2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

GP-based Runtime Planning, Learning, and Recovery for Safe UAV Operations under Unforeseen Disturbances

Esen Yel and Nicola Bezzo

***Abstract*— Autonomous vehicles are typically developed and trained to work under certain system and environmental conditions defined at design time and can fail or perform poorly if unforeseen conditions such as disturbances or changes in model dynamics appear at runtime. In this work, we present a fast online planning, learning, and recovery approach for safe autonomous operations under unknown runtime disturbances. Our approach estimates the behavior of the system with an unknown model and provides safe plans at runtime under previously unseen disturbances by leveraging Gaussian Process regression theory in which a model is continuously trained and adapted using data collected during the autonomous operation. A recovery procedure is event-triggered any time a safety constraint is violated to guarantee safety and enable learning and replanning. The proposed framework is applied and validated both in simulation and experiment on an unmanned aerial vehicle (UAV) delivery case study in which the UAV is tasked to carry an a priori unknown payload to a goal location in a cluttered/constrained environment.**

I. Introduction

Autonomous unmanned aerial vehicles (UAVs) and in particular multi-rotor vertical take-off and landing (VTOL) vehicles are rapidly finding their way into our society especially for transportation/delivery applications. In fact, thanks to their design and capabilities, they provide fast transportation between different locations and as demonstrated recently, they can be used to carry packages and even for medicine delivery. However, real-world operations bring several challenges, often not known a priori: state estimation uncertainties such as deprecated localization in GPS denied environments, external disturbances like wind, changes in model dynamics and system aging, component failures, and variation in payload distribution are some example of such factors that could lead to undesired conditions like the loss of stability and even collisions if not properly considered at design time or at runtime.

To deal with disturbances and noises appearing at runtime, data driven machine learning techniques have recently demonstrated considerable potential as an alternative to traditional model-based reachability techniques thanks to their computational efficiency. However, as the performance of such techniques highly depends on the training data, it becomes challenging to provide safety guarantees when the training data are limited or when the test data are outside of the training boundaries. It is thus essential to adapt and update the learning enabled components of the system with new data acquired at runtime all while guaranteeing safety.

For example, let's consider an autonomous delivery case study in which a UAV has to carry unknown loads between two points while flying through narrow corridors and windows and around and over obstacles of varying heights and dimensions, as depicted in Fig. 1. The load will affect the motion of the vehicle creating deviations from the desired planned trajectory and could lead to unstable and unsafe conditions (e.g. a collision), if not properly considered during planning.

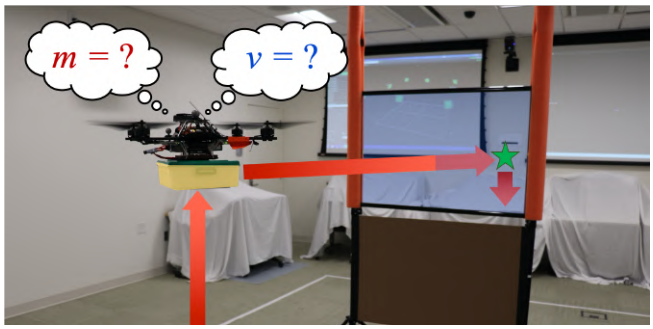


Fig. 1. Pictorial representation of the problem of planning with unknown payload.

In this work, we propose a recursive dynamic Gaussian Process (GP) regression-based framework to detect disturbance changes and learn and adapt the system performance at runtime to guarantee safety (i.e., no collision) and liveness (i.e., completion of the task) conditions. To better drive the explanation of our technique, we tackle a UAV pick-up and delivery problem (e.g.,package delivery) in which the payload is treated as a disturbance since the system might be required to carry different a priori unknown payloads during real-world applications. As the user may not know system dynamics and may not have access to the controller especially when off-the-shelf commercial vehicles are used, changes in the payload will affect the system behavior and cause the system to deviate from its desired behavior, and potentially lead to unsafe states. For this reason, the system needs to estimate the disturbance and its effects and adapt its plan accordingly and quickly at runtime. We propose to use GP regression built on offline training data to estimate the disturbance and deviations at runtime. As the training could be limited (not complete or shifting), the system may not behave as expected at runtime when the conditions are outside of the training boundaries. To deal with these situations, we consider an event-triggered runtime recovery approach and propose a method to update the trained GP model with the new information gained at runtime to improve system performance and safety. To summarize, with this work, we aim to solve the following challenges:

- *how to estimate, recover, update, and adapt the decision of a learning enabled component online, using data obtained at runtime, while guaranteeing the autonomous*

Departments of Engineering Systems and Environment and Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904, USA {esenyel, nbezzo}@virginia.edu

system safety and improving its future performance.

The contribution of this work is four-fold: 1) we develop a GP-based approach to estimate the unknown disturbance and the corresponding deviations of a system resulting from such disturbance; 2) we propose an approach to adapt system performance (i.e., speed) along the planned trajectory based on environmental constraints and the GP-based estimation 3) we propose an event-triggered recovery method for situations in which the system does not behave as expected due to the lack or inaccuracy of the training data, and 4) we propose an approach to dynamically update the GP model to improve runtime performance as the system obtains information.

The rest of the paper is organized as follows: related works about online learning and payload disturbance rejection are presented in Section II. The problem is then formally defined in Section III and we present our proposed fast deviation estimation and online recovery approach in Section IV. We validate our technique with simulations and experiments in Sections V and VI respectively and finally provide concluding remarks and future work in Section VII.

