2021 International Conference on Unmanned Aircraft Systems (ICUAS)
Athens, Greece. June 15-18, 2021

Safe Sampling-Based Air-Ground Rendezvous Algorithm for Dense Street Maps*

Gabriel Barsi Haberfeld[1], Aditya Gahlawat[2], and Naira Hovakimyan[3]

***Abstract*— Demand for fast and economical parcel deliveries in urban environments has risen considerably in recent years. A framework envisions 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. Notably, we consider the problem of planning a rendezvous path for the UAS to reach a human driver, who may choose between N possible paths and has uncertain behavior, while meeting strict safety constraints. The long planning horizon and safety constraints require robust heuristics that combine learning and optimal control using Gaussian Process Regression, sampling-based optimization, and Model Predictive Control. The resulting algorithm is computationally efficient and shown to be effective in a variety of qualitative scenarios.**

I. INTRODUCTION

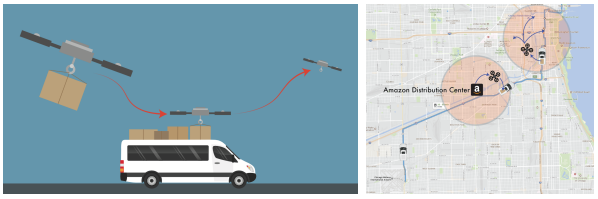
Modern transportation solutions can accumulate more than half of the total shipping cost on the transportation portion between the final distribution center and the customer [1]. This fact is known as the *last-mile problem*. Our proposed framework consists of using the existing large networks of ride-sharing services (Uber, Lyft) to cover most of the distance from the final distribution center to the customer. This process uses knowledge of these vehicles' destination to plan deliveries, where a 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 customer. An example scenario is illustrated in Fig. 1. 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. Environmental factors such as wind, package mass, sloshing of package contents, battery age, and others contribute to these safety concerns. However, because of the long planning horizons associated with these missions, the primary source of risk and uncertainty arises from the inaccurate driver behavior, where a driver might be slower, faster, or generally unpredictable. In this paper, we build on our previous solution for simple missions [2].

*This work was supported by NSF NRI award 1830639.

[1]Gabriel Barsi Haberfeld is a Ph.D. Student with the Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign gbh2@illinois.edu.

[2]Aditya Gahlawat is a Postdoctoral Researcher with the Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign gahlawat@illinois.edu.

[3]Naira Hovakimyan is with Faculty at the Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign nhovakim@illinois.edu.



(a) (b)

Fig. 1: Air-ground rendezvous procedure. Left: the UAS needs to meet an uncontrollable ground vehicle with uncertain trajectory. Right: UASs intercept the vehicle at various points to complete the delivery. In this example a package is carried by a ride-sharing vehicle departing from Chicago Midway Airport bound to Downtown Chicago.

The main extension is that the method now admits non-differentiable paths and allows the uncertainty of path choice, where a driver might choose a different route than the one shared with the UAS a priori.

Model Predictive Control (MPC) is a popular method for solving local Optimal Control Problems (OCP) in real-time [3], where the OCP is solved at each iteration of the control loop. Although versatile, traditional MPC is not equipped to deal with large uncertainties over long planning horizons due to exponentially increasing uncertainty propagation in the planning stage. To address these issues two common solutions are (a) stochastic MPC (SMPC) [4], [5] and (b) Robust MPC [6], [7]. Stochastic MPC is often referred to as *risk-neutral*, as it aims to minimize expectations singularly, while Robust MPC accounts for worst-case scenarios. In some cases, an absolute approach is desirable, but often the problem requires a trade-off between high risk and robustness, as not to diverge too far away from optimality. In the context of urban aerial logistics, we aim to minimize the risk of running out of battery and inevitably crashing. In this paper, we handle tractability problems of optimizing over risk measures [8] with a gradient-free sampling-based approach. The method shown in Section III also allows planning over multiple non-differentiable paths.

In our previous work [2] 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

2021 International Conference on Unmanned Aircraft Systems (ICUAS) | 978-1-6654-1535-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICUAS51884.2021.9476815

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 work, 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 [9]. 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 II we formalize the problem setup.

