientations provide an almost chaotic behavior to route planning, and a challenge to our risk learning efforts.

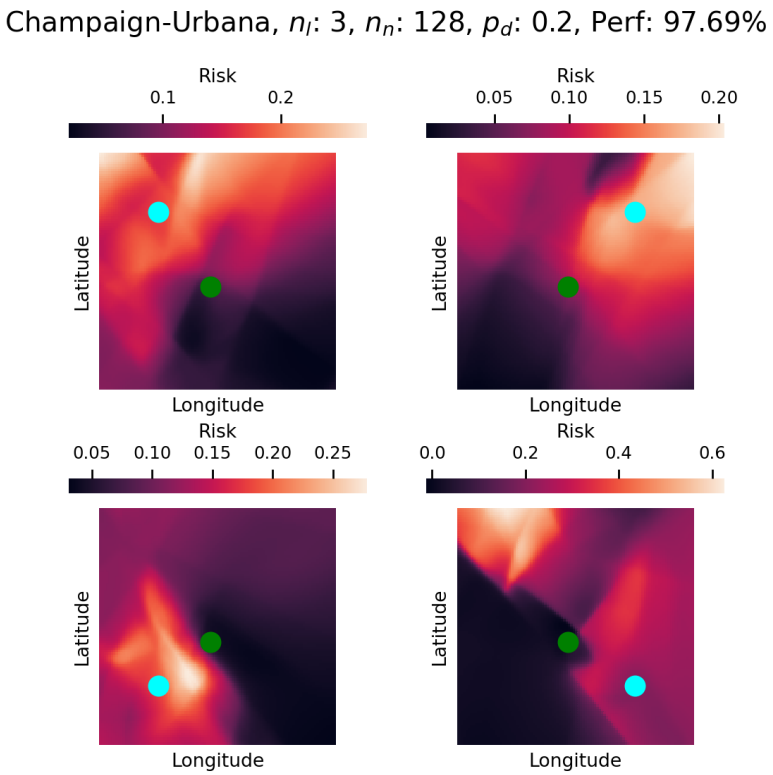



Figure 6.16: Heatmap of risks for different destinations for the Champaign-Urbana area: depot location in green, destination in cyan. Hotter colors indicates origins of higher risk, when selecting drivers for a given delivery destination.

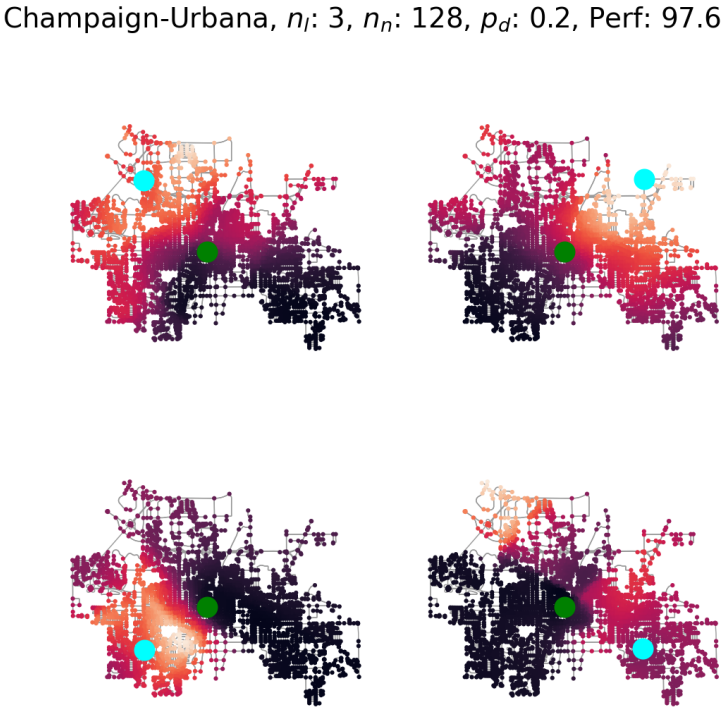



Figure 6.17: Heatmap risks for different destinations overlaid on the Champaign-Urbana area map: depot location in green, destination in cyan. Hotter colors indicates origins of higher risk, when selecting drivers for a given delivery destination.