II. Related Work

As UAVs become popular to provide solutions to real world problems, it becomes critical to guarantee their safety and performance under unforeseen situations like noises, disturbances and model dynamics changes. Model-based reachability analysis techniques such as Hamilton-Jacobi reachability [1], funnels [2] and robust control invariant tubes [3] are widely used in robotics applications to provide safety guarantees under unknown disturbances and to provide safe learning for uncertain systems [4]. Even though traditional, model-based reachability techniques are useful to provide safety bounds, they require model knowledge and their computational complexity makes them challenging to be used at runtime. Alternatively, data-driven machine learning approaches have been used to provide safety and performance guarantees. In [5], neural networks are used to approximate the optimal controller designed by reachability analysis. Gaussian Processes are also powerful learning-based tools which are used to estimate the system disturbances at runtime to improve system performance [6] and to provide safe controller tuning [7]. Both Gaussian Processes-based model learning [8] and Bayesian linear regression [9] show good performance in repetitive tasks with online learning to estimate the changing model dynamics and improve the system performance. In our previous work [10], we used a pre-trained GP regression model to estimate reachable sets at runtime for a system moving in open loop under intermittent sensing while the testing conditions are bounded by the training conditions. In this work, we extend this idea and introduce a GP regression-based adaptive approach to provide safe planning under unknown disturbances at runtime. As the performance of the learning enabled systems highly depend on the training data, we propose a recursive learning and online recovery approach to update the pre-trained GP regression model with data acquired at runtime, which allows the system to improve over time and work outside of the training boundaries.

We apply our approach to solve a package delivery problem with unknown payload. As this problem is gaining importance with the increased use of UAVs for delivery purposes, it has been addressed by the robotics community by estimating the system model under payload disturbance and designing controllers and trajectories to improve performance [11], [12], [13]. Different from these works, we solve the payload disturbance problem when the system dynamics are not available and the controller can not be accessed.

III. Problem Formulation

In this work, we are interested in finding an online adaptive policy for safe UAV navigation under unforeseen disturbances.

A general UAV with state $\boldsymbol{q}$ can be modeled as a non-linear system of the form $\dot{\boldsymbol{q}} = f(\boldsymbol{q}, \boldsymbol{u})$. Typical control techniques like PID controllers are designed to control $\boldsymbol{u}$ to minimize tracking error $\boldsymbol{e} = \boldsymbol{q}_{des} - \boldsymbol{q}$ for the given system dynamics. If a disturbance $\boldsymbol{d}$ is present and unknown, the system dynamics change: $\dot{\boldsymbol{q}} = \tilde{f}(\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{d})$. If the controller is designed under no disturbance assumption, or to work under different disturbance boundaries, the tracking performance of the system will be deteriorated leading to possible unsafe states (e.g., a collision). By changing the motion planning of the system, however, it could be possible to perform safe operation under limited performance (e.g., reduced speed).

This problem is formally defined as follows:

Problem 1: ***Runtime Fast Adaptive Safe Planning:*** Given a UAV tasked to navigate in a cluttered environment, under the effect of an unknown disturbance $\boldsymbol{d}$, find an online policy to quickly adapt its motion plan, and constantly update the prediction to satisfy the following safety constraint:

- *Safety Constraint:* The UAV must avoid collision with the obstacles in the environment during its motion:

$$\|\boldsymbol{p}(t) - \boldsymbol{o}_i\| \geq \boldsymbol{\delta} \quad \forall t \in [t_0, T], \forall i \in \{1, \ldots, n_o\} \quad (1)$$

  where $\boldsymbol{p}(t) = [x, y, z] \subset \boldsymbol{q}$ is the 3D position of the vehicle at time t. Similarly, $\boldsymbol{o}_i$ is the 3D position of the i^{th} obstacle, with n_o the number of obstacles in the environment and $\boldsymbol{\delta}$ is the minimum safe distance to the obstacle considering the dimensions of the quadrotor.

In order to solve Problem 1, we leverage GP-based theory to predict the future states of the system, which requires to identify the current configuration of the system, as formally defined in the following problem:

Problem 2: ***Model Disturbance Estimation:*** Given a system with a pre-trained learning enabled component to estimate future states of the system, find a policy to quickly identify the disturbance in the model at runtime which causes the system to deviate from its nominal behavior.

Case Study: ***Payload Disturbance Estimation:*** While our proposed framework is general for different types of systems and runtime disturbances, for the sake of clarity in this paper we will focus primarily on payload pick-up/drop-off UAV operations in which the disturbance is the unknown payload lifted by the vehicle at runtime: a UAV is tasked to lift and carry an a priori unknown load to a goal while avoiding obstacles in a cluttered environment. The low-level controller of the UAV is considered as a black box while within the high level planner different waypoints as well as the velocity of the UAV can be changed depending on the mission. We assume that the vehicle will minimize its mission time and

will try to travel to the goal with the highest speed possible. However, moving at high speed may make the system deviate from its desired behavior in particular in the z level due to the system dynamics and limited thrust as shown in Fig. 2, whereas reducing the speed can help the system to perform better tracking. Hence, the primary goal of this work is to find a policy to obtain the highest average speed for a UAV to avoid vertical obstacles while carrying an a priori unknown payload. From Problem 1, the safety constraint to respect is thus:

$$|z(t) - z_i^o| \geq \delta \quad \forall t \in [t_0, T], \forall i \in \{1, \dots, n_o\} \tag{2}$$

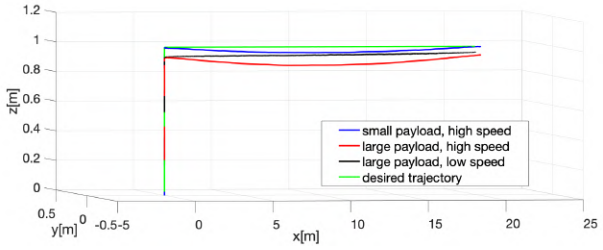


Fig. 2. System behavior comparison with different payload disturbances and speeds. To minimize deviations while carrying a large load it is necessary to reduce speed.