A. Related work

Several papers have considered risk measures in planning and handling uncertainties in an MPC framework, as summarized in [3], [10], and shown in [7], [11], [12], [13], [14], [15], [16]. In [11], the authors study uncertainty propagation to ensure chance constraints on a race car; results show that the algorithm can learn uncertainty in the dynamics, associating risk with the unknown dynamics, and plan so that the trajectories are safe. In [12], 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 [15], modern developments in [16] 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 [17] 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.

Lastly, 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 paper, we sample rendezvous times and their associated rendezvous location. Two approaches are directly related to this paper. In [9] the authors present a novel method to find trajectories for mobile robots in cluttered environments. In [18], the authors present a sampling-based MPC that integrates risk management using Conditional Value-at-Risk (CVaR) [13, Sec. 3.3]. Both motion planning and usage of CVaR are relevant to this paper, both of which are described in Section III.

B. Problem Novelty

The rendezvous (or interception) problem is not new. Traditionally these problems fall into two categories: interception of a target on a known path or interception of a target with an unknown trajectory [19]. Full knowledge of target behavior makes the problem trivial as shown in [20]. In these scenarios, the goal is to find the *optimal* trajectory for interception. Conversely, no knowledge of target behavior requires problem relaxation [21], [22] with little guarantees. In this paper, we require certain constraints usually not afforded by latter, and have only partial target behavior knowledge. Thus, we require a custom solution that exploits the particulars of the rendezvous problem to provide safety guarantees.

C. 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 paper 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 [9] 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 paper is structured as follows: in Section II, we introduce and define the problem in algorithmic format. In Section III, we present the three main components of this approach: the

model learning, the importance sampling scheme, and the MPC controller. In section IV, we demonstrate results for individual modules and two full planning stack examples. Finally, in Section V, we provide concluding remarks and discuss the shortfalls of this approach and future directions to address them, respectively.

II. Problem Formulation

We begin by defining the notion of persistent safety.

Definition 1 (Persistent Safety)**.** *Let $x_{k+1} = f(x_k, u_k)$ be a dynamical 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 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, reducing the risk of running out of battery or fuel.

Consider a set $\mathcal{P}$ of N indexed parametrized paths $\mathcal{P} = \{p_j(\theta),\ j = 1, \dots, N\} \subset \mathcal{R}$, $p : \mathbb{R}^+ \to \mathbb{R}^2$, $\theta \in \mathbb{R}^+$ on a planar Euclidean region $\mathcal{R} \in \mathbb{R}^2$, and historical velocity data along each path $\dot{\theta}^h_j(t)$, $\dot{\theta}^h_j : \mathbb{R}^+ \to \mathbb{R}$ 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 $\dot{\theta}^d_j(t)$ 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 $\{1, \dots, N\}$. 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 θ 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 $\theta^d_j(t_{R,j})$ 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 $\theta^d_j(t_{R,j})$ 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 $g_j(\theta^d_j | t_{R,j})$. We approximate driver behavior by learning a deviation mean function $d(\theta^h_j(t)) : \mathbb{R} \mapsto \mathbb{R}$ and variance function $\Sigma_d(\theta^h_j(t)) : \mathbb{R} \mapsto \mathbb{R}$. The deviation function is such that $\theta^d_j(t) = \theta^h_j(t) + d(\theta^h_j(t))$ if learned exactly. Section III-A 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 III-C. 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 $\mathcal{A}^\star(\mu^\star_\mathcal{A}, \Sigma^\star_\mathcal{A})$. 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 Σ_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_{tgt}, such that $p^\star = p_{\text{tgt}}(\theta^d_{\text{tgt}}(t_{R,\text{tgt}}))$ is the optimal rendezvous location.

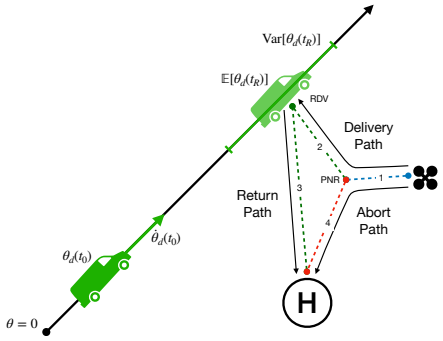


Fig. 2: Overview of the problem setup at time instance t_0 for a single path. Uncertainty in driver behavior and path choice requires multiple plans. Each path has associated risk and cost. We plan a Point-of-No-Return to afford extra time for data acquisition.