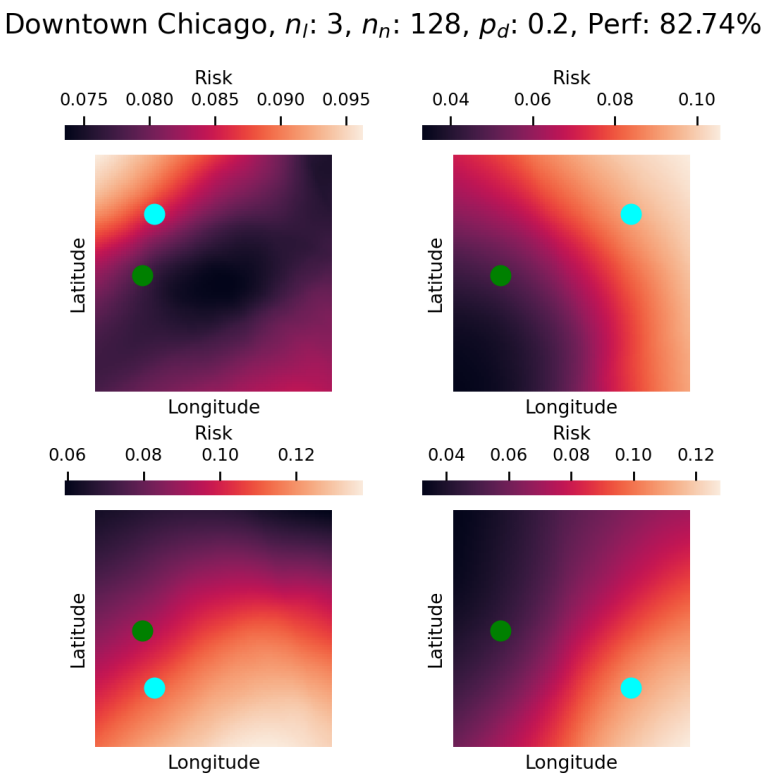



Figure 6.18: Heatmap of risks for different destinations for the Downtown Chicago area: depot location in green, destination in cyan.

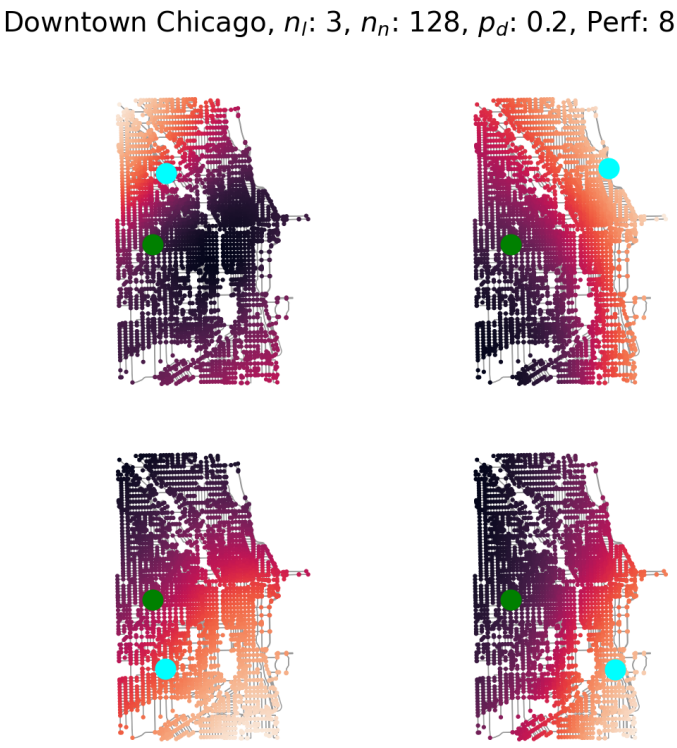



Figure 6.19: Heatmap risks for different destinations overlaid on the Downtown Chicago area: depot location in green, destination in cyan.