IV. Gaussian Process-based Safe Planning, Recovery and Adaptation

In this section, we describe our framework for safe planning, recovery, and adaptation in UAV navigation operations under unknown payload disturbances. Our proposed approach consists of offline and online stages, as shown in Fig. 3. During the offline stage, a set of trajectories consisting of different lengths, goals, and average speeds are run on the UAV with a bounded payload disturbance. For each trajectory, the maximum deviation from the desired z level, and the time taken by the UAV to take off from the ground are recorded.

A GP-based regression is then trained and considered at runtime to estimate the maximum deviation from the desired trajectory and in turns compute the maximum average speed to maintain during the operation and avoid obstacles along the way. We choose to use GP regression to solve these problems because of its ability to provide confidence intervals which makes it easier to handle noise and uncertainties as well as adapt its model at runtime. However, the GP regression may result in inaccurate estimations when the test conditions at runtime are outside of the training boundaries, or due to errors in payload estimation. In such cases, a recovery procedure is triggered at runtime when an unsafe situation is detected by monitoring the system's position and velocity in z direction. The GP regression model is then updated with the data acquired at runtime to improve future decisions.

In the following section, we explain the proposed GP-based z deviation estimation approach.

A. Gaussian Process Regression for Deviation Estimation

In order to enable online prediction of deviations during an operation, a GP regression model is trained from a rich library of trajectories under different average velocities and payloads. We will refer to such GP model as GP^d to differentiate from the one for payload GP^p introduced in Section IV-C. Since the controller is treated as a black box and cannot be accessed and adapted according to the payload, the deviation from the desired height increases with the payload and average speed, as demonstrated in Fig. 2. The maximum deviation from the desired height for an average speed $\bar{v}$ and for a payload mass m_p is extracted from the recorded trajectory as follows:

$$d_z^m(\bar{v}, m_p) = \max_{t \in [T_h, T_h + T_g]} \max_{\boldsymbol{p}_g \in \mathcal{P}_g} |z_{\tau(g,\bar{v})}(t) - z(t)| \tag{3}$$

where $z_{\tau(g,\bar{v})}(t)$ is the desired height of the trajectory with average speed $\bar{v}$, T_h is the vertical trajectory duration, T_g is the duration of the trajectory to the goal location, $\boldsymbol{p}_g$ is the position of the goal g and $\mathcal{P}_g$ is the set of all goal positions used in training.

To estimate the maximum deviation from the desired height based on the estimated payload mass and the average speed of the planned trajectory, we leverage Gaussian Process regression theory. Training inputs for the GP^d regression consist of the payload mass and average speed values: $\boldsymbol{x} = [\boldsymbol{x}_{1,1} \cdots \boldsymbol{x}_{i,j} \cdots \boldsymbol{x}_{n_v,n_m}] \in \mathbb{R}^{(n_v n_m) \times 2}$ where $\boldsymbol{x}_{i,j} = [\bar{v}_i, m_{p,j}]^T, \forall i \in [1, \cdots, n_v], \forall j \in [1, \cdots, n_m]$, where n_v and n_m are the number of average speed and payload mass values in the training set. The output of the training is the maximum deviation value: $\boldsymbol{d_z} = [d_z^m(\boldsymbol{x}_{1,1}), \cdots, d_z^m(\boldsymbol{x}_{i,j}), \cdots, d_z^m(\boldsymbol{x}_{n_v,n_m})] \in \mathbb{R}^{(n_v n_m)}$. Given the training set $\mathcal{D} = [\boldsymbol{x}, \boldsymbol{d_z}]$, by the definition of GP, training and test outputs follow a joint Gaussian distribution:

$$\begin{bmatrix} \boldsymbol{d_z} \\ d_z^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x}) \\ \mu(\boldsymbol{x}^*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I} & \boldsymbol{K}_* \\ \boldsymbol{K}_*^T & \boldsymbol{K}_{**} \end{bmatrix} \right) \tag{4}$$

where $\boldsymbol{K} = \begin{bmatrix} k(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,1}) & \cdots & k(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{n_v,n_m}) \\ \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_{n_v,n_m}, \boldsymbol{x}_{1,1}) & \cdots & k(\boldsymbol{x}_{n_v,n_m}, \boldsymbol{x}_{n_v,n_m}) \end{bmatrix}$, $\boldsymbol{K}_* = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}^*) & \cdots & k(\boldsymbol{x}_{(n_v n_m)}, \boldsymbol{x}^*) \end{bmatrix} \in \mathbb{R}^{(n_v n_m)}$ and $\boldsymbol{K}_{**} = k(\boldsymbol{x}^*, \boldsymbol{x}^*) \in \mathbb{R}$. σ_ε^2 is the noise level associated with the observations, μ is the mean function and k is the covariance function and $\boldsymbol{x}^* = [\bar{v}^*, m_p^*]^T \in \mathbb{R}^2$ is the test input. The widely used Matern kernel is considered here as a covariance function and the hyperparameters are set by optimizing the marginal likelihood [14]. The predictive posterior distribution of d_z^* is also a Gaussian distribution:

$$p(d_z^* | \boldsymbol{x}^*, \boldsymbol{x}, \boldsymbol{d_z}) \sim N(\mu_*, \sigma_*^2) \tag{5}$$

where μ_* and σ_*^2 are calculated as follows:

$$\mu_* = \mu(\boldsymbol{x}^*) + \boldsymbol{K}_*^T (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} (\boldsymbol{d_z} - \mu(\boldsymbol{x})) \tag{6}$$

$$\sigma_*^2 = \boldsymbol{K}_{**} - \boldsymbol{K}_*^T (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} \boldsymbol{K}_* \tag{7}$$