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 ρ_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 2 illustrates the setup for a single path. In Section III-D we present the multi-path setup, which is illustrated in Figure 3.

The discretized single integrator dynamics of the UAS are

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

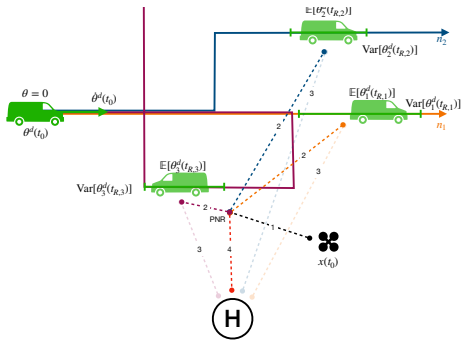


Fig. 3: Overview of the problem setup at time instance t_0 for multiple paths. Additional uncertainty in path choice: each path has it's own individual driver position uncertainty. Planning for all outcomes is intractable.

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:

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

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 α 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 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 [23], [24] can formalise this assumption. We can now present the problem formulation.

Problem 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*

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

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, ε 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 III we present the tools that solve Problem 1 by altering its different components. Although we never solve the Optimal Control Problem shown above explicitly, we reach an equivalent solution through multiple planning stages.

III. Methods

A. Driver Model Learning

This section discusses the learning component introduced in Section II. 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 [25], [26]. 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 paper, 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 [11]. 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 III-C and modern numerical integration methods to provide a computationally efficient integration procedure.

As described by Williams and Rasmussen [27], 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 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 $y_i \sim \mathcal{N}(d(x_i), \sigma_n^2)$ and $d \sim \mathcal{N}(0, K_d(x, x'))$ for a kernel K_d. The choice of kernel functions depends on the particulars of the problem. In this paper, we use the Matérn Kernel [28]. 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

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

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 [27], [29]. One of the main benefits towards the risk-averse efforts in this paper of using GPR is that estimates are computed in predictive distributions. The resulting distributions will provide tools for risk assessment in Sections III-C and III-D. 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 paper, we mitigate this issue with Sparse Gaussian Processes [27, Sec. 8.4]. In particular, we use Deterministic Training Conditionals (DTC) [30], [31]. 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 4 compares DTC to full GPR. Figure 5 shows a fitting performance comparison between the two methods.

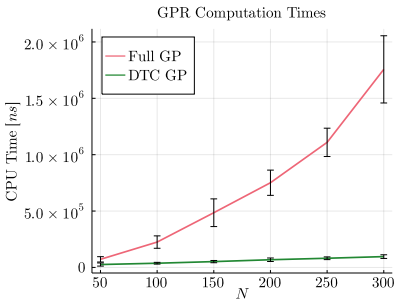


Fig. 4: Median computation times for full GPR and DTC GPR. Bars represent standard deviation, N indicates the amount of data points. At $N = 300$ a full GP regresses in a median time of 1.555ms, while DTC finishes in $88.827\mu\text{s}$. All computations done on a single core of a 2012 Intel Core i7.

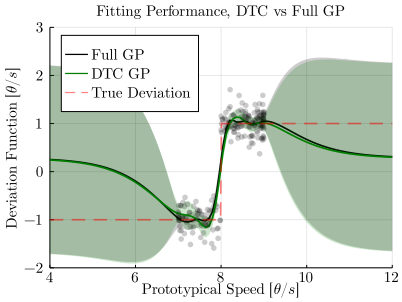


Fig. 5: Fitting performance: DTC performs effectively the same as a full GP in this application. The goal is to approximate the true deviation function from observed data. Shaded area indicates 95% confidence bounds.

B. 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. II, 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 (1) and (2).

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 (S_L), and the abort location (S_A), as shown in Figure 2. 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 III-A and elite samples from the importance sampling algorithm in Section III-C, 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.

C. 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 III-A to estimate where the rendezvous location is. We perform this optimization problem using the cross-entropy (CE) algorithm [32]. 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.