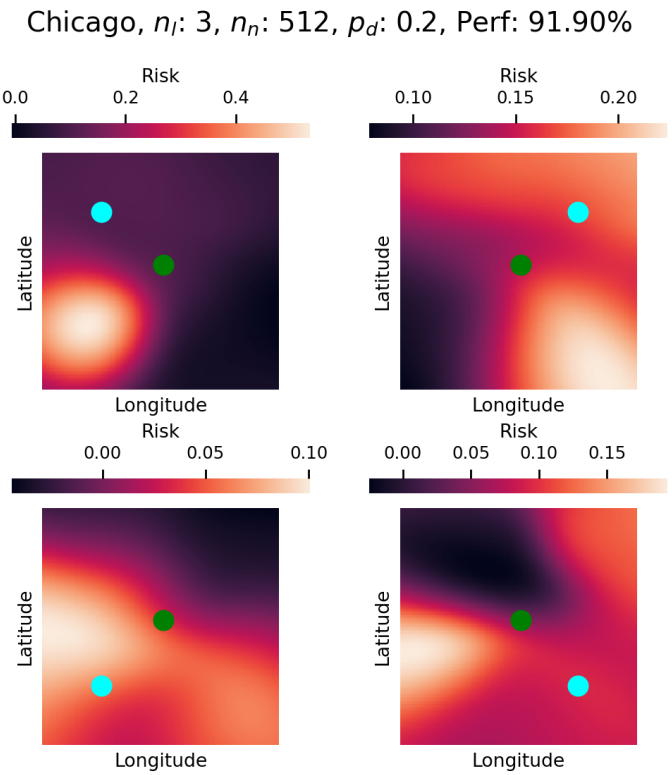



Figure 6.20: Heatmap of risks for different destinations for the Chicago area: depot location in green, destination in cyan.


Figure 6.21: Heatmap risks for different destinations overlaid on the Chicago area map: depot location in green, destination in cyan.

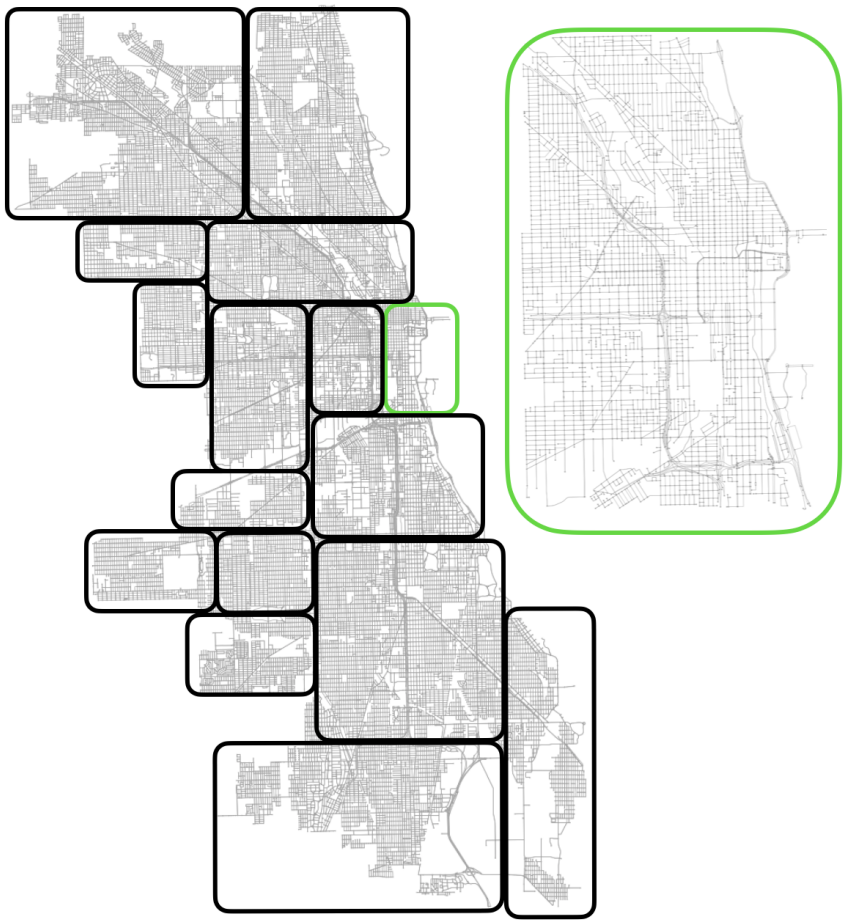


Figure 6.22: Segmentation of Chicago, with a Downtown Chicago highlighted.