Finally, the maximum deviation for a test input $\boldsymbol{x}^*$ is estimated by using the upper limit of the 95% confidence interval which is computed by:

$$\bar{d_z}^*(\bar{v}^*, m_p^*) = \mu_* + 2\sigma_* \tag{8}$$

with μ_* and σ_* obtained from (6) and (7) respectively.

To create a training set, given $m_q = 176$g the mass of the UAV used in the simulations, the trajectories are executed with four different payload masses $m_{p,1} = 0 \times m_q = 0$g (blue curve), $m_{p,2} = 0.25 \times m_q = 44$g (red curve),

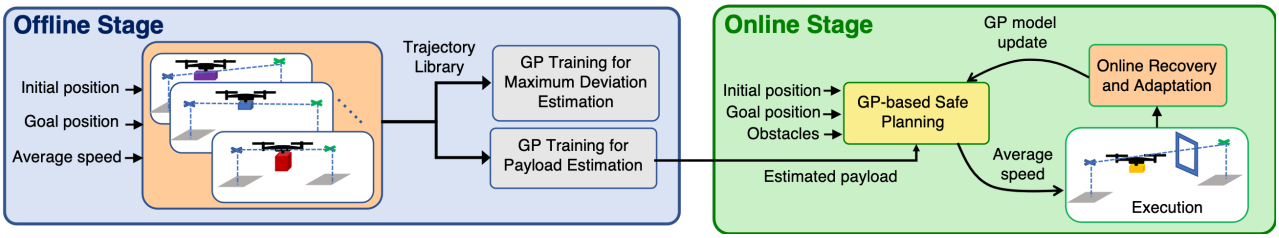
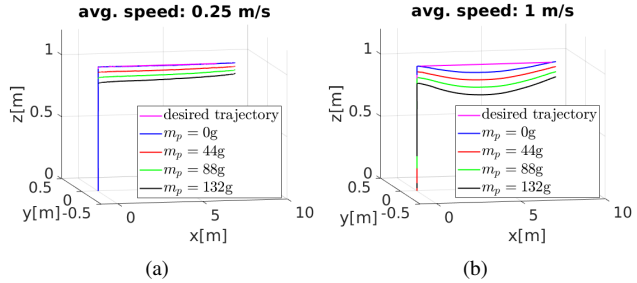


Fig. 3. Architecture of the proposed approach.

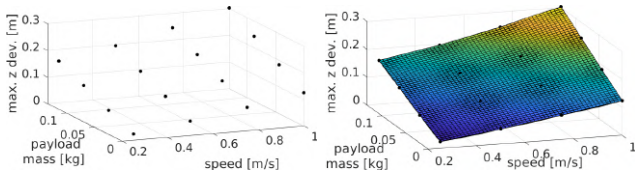
$m_{p,3} = 0.5 \times m_q = 88$g (green curve), $m_{p,4} = 0.75 \times m_q = 132$g (black curve) using a quadrotor model defined in [15]. The trajectory library includes minimum snap trajectories [16] with four different average speed values: $\bar{v} \in \{0.25, 0.50, 0.75, 1.00\}$m/s. Fig. 4 shows two examples of such training trajectories with a quadrotor tasked to reach first a 1m height and then move in the $+x$ direction for 8m under the four loads and two of the speeds specified above: $\bar{v} = 0.25$m/s in Fig. 4(a) and $\bar{v} = 1$m/s in Fig. 4(b).



(a) (b)

Fig. 4. Sample trajectories from the training library running with average speed of a) 0.25m/s, and b) 1.0m/s under four payload disturbances.

The trajectory library contains 200 different trajectories executed with the four different payload masses and Fig. 5(a) shows the recorded maximum deviation values for each average speed and payload mass value in the training set. The results of the GP regression training is shown in Fig. 5(b).



(a) Maximum z deviation as a function of payload and average speed.

(b) GP regression results for maximum deviation estimation obtained using GPML toolbox [17].

Fig. 5. Offline training for maximum deviation estimation at runtime.

The estimated maximum deviation is then used at runtime to plan and adapt the average speed of the UAV trajectory to reach its desired goal fast without violating the safety constraints, as explained in the following section.

B. Fast, Runtime Speed Adaptation, Online Recovery and Learning

As noted in the previous section, a payload has two effects on the UAV: 1) it slows down its take-off procedure bringing the vehicle to a lower height from the desired height and 2) with larger speeds, it creates more deviation from the desired trajectory while moving toward its goal. Thus, as soon as the UAV has reached a certain height before it starts moving toward the goal, it needs to quickly decide the average speed to maintain during the trajectory in order to reach the desired goal as fast as possible while maintaining safety at all times. To this end, the trained GP^d obtained in the previous section is used to determine the maximum speed of the UAV to safely navigate in the environment and avoid collisions, as follows:

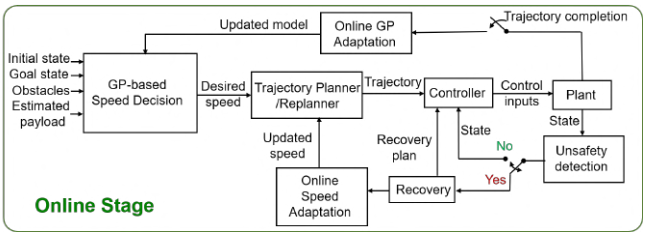


Fig. 6. Architecture of the proposed runtime speed adaptation, online recovery and learning approach.

$$\bar{v}^* = \max_{\bar{v} \in \mathcal{V}_s(\bar{m}_p)} \{\bar{v} | \bar{d}_z(\bar{v}, \bar{m}_p) < \zeta\} \tag{9}$$

where $\bar{m}_p$ is the estimation of the payload mass (discussed in the next section), $\mathcal{V}_s(\bar{m}_p)$ is the set of stable speeds for the given payload, and ζ is the maximum allowed deviation:

$$\zeta = \min_{i \in [0, n_o], t \in [T_h, T_h + T_g]} |z_\tau(t) - z^{o,i}| \tag{10}$$

where $z_\tau(t)$ is the desired height at time t and $z^{o,i}$ is the z position of the i^{th} obstacle located along the desired trajectory. Once the desired average speed is picked, a trajectory is planned to the desired goal location, and the system starts to move following the desired trajectory. However, because the training is limited, it does not cover all the possible runtime conditions. When the test input at runtime is outside of the training boundaries, the system may behave different than expected. For example, when the system carries a large payload and choose to travel with high speed, it may lose height and become unsafe or if the deviation estimation is inaccurate the vehicle may collide with the obstacles. To prevent such situations, we follow an online recovery and adaptation procedure which is summarized in Fig. 6.

To detect inaccuracies in prediction, changes in the z velocity and position of the system are monitored at runtime: A high negative $\dot{z}$ is considered unsafe because it causes the system to lose height and become unsafe and a low z close to the clearance level is also considered unsafe because it can cause the system to collide with the obstacles. These situations trigger a recovery mode in which the vehicle stops its current go-to-goal task and switches to a hovering mode to bring the system to a certain height and decide a new speed. To guarantee that recovery is possible, the maximum payload and speed limits were tested offline.

If the recovery is performed because the z level of the UAV becomes lower than the allowed level to pass the obstacle, a new speed is chosen by considering a lower deviation threshold. If the recovery is performed due to high $-\dot{z}$, the detected unsafe speed is added to the training set and labeled unsafe.

The set of stable speeds is decided using Support Vector Machine (SVM) classification trained with the training input $\boldsymbol{x} \in \mathbb{R}^{(n_v n_m)\times 2}$ and training output $\boldsymbol{s} \in \mathbb{R}^{(n_v n_m)}$ which is initially set to $\boldsymbol{s} = \vec{\mathbf{1}}$ (i.e.,safe) Therefore the initial set of stable speeds consists of all admissible speed values.

$$\boldsymbol{x}_{svm} = [\boldsymbol{x}, [\bar{v}, \bar{m}_p]^T], \boldsymbol{s}_{svm} = [\boldsymbol{s}, 0] \tag{11}$$

After retraining the SVM with the updated dataset that includes both safe and unsafe trajectories, a new speed value is picked using (9). If the selected speed is still detected to cause the system to deviate more then the permitted threshold, a recovery is triggered and a lower speed is considered. Once the system completes its trajectory, the estimated payload, final average speed ($\bar{v}_f$) and the maximum recorded deviation are added to the training set and used to retrain the GP online for more accurate decisions in future iterations.

$$\boldsymbol{x}^{gp} = [\boldsymbol{x}, [\bar{v}_f, \bar{m}_p]^T]], \boldsymbol{d}_z^{gp} = [\boldsymbol{d}_z, d_z(\bar{v}_f, \bar{m}_p)] \tag{12}$$

The final speed and the payload estimation pair is labeled as safe and SVM model is updated as well. By following this online recovery procedure, the system is able to use the pre-trained learning enabled components to make decisions at runtime, even if the test conditions are different from the training conditions. Furthermore, the system is able to improve the decisions over time thanks to the online learning.

The drawback of such approach is that the computational complexity of the GP regression and SVM classification increases with the data size. However, this drawback can be overcome by keeping the training data size bounded. This can be achieved by removing the redundant data collected at runtime. In this work, the size of the data set used by the learning enabled components was within the range of fast training and estimation, therefore we left data pruning as a future work.

In order to be able to use the GP-based deviation estimation model explained in Section IV-A and to update this model at runtime as explained in Section IV-B, the payload disturbance should be estimated because input to GP^d. We again leverage GP regression theory to estimate the payload and explain the details in the next section.

C. Gaussian Process Regression for Payload Estimation

The UAV is tasked to reach a desired height in a certain amount of time after picking up an object of unknown mass. In order to estimate the amount of payload m_p that the UAV is carrying, we leverage again GP regression based on a library of previously collected data. To train such GP, GP^p, the UAV is tasked to reach a desired height following a vertical trajectory for a fixed time duration under different payloads and measurement and process noises.

Since the internal controller is treated as a black box, due to the thrust-to-weight ratio properties inherent of UAVs, the increased mass of the vehicle will affect the time it takes to reach the desired height and thus there is a direct correlation between take-off time and weight.