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

$$
\mathbb{E}[\theta_i^d(t)] = \theta^d(t_0) + \int_{t_0}^{t^{i,j}} \dot{\theta}_i^h(t) + d(\dot{\theta}_i^h(t))\, \mathrm{d}t, \qquad (4)
$$

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 paper. A parallel implementation would provide significant speedup for higher values of n_s than the ones used in this paper (we use $n_s = 5$, for reference). Using (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 III-B). Temporal constraints mean that two equally close rendezvous candidates have two different times for the UAS to reach that location. The energy dynamics (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 (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

$$\mathbf{E}^{i,j} = m_a(t_j - t_0)\left[\frac{1}{2}\left\|v_r^{i,j}\right\|_2^2 + \alpha\right] + m_b(t_l - t_j)\left[\frac{1}{2}\left\|v_l^{i,j}\right\|_2^2 + \alpha\right], \quad (5)$$

where $t_l = \sum_{k=1}^{3} t_k$ is provided by the MPC solution in Section III-B (in the absence of a solution in the first iteration we do not compute the second term of (5)), $\|\cdot\|_2$ denotes the 2-norm of a vector, and

$$v_r^{i,j} = \frac{|\mathbf{p}^{i,j} - x_0|}{t_j - t_0}, \quad v_l^{i,j} = \frac{|S_L - \mathbf{p}^{i,j}|}{t_l - t_j}, \quad (6)$$

where S_L is the landing location as described in Section III-B. For this paper, we set $l(n^{i,j}) = \mathbf{E}^{i,j}$. In Subsection III-C.2 we add a risk-averse component to the calculation of $\mathbf{E}$. These equations reflect the (generic) energy dynamics we consider in this paper, 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 $p^\star = p_{\mathrm{tgt}}(\theta^d_{\mathrm{tgt}}(t_{R,\mathrm{tgt}}))$ to be forwarded to the MPC planner. Let $S_e^{i,1}$ be the first column of S_e. Then for $\mathrm{tgt} \in \{1, \dots, N\}$ let $\mathrm{tgt} = \arg\min_i c\left(\mathcal{S}_e^{i,1}\right)$. 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 paper.

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 paper, however, selecting the best global sample is overly ambitious. If the driver chooses any path other than p_{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 paper, 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 paper, 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 [33] to quickly partially sort the array of samples.

2) Risk Assessment: Quantifying risk is the effort to determine a measure ρ that maps a set of random variables to a real number representing the probability or expected value of an undesirable outcome [14], [13]. 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 [15], (Conditional, Tail) Value-at-Risk [13, Sec. 3.3], and Downside Variance [13, 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 [34]. In this paper, we use two risk measures; ρ_r is the rendezvous risk measure used by the rating system to select samples of least risk, and ρ_d is the decision risk measure used in Subsection III-D 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 ρ_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. (6).

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

$$h^{i,j} = \gamma \int_{t_0}^{n^{i,j}} \Sigma_{d,i}(\dot{\theta}_i^h(t))\,\mathrm{d}t, \quad (7)$$

where $\gamma \in \mathbb{R}^+$ is a scaling factor. Equation (7) makes it so that $p_i(\mathbb{E}[\theta_i^d(t_j)]) \pm h^{i,j}$ represents a confidence interval in θ. 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 $r^{i,j} = \Gamma(\mathbf{p}^{i,j}, x_0)$

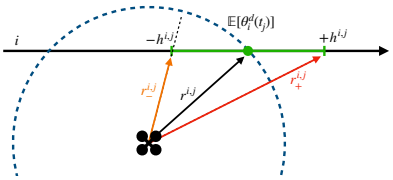


Fig. 6: Downside Risk as potential required range gain. The red outcome forces the UAS to spend more energy to meet the car. The extra energy is the downside potential, used as risk measure.

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 ρ_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 6 shows a visual representation of the different ranges. This risk measure is an approximation of Conditional Value-at-Risk (CVaR). In its common form, CVaR_γ represents the expected value of the γ-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 σ-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.

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. We update μ and Σ by taking mean and variance row-wise from $\mathcal{S}_e$ with $\mu_\mathcal{A} = \mathrm{Mean}(\mathcal{S}_e)$ and $\Sigma_\mathcal{A} = \mathrm{Var}(\mathcal{S}_e) + \lambda$, where $\lambda \in \mathbb{R}$ is a small positive number. The scalar parameter λ 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.

D. 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 1. The functions in Algorithm 1 and their correlated method are shown in Table I.