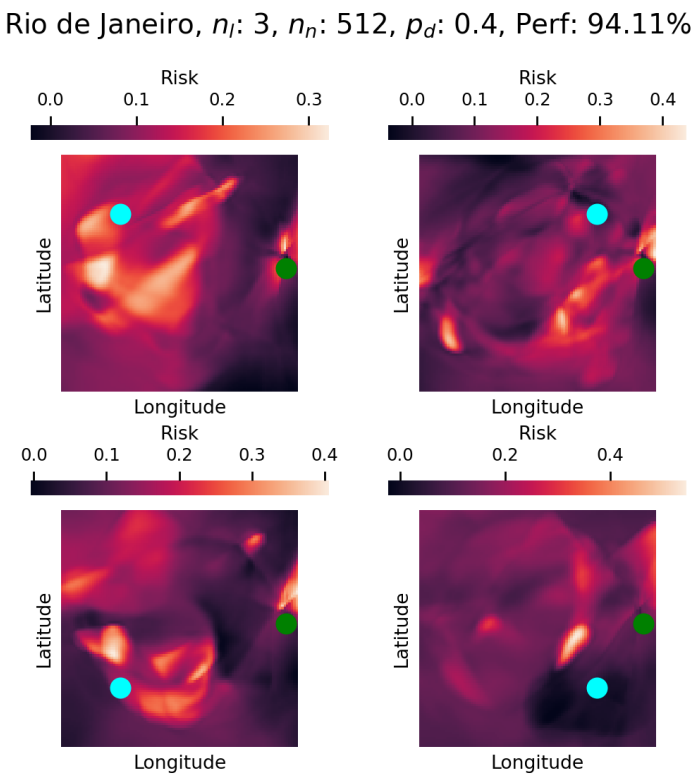



Figure 6.23: Heatmap of risks for different destinations for the Rio de Janeiro area: depot location in green, destination in cyan.

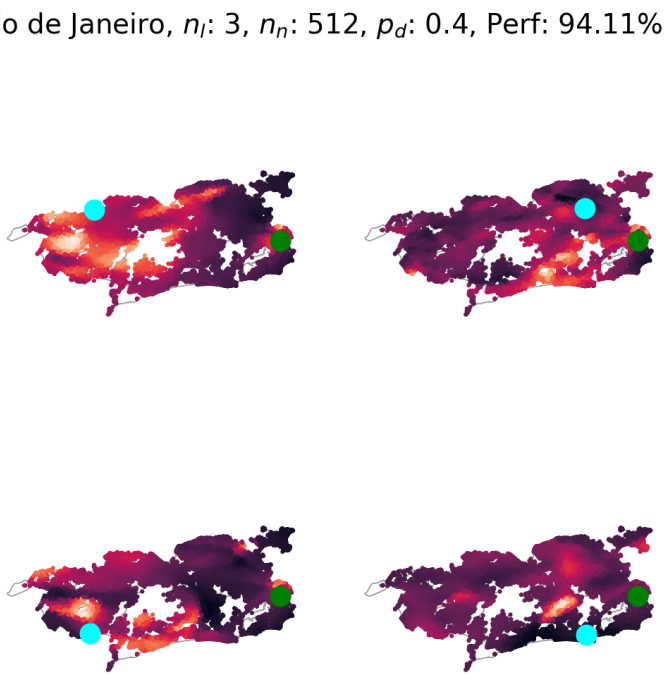



Figure 6.24: Heatmap risks for different destinations overlaid on the Rio de Janeiro area map: depot location in green, destination in cyan.