During training, for each m_p value, the time which takes the system to pass a given z level is obtained from the collected data as follows:

$$t_h = \min_{t\in[0,T_h]} \{t | z(t) \geq z_h\} \tag{13}$$

where $z_h \geq 0$ is the user defined detection height to estimate the weight of the load and T_h is the vertical trajectory duration. These observations are recorded in the training set: $\boldsymbol{t_h} = [t_{h,1}, \cdots, t_{h,n}]$ and $\boldsymbol{m_p} = [m_{p,1}, \cdots, m_{p,n}]$ where n is the size of the training set. Given the training set of the collected observations of $\mathcal{M} = \{\boldsymbol{t_h}, \boldsymbol{m_p}\}$, the goal is to predict the payload mass for a new input t_h^* by drawing m_p^* from the posterior distribution $p(m_p^*|\mathcal{M})$. By definition of GP [14], the training set output $\boldsymbol{m_p}$ and the test output m_p^* are joint multivariate Gaussian distributed:

$$\begin{bmatrix} \boldsymbol{m_p} \\ m_p^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{t_h}) \\ \mu(t_h^*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}^m + \sigma_\epsilon^2 \boldsymbol{I} & \boldsymbol{K}_*^m \\ (\boldsymbol{K}_*^m)T & \boldsymbol{K}_{**}^m \end{bmatrix} \right) \tag{14}$$

where σ_ϵ^2 is the noise level associated with the observations, μ is the mean function and k is the covariance function. $\boldsymbol{K}^m \in \mathbb{R}^{n\times n}$ has entries $\boldsymbol{K}_{(i,j)}^m = k(t_{h,i}, t_{h,j})$ for $i,j \in \{1, \cdots, n\}$, $\boldsymbol{K}_*^m = \begin{bmatrix} k(t_{h,1}, t_h^*) & \cdots & k(t_{h,n}, t_h^*) \end{bmatrix} \in \mathbb{R}^{n\times 1}$ and $\boldsymbol{K}_{**}^m = k(t_h^*, t_h^*) \in \mathbb{R}$ with a Matern kernel [14] as a covariance function. The predictive posterior distribution of m_p^* is also a Gaussian distribution:

$$p(m_p^*|t_h^*, \boldsymbol{t_h}, \boldsymbol{m_p}) \sim N(\mu_*^m, (\sigma_*^m)^2) \tag{15}$$

with the following mean and variance:

$$\mu_*^m = \mu(t_h^*) + (\boldsymbol{K}_*^m)^T (\boldsymbol{K}^m + \sigma_\epsilon^2 \boldsymbol{I})^{-1} (\boldsymbol{m_p} - \mu(\boldsymbol{t_h})) \tag{16}$$

$$(\sigma_*^m)^2 = \boldsymbol{K}_{**}^m - \boldsymbol{K}_*^T (\boldsymbol{K}^m + \sigma_\epsilon^2 \boldsymbol{I})^{-1} \boldsymbol{K}_*^m \tag{17}$$

Once the vehicle lifts an object from the ground at runtime, the payload mass is estimated based on the time it takes for it to reach z_h level using the upper limit of the confidence interval calculated as follows:

$$\bar{m}_p^* = \mu_*^m + 2\sigma_*^m \tag{18}$$

where μ_*^m and σ_*^m are calculated according to (16) and (17) respectively. This mass estimation is then used as one of the inputs for the GP^d presented in Section IV-A to estimate the deviations from the desired height and make decisions about the optimal speed to maintain along a trajectory.

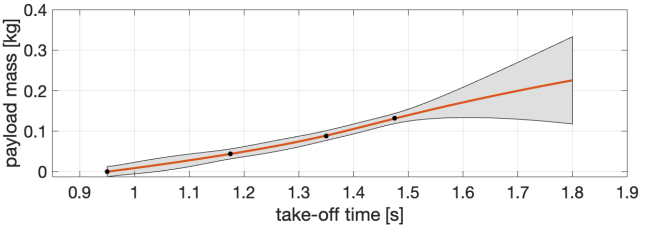


Fig. 7. GP regression results for payload estimation based on the take-off time using GPML toolbox [17].

The GP^p regression model trained for payload estimation is shown in Fig. 7. Training was executed under the same conditions used for training GP^d and with a detection height threshold set to $z_h = 0.05$m.

V. SIMULATIONS

We validate the proposed online adaptation, recovery and learning approach with a quadrotor UAV pick-up and a drop-off case study: specifically, the vehicle is tasked to lift and carry objects with a priori unknown weights between two designated locations ($\boldsymbol{p}_0 = [0,0,0]$m, $\boldsymbol{p}_1 = [20,0,0]$m) flying at $z = 1$m level through windows in the middle. We performed 100 simulations with random window positions and observed safe behavior of the system. In this section, we show one case with different mass values where there are two windows in the environment located at $(x_w = 6, z_w = 0.62)$m and $(x_w = 12, z_w = 0.45)$m shown in Fig. 8 where z_w is the height of the obstacle to avoid and for the window shaped obstacle it is the height of the frame base. The quadrotor is required to pass through these windows to move between the designated locations.