The overarching logic is the same as discussed in Section I. 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 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.

Algorithm 1: Mission Algorithm

$\mathcal{D} \leftarrow$ Initial Data
while $t_1 > \varepsilon$ **do**
  $d,\ \Sigma_d \leftarrow \mathrm{Regress}(\mathcal{D})$
  $\mathcal{S} \leftarrow \mathrm{Sample}(\mathcal{A}, N_s)$
  $p^\star,\ \mathcal{S}_e \leftarrow \mathrm{Rank}(\mathcal{S}, \mathbf{x}, \mathbf{t}, d, \Sigma_d)$
  $\mu_\mathcal{A},\ \Sigma_\mathcal{A} \leftarrow \mathrm{UpdateParameter}(\mathcal{S}_e)$
  $\mathbf{v}, \mathbf{x}, \mathbf{t} \leftarrow \mathrm{MPC}(p^\star, \mathbf{x})$
  Send Control Input $\mathbf{v}$ to UAS
  $\mathcal{D} \leftarrow \mathrm{Append}(\text{Sensor Data}, \mathcal{D})$
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. III-A
Sample	Returns N_s samples from $\mathcal{A}$
Rank	Ranking Procedure in Sec. III-C
UpdateParemeter	Updates parameters of $\mathcal{A}$ as in Sec. III-C.3
MPC	Computes MPC control inputs as in Sec. III-B
Append	Appends new sensor data to $\mathcal{D}$

TABLE I: Correlation between Algorithm 1 and this paper's methods.

As discussed in Section III-C, for online optimization purposes we approximate CVaR_γ. When deciding whether to abort or attempt a rendezvous, however, we can afford an expensive one-time computation of CVaR_γ. We briefly define CVaR_γ 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: $\mathrm{VaR}_\gamma(\mathbf{X}) = \inf\{x \in \Omega_\mathbf{X} : F_\mathbf{X}(x) \geq \gamma\}$ and

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

where $\gamma \in [0, 1]$ is a real-valued quantile. For the purposes of this paper, 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 μ_d and Σ_d for each of the paths' optimal rendezvous location.

IV. Results

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

A. Learning Performance

We start by presenting results on the learning algorithm in Section III-A. The learning problem seeks to find a deviation function d that captures the driver behavior. Figures 7 and 8 show two points in time. Figure 7 is taken after 10s of data

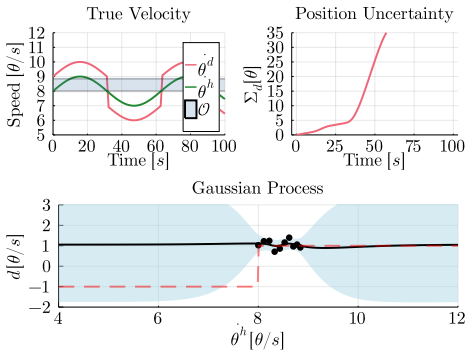


Fig. 7: GPR learning process snapshot after 10s. Shaded area indicates 95% confidence bounds.

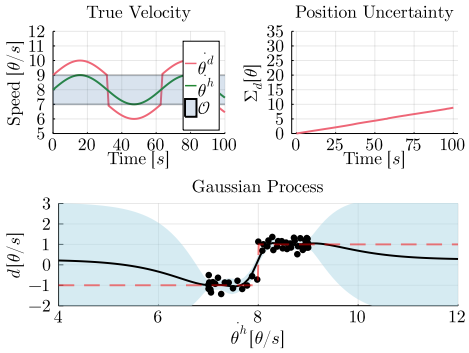


Fig. 8: GPR learning process snapshot after 50s.

gathering, and Figure 8 after 50s. The nature of $\dot{\theta}^h$ ensures that in Figure 7, $\mathcal{O}$ 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 8 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 $\dot{\theta}^d(t) = \dot{\theta}^h(t) + \text{sign}(\dot{\theta}^h(t) - 8)$.

B. Importance Sampling

Here, we present results on the importance sampling algorithm in Section III-C. Since an analytical form of the optimal rendezvous location does not exist, we 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 9 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.