# CHAPTER 7

# CONCLUSION

In this dissertation, we studied two safety-critical aspects of a novel approach to autonomous logistics. First, we identified deficiencies with current aerial manipulation techniques, and proposed and implemented a novel alternative. Second, we identified a safety-critical concern that uncertain human behavior may cause unmanned aerial systems to run out of resources and crash. In urban scenarios, such a crash endangers human lives. We aimed to quantify different aspects of driver behavior in regards to its impact on motion planning algorithms. In total, three algorithms are capable of addressing the two greatest sources of risk: speed variation and route choice. We progressively advanced the algorithms in three parts.

Chapter 3 investigated mechanical and algorithmic problems in aerial manipulation. Such platforms are necessary in the context of this dissertation's framework during package drop-off and pick-up. Augmenting UAS's with manipulation devices improves precision and disturbance rejection for accurately placing the parcel on top of the moving vehicle. Through a combination of a novel aerial manipulator mechanical design and robust adaptive $\mathcal{L}_1$ we achieved satisfactory performance. Additionally, by exploring the redundant nature of the system, an optimal trajectory generator was created. The algorithm is capable of conforming to various constraints, and guarantees safe operation.

Chapter 4 developed the foundations for the framework. In it, we proposed a hybrid learning-planning stack that, from noisy driver data, is capable of estimating driver behavior and planning on it. We learn driver behavior via Bayesian Linear Regression (BLR), arguably the simplest method in the Gaussian Process Regression (GPR) family. With basic assumptions, we showed that a Model Predictive Controller (MPC) plans a safe Point-of-No-Return (PNR) such that up to that point the UAS is capable of returning home with just enough resources left. We crafted the algorithm this way so that between takeoff and the PNR, the UAS is free to gather as much data as possible and

improve risk prediction. In this context, we chose pertinent risk measures to quantify human behavior. If at PNR we calculate that the risk is acceptable, we proceed with rendezvous. Otherwise, the UAS returns to the depot and tries again later. Such structure could be seen as conservative, but it is a well-known strategy in the aerospace industry. We showed that the algorithm works as expected but has two downfalls. First, it only works for a single path, when we know that route choice is a major source of risk. Second, it has a laborious setup that requires smooth paths. The succeeding algorithm fixes these deficiencies.

Chapter 5 a sampling-based approach that allows multiple paths and paths of arbitrary shape was developed. Succinctly, we divided the original MPC module into two parts. A gradient-based part chooses the location and time of the point of no return, and a gradient-free, sampling-based optimizer finds the optimal (risk and effort-wise) rendezvous location. In this algorithm, we expand the learning module to use a full Gaussian Process Regressor, that is approximated through Deterministic Training Conditionals with no accuracy loss. We showed that the entire learning and optimization stack converges fast, usually in less than five iterations. Additionally, we identified different strategies for choosing the rendezvous point. We focus on choosing the best possible point, or the worst possible point. The primary goal is to maximize the rate of successful rendezvous missions, to that end, we postulate that choosing the worst sample is the safest choice. Simply put, if we manage to find a safe rendezvous route for the *worst of the best*, then all others have equal or lesser risk. We show that this is indeed the case with numerical experiments.

In Chapter 6, we set out to find a ranking algorithm for city-wide risk assessment of drivers, with risk related to route choice. We used real-world street maps and simulated hundreds of thousands of drivers traversing four locations: Champaign-Urbana, Downtown Chicago, Chicago, and Rio de Janeiro. We used a Deep Neural Network of varying sizes to obtain a ranking performance of around 90% on average. The network found features that we intuitively expect, and some non-intuitive ones that in retrospect do agree with the premise. We also found that for the largest scenarios (Chicago and Rio de Janeiro) the error when incorrectly ranking a driver is unacceptable. We postulate that this is due to overfitting and excessive granularity. We proposed two ways of addressing this shortcoming and made the datasets available for future researchers to improve on our methods.