When the vehicle picks up an object from the ground, the payload is estimated based on the time it takes to take off from the ground (i.e., to reach a height $z_h = 0.05$m) according to (18) using the pre-trained GP^p. After estimating the payload mass, the average speed to go to the goal is picked using (9), based on the GP^d. As the vehicle completes its trajectory, GP^d is updated with the data obtained at runtime. In Fig. 8(a), we show the actual path (blue curve) and the desired trajectory (magenta curve) of the quadrotor which is tasked to carry a 192.2g payload. The vehicle estimates the payload as 223.5g and decides to move to its goal with $\bar{v} = 0.7$m/s based on the pre-trained GP^d shown in Fig. 5(b). After successfully reaching the goal location, the quadrotor updates GP^d with the estimated payload mass and observed maximum deviation. The updated GP^d result is shown in Fig. 9(a). The estimated payload mass and speed pair is also labeled as safe and added to the safety data set shown in Fig. 10(a) where the added point is marked with a blue circle. In Fig. 8(b), we show the 3^{rd} task where the quadrotor carries a 200g payload which is estimated as 285.7g and it decides to go to its goal with a speed $\bar{v} = 0.85$m/s. After the vehicle starts moving towards its goal, its velocity in the $-z$ direction increases and the inconsistency detector discussed in Section IV-B triggers a recovery action. After completing the recovery action, the initial velocity and payload estimation pair is labeled as unsafe and added to the data set (marked with a black circle in Fig. 10(b)) and the vehicle decides to move with average velocity $\bar{v} = 0.57$m/s which is picked according to the updated SVM safety decision model. After the UAV is able to complete its trajectory safely with the new average speed value, GP^d is updated with this speed value and estimated payload mass pair as shown in Fig. 9(b).

In Fig. 8(c), we show the 21^{st} task where the quadrotor UAV carries the same load as in the third task. Thanks to the data acquired online and updated GP regression and SVM decision models, the vehicle decides to travel with average velocity 0.41m/s and is able to complete its trajectory without performing a recovery action. The safety decision model with all the data collected offline and online is shown in Fig. 10(c). As the system obtains more data, it becomes able to make better decisions about the speed avoiding recovery operations. The GP^d after 21 tasks is presented in Fig. 9(c) and because of the uncertainties in the data collected online, the confidence interval of the GP regression becomes larger. Because this work considers the safety of the vehicle as a priority, both the deviation estimation and safety decisions are performed in a conservative way as demonstrated in this set of simulations. The vehicle is able to complete its tasks without going into unsafe states, and improve its decision making over time thanks to our online recovery and learning approach.

These demonstrated simulations were run on an Intel Core i9-9900K CPU at 3.60GHz taking 62.83ms on average to update GP^d and 0.97 ms on average to estimate the maximum deviation estimations using the trained GP regression model. SVM decision model updates took 4.82ms on average. These results demonstrate that the proposed GP based approach is suitable for online applications.

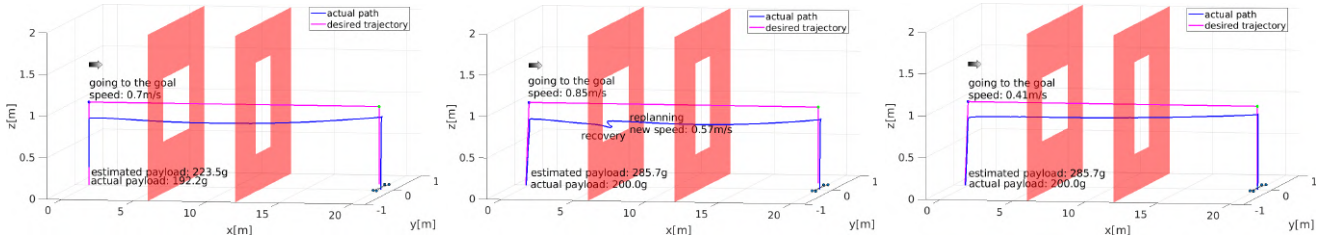
VI. EXPERIMENTS

Experiments were performed on an AscTec Hummingbird quadrotor UAV tasked to fly with different payload disturbances and through a window located at varying distances from the robot's initial position. A Vicon motion capture system was used to monitor the state of the quadrotor. The control commands to the quadrotor are communicated through the Robot Operating System (ROS). We used Matlab GPML Toolbox [17] to perform GP regression and the interface between Matlab and ROS was established through the Matlab ROS Toolbox.

The training data for payload and deviation estimation for GP regression were collected by running a trajectory with five different average speed values: $\mathcal{V} = \{0.25, 0.50, 0.75, 1.00, 1.25\}$m/s, and with three different payload disturbance: $\mathcal{M}_p = \{100, 300, 500\}$g. In order to perform payload estimation, the take-off time of the trajectories in the training set is measured using (13) for $z_h = 0.2$m. GP^p results for payload estimation are shown in Fig. 11. As the purpose of the quadrotor is to reach its goal location as quickly as possible, it does not wait until it reaches a certain height to create a trajectory to the goal. During the online stage, the system trains a GP^d using the data collected at the offline stage to estimate the deviation at a position where the system is required to pass through a narrow window. After the system starts moving vertically, the payload disturbance is estimated using (18) based on the observed take-off time and the system plans a trajectory with an average speed decided using (9).

In Fig. 12, we compare the actual path and speeds followed by the UAV with different payloads and positions of the window. In Fig. 13(a), we demonstrate the experiment setup for a quadrotor starting its mission at $\boldsymbol{p}_0 = [-2,0,0]$m and tasked to reach the goal located at $\boldsymbol{p}_1 = [2,0,0]$m by flying at $z = 1.5$m height in an environment with a window shaped obstacle with 0.6m clearance. The overlapped sequence of snapshots for a quadrotor carrying 400g payload though a window located in three different locations: a) $x_w = -0.5$m b) $x_w = 0$m and in c) $x_w = 1$m at $z_w = 1.2$m are shown in Fig. 13. The payloads tested in these experiments were not included in the training set.