C. Full Planning Stack

Consider the map in Figure 10. 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

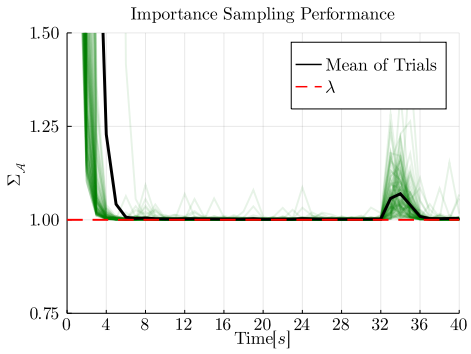


Fig. 9: Average convergence rate (black) of $\Sigma_{\mathcal{A}}$ for 100 trials (green). At $t = 10\pi$s, $\mathcal{O}$ starts to cover new information. The shift in the learned driver behavior causes the sampling algorithm to react as indicated by the momentary increase in $\Sigma_{\mathcal{A}}$.

in this paper. Figure 11 compares the risk associated with the secondary path (that is, the path that is not p_{tgt}) for the two importance sampling strategies. As expected, using Worst First yields reduced risk. The choice of κ 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 12 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.

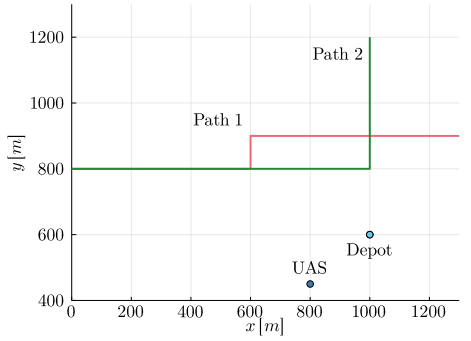


Fig. 10: Mission map representing a section of an urban grid.

V. 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 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

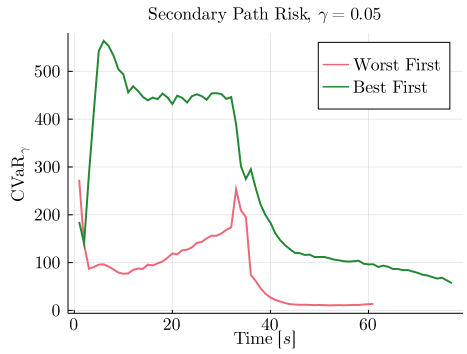


Fig. 11: Same mission parameters, equally seeded, for the two strategies proposed in Section III-C. Plot terminates when $t_1 < \varepsilon = 1$. Worst First finds trajectories with least risk should the driver choose a path $p \in \mathcal{P} \setminus p_{\text{tgt}}$.

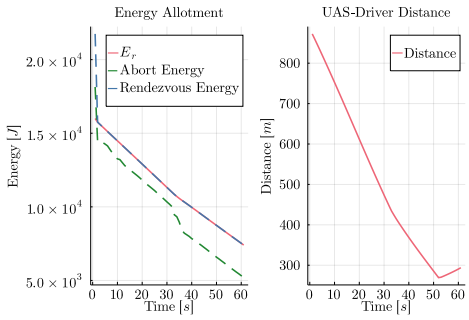


Fig. 12: Results using Worst First strategy. The algorithm uses all available energy to try and minimize risk. Distance increase towards the end is due to path geometry.

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.

REFERENCES

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

[2] G. B. Haberfeld, A. Gahlawat, and N. Hovakimyan, "Risk sensitive rendezvous algorithm for heterogeneous agents in urban environments," *arXiv:2002.05749*, 2020.

[3] 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.

[4] D. Bernardini and A. Bemporad, "Stabilizing model predictive control of stochastic constrained linear systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1468–1480, June 2012.

[5] J. A. Primbs and C. H. Sung, "Stochastic receding horizon control of constrained linear systems with state and control multiplicative noise," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 221–230, 2009.

[6] D. Mayne, M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219 – 224, 2005.

[7] 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.

[8] G. Schildbach, L. Fagiano, C. Frei, and M. Morari, "The scenario approach for stochastic model predictive control with bounds on closed-loop constraint violations," *Automatica*, vol. 50, no. 12, p. 3009–3018, Dec 2014. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2014.10.035

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

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

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

[12] 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.

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

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

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

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

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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," 2019, pp. 1–10.

[26] 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.

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

[28] 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.

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

[30] 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.

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

[32] 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.

[33] 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.

[34] 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.