## 7.1 Future Research Directions

In this section, we discuss future research directions for all studies shown in this dissertation. In Chapter 3 we showed that the novel design achieves the tasks set for the device. From here the main deficiency is collision detection. The current controller can avoid collision between each link in the manipulator, but not between the robot and the environment. A comprehensive review of traditional methods that accomplish this can be found in [62]. Here we propose risk-averse MPC as a viable option as well. Instead of long-term risk, the controller can estimate local, short-term, risk in the vehicle's movement in a similar spirit to [63]. A driver constantly performing various corrections would require a more conservative approach vector. Additionally, this type of behavior can be embedded into the ranking process so that drivers equally ranked by the algorithm in Chapter 6 can be distinguished via this quantity.

Chapter 4 and its extension in Chapter 5 made a few assumptions on human behavior that may not be entirely realistic. First, we assumed that driver behavior is only dependent on a single individual. We propose here that a complete model should investigate the effects of traffic flow [64] in this problem. Traffic flow can heavily influence the speed profile of a driver, particularly when it comes to preferred lanes of traffic or following a specific distance. Additionally, we assumed an equal probability of path choice. As discussed in Chapter 5 we assumed that when the path prunes the driver will choose between the branches with equal probability. We propose that this assumption be investigated and that a new controller be synthesized that handles non-uniform probabilities. If the weights for each path are provided, they can be easily implemented in the cost function. If not, a completely new approach may be necessary.

In Chapter 6 we briefly discussed possible future directions for this research vector. We proposed two approaches, one of which we implemented in a simple form. First, we proposed that large urban areas be subdivided into smaller areas we called "districts" depicted in Figure 6.22. We reasoned this because satisfactory performance was achieved in areas the size of Champaign-Urbana as seen in Section 6.4.1. The open questions in this approach are the flow modeling between each district and algorithm design considering the new approach. One example is district selection; a potential new algorithm could be able to choose the optimal rendezvous district, one that offers the least risk.

In fact, we can connect the results of Chapter 5 to this question, but instead of solving over actual paths, we can solve over virtual paths that connect the districts. This is potentially trivial and could provide significant improvements to this approach. Still in Chapter 6 we proposed and superficially studied an approach of selecting only major access ways in large urban areas. We showed, without changing the method, that we can decisively improve performance this way. Future research that takes this path needs to answer questions on how good of an approximation these reduced graphs are. Another way of posing this question would be how well can we extrapolate models over reduced graphs to full graphs. Some research [65] has been done in a similar vein, but the particulars of the risk-averse rendezvous problem may require specialized weaponry.

# CHAPTER 8

# REFERENCES


[1] F. Borrelli, "Model predictive control lecture notes," 2014.

[2] G. Boeing, "Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126–139, 2017.

[3] M. Joerss, J. Schroeder, F. Neuhau, C. Klink, and F. Mann, "Parcel delivery: The future of last mile," 2016.

[4] M. Pomerleau, "Future of drones is small and cheap." 2017, https://www.c4isrnet.com/unmanned/uas/2017/05/12/future-of-drones-is-small-and-cheap/ [cited October 2017].

[5] G. Garimella and M. Kobilarov, "Towards model-predictive control for aerial pick-and-place," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA: Institute of Electrical & Electronics Engineers (IEEE), May 2015. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2015.7139850

[6] D. Lunni, A. Santamaria-Navarro, R. Rossi, P. Rocco, L. Bascetta, and J. Andrade-Cetto, "Nonlinear model predictive control for aerial manipulation," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2017, pp. 87–93.

[7] M. Kamel, K. Alexis, and R. Siegwart, "Design and modeling of dexterous aerial manipulator," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 4870–4876.

[8] T. W. Danko, K. P. Chaney, and P. Y. Oh, "A parallel manipulator for mobile manipulating UAVs," in *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*. Woburn, MA: Institute of Electrical & Electronics Engineers (IEEE), May 2015. [Online]. Available: http://dx.doi.org/10.1109/TePRA.2015.7219682

[9] R. M. Jones, D. Sun, G. B. Haberfeld, A. Lakshmanan, T. Marinho, and N. Hovakimyan, "Design and control of a small aerial manipulator for indoor environments," in *AIAA Information Systems - AIAA Infotech @ Aerospace, AIAA SciTech Forum*. Grapevine, TX: American Institute of Aeronautics and Astronautics, January 2017.

[10] N. E. Daidzic, "General solution of the wind triangle problem and the critical tailwind angle," *The International Journal of Aviation Sciences (IJAS)*, vol. 1, no. 1, pp. 57–93, 2016.

[11] I. E. Weintraub, M. Pachter, and E. Garcia, "An introduction to pursuit-evasion differential games," *arXiv:2003.05013*, 2020.

[12] R. Ritz, M. W. Müller, M. Hehn, and R. D'Andrea, "Cooperative quadrocopter ball throwing and catching," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4972–4978.

[13] F. Belkhouche, B. Belkhouche, and P. Rastgoufard, "Parallel navigation for reaching a moving goal by a mobile robot," *Robotica*, vol. 25, no. 1, p. 63–74, 2007.

[14] R. Yanushevsky, *Modern Missile Guidance*. Taylor & Francis, 2018.

[15] M. Kobilarov, "Cross-entropy motion planning," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 855–871, 2012.

[16] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, Dec 2016.

[17] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967 – 2986, 2014.

[18] B. G. Park and W. H. Kwon, "Robust one-step receding horizon control of discrete-time markovian jump uncertain systems," *Automatica*, vol. 38, no. 7, pp. 1229 – 1235, 2002.

[19] L. Hewing and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *CoRR*, 2017.

[20] Y. Chow and M. Pavone, "A framework for time-consistent, risk-averse model predictive control: Theory and algorithms," in *2014 American Control Conference*, June 2014, pp. 4204–4211.

[21] D. Guégan and B. K. Hassani, *Risk Measurement*. Springer, 2019.

[22] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath, "Coherent measures of risk," *Mathematical Finance*, vol. 9, no. 3, pp. 203–228, 1999.

[23] P. Whittle, "The risk-sensitive certainty equivalence principle," *Journal of Applied Probability*, vol. 23, pp. 383–388, 1986.

[24] P. Sopasakis, D. Herceg, A. Bemporad, and P. Patrinos, "Risk-averse model predictive control," *Automatica*, vol. 100, p. 281–288, Feb 2019.

[25] A. Rucco, P. B. Sujit, A. A. P., J. B. de Sousa, and F. L. Pereira, "Optimal rendezvous trajectory for unmanned aerial-ground vehicles," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 2, pp. 834–847, April 2018.

[26] S. Boyd, S. Boyd, L. Vandenberghe, and C. U. Press, *Convex Optimization*, ser. Berichte über verteilte messysteme. Cambridge University Press, 2004, no. pt. 1. [Online]. Available: https://books.google.com/books?id=mYm0bLd3fcoC

[27] D. Liberzon, *Calculus of variations and optimal control theory*. Princeton university press, 2011.

[28] R. E. Kopp, "Pontryagin maximum principle," in *Mathematics in Science and Engineering*. Elsevier, 1962, vol. 5, pp. 255–279.

[29] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1433–1440.

[30] H. Markowitz, "Portfolio selection," 1959.

[31] A. Ruszczyński and A. Shapiro, "Optimization of risk measures," in *Probabilistic and randomized methods for design under uncertainty*. Springer, 2006, pp. 119–157.

[32] R. T. Rockafellar, S. Uryasev et al., "Optimization of conditional value-at-risk," *Journal of risk*, vol. 2, pp. 21–42, 2000.

[33] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello, "The sample average approximation method for stochastic discrete optimization," *SIAM Journal on Optimization*, vol. 12, no. 2, pp. 479–502, 2002.

[34] A. Ali, J. Z. Kolter, S. Diamond, and S. P. Boyd, "Disciplined convex stochastic programming: A new framework for stochastic optimization." in *UAI*, 2015, pp. 62–71.

[35] M. Grant and S. Boyd, "Cvx: Matlab software for disciplined convex programming, version 2.1," 2014.

[36] I. Dunning, J. Huchette, and M. Lubin, "Jump: A modeling language for mathematical optimization," *SIAM review*, vol. 59, no. 2, pp. 295–320, 2017.

[37] R. Clavel, "Conception d'un robot parallèle rapide à 4 degrés de liberté," Ph.D. dissertation, EPFL, Lausanne, 1991.

[38] Robotis, "Dynamixel rx-24f," Robotis, September 2017, http://www.robotis.us/dynamixel-rx-24f-hn07-n101/ [cited September 2017].

[39] S. B. Park, H. S. Kim, C. Song, and K. Kim, "Dynamics modeling of a delta-type parallel robot (isr 2013)," in *IEEE ISR 2013*, Oct 2013, pp. 1–5.

[40] N. Hovakimyan and C. Cao, $\mathcal{L}_1$ *Adaptive Control Theory: Guaranteed Robustness with Fast Adaptation*. SIAM, 2010.

[41] F. Fahimi, *Autonomous Robots*. Springer, 2009.

[42] A. Maciejewski and C. A. Klein, "Numerical filtering for the operation of robotic manipulators through kinematically singular configuration," in *Journal of Robotic Systems*, 1988.

[43] A. Le Rhun, F. Bonnans, G. De Nunzio, T. Leroy, and P. Martinon, "A stochastic data-based traffic model applied to vehicles energy consumption estimation," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2019.

[44] X. Ren, D. Wang, M. Laskey, and K. Goldberg, "Learning traffic behaviors by extracting vehicle trajectories from online video streams," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, 2018, pp. 1276–1283.

[45] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[46] C. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[47] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[48] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, 2008, pp. 1177–1184.

[49] S. R. Kuindersma, R. A. Grupen, and A. G. Barto, "Variable risk control via stochastic optimization," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 806–825, 2013.

[50] D. Fan, A. Agha, and E. Theodorou, "Deep Learning Tubes for Tube MPC," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[51] K. Pereida and A. P. Schoellig, "Adaptive model predictive control for high-accuracy trajectory tracking in changing conditions," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7831–7837.

[52] C. Cao and N. Hovakimyan, "Design and analysis of a novel $\mathcal{L}_1$ adaptive control architecture with guaranteed transient performance," *IEEE Transactions on Automatic Control*, vol. 53, no. 2, pp. 586–591, 2008.

[53] M. G. Genton, "Classes of kernels for machine learning: a statistics perspective," *Journal of machine learning research*, vol. 2, no. Dec, pp. 299–312, 2001.

[54] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When gaussian process meets big data: A review of scalable gps," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[55] L. Csató and M. Opper, "Sparse on-line gaussian processes," *Neural computation*, vol. 14, no. 3, pp. 641–668, 2002.

[56] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "Chapter 3 - the cross-entropy method for optimization," in *Handbook of Statistics*, ser. Handbook of Statistics, C. Rao and V. Govindaraju, Eds. Elsevier, 2013, vol. 31, pp. 35–59.

[57] C. Martınez, "Partial quicksort," in *Proc. 6th ACMSIAM Workshop on Algorithm Engineering and Experiments and 1st ACM-SIAM Workshop on Analytic Algorithmics and Combinatorics*, 2004, pp. 224–228.

[58] B. Brown and E. Laurier, "The normal natural troubles of driving with gps," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2012, pp. 1621–1630.

[59] B. P. V. Samson and Y. Sumi, "Exploring factors that influence connected drivers to (not) use or follow recommended optimal routes," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–14.

[60] G. B. Haberfeld, A. Gahlawat, and N. Hovakimyan, "Safe sampling-based air-ground rendezvous algorithm for dense street maps," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2021, pp. 413–422.

[61] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proceedings of the 31st international conference on neural information processing systems*, 2017, pp. 972–981.

[62] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*. Springer, 2008, vol. 200.

[63] S. Yi, H. Li, and X. Wang, "Pedestrian behavior understanding and prediction with deep neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 263–279.

[64] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2014.

[65] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.