In Fig. 12(a), the window is positioned at $x_w = -0.5$m and the quadrotor is tasked to move with 50g, 200g, 400g and 550g payloads. When the payload is 50g (cyan curve), the

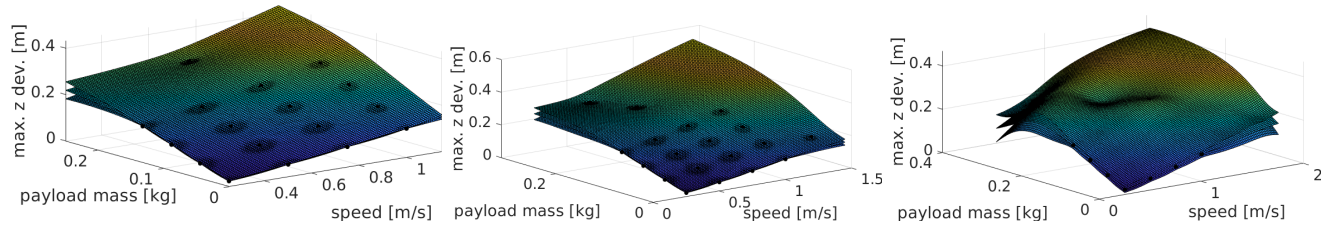


(a) 1st task with 192.2 g payload. The estimated payload is 223.5g.

(b) 3rd task with 200.0 g payload. The estimated payload is 285.7g.

(c) 21st task with 200.0 g payload. The estimated payload is 285.7g.

Fig. 8. Desired and actual trajectories of the quadrotor while it is performing a pick up/drop off task in an obstacle cluttered environment. The arrow shows the direction of the trajectory.

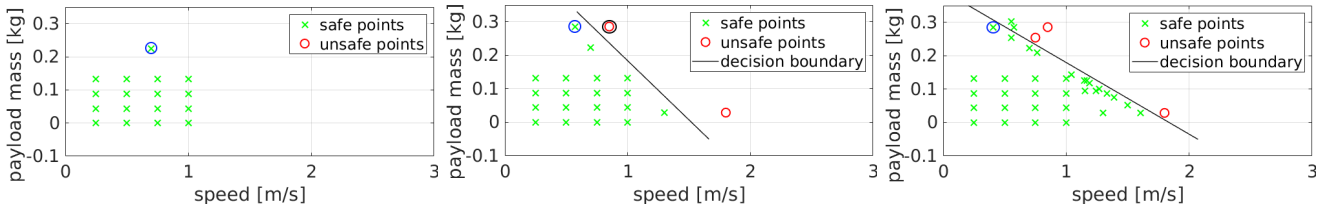


(a) Updated GP estimation after the 1st task.

(b) Updated GP estimation after the 3rd task.

(c) Updated GP estimation after the 21st task.

Fig. 9. Updated GP regression estimation models after each task with the data acquired at runtime.



(a) Updated safety decisions after the 1st task.

(b) Updated safety decisions after the 3rd task.

(c) Updated safety decisions after the 21st task.

Fig. 10. Updated SVM safety decision models after each task with the data acquired at runtime.

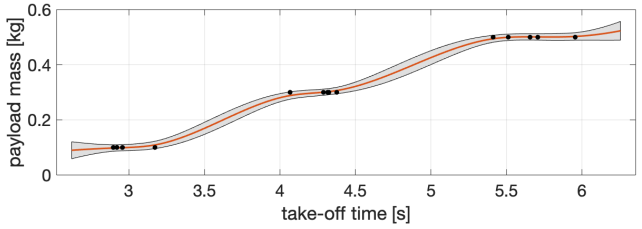
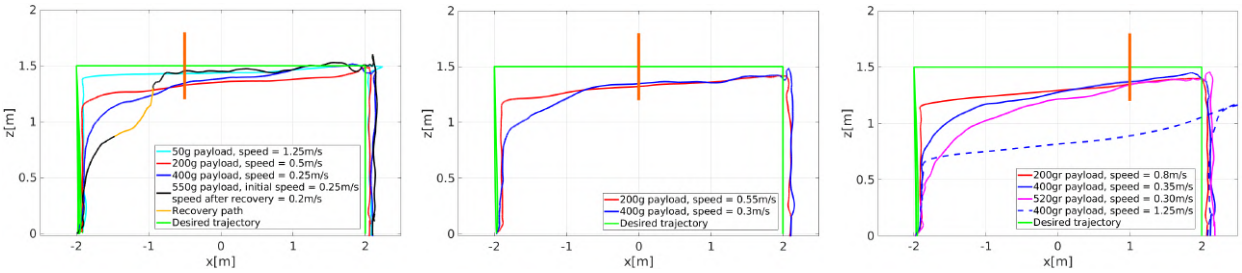


Fig. 11. GP regression results for payload estimation based on the take-off time for the experiment test bed.

quadrotor is able to reach its goal position with $\bar{v}^* = 1.25$m/s which is the maximum permitted average speed value. For payload disturbance 200g (red curve) and 400g (blue curve), the system decides to go with average speed values $\bar{v}^* = 0.5$m/s and $\bar{v}^* = 0.25$m/s respectively successfully passing through the window without collision. With high payload values, decreasing the speed allows the system to follow its trajectory more closely than lower payload values over time. With 550g (black curve), the system cannot find an average speed value which keeps the deviation lower than the desired threshold and decides to go with minimum average speed of 0.25m/s in the beginning and performs a recovery (yellow curve) due to its low height before the obstacle. After recovery, a trajectory with average speed of 0.2m/s is generated and the quadrotor is able to pass the window without a collision. In Fig. 12(b), we show the experiment results for the same window located at $x_w = 0$m. As the deviation in z gets smaller over time, the quadrotor was able to move with higher speeds in this case. For payload disturbance 200g and 400g, the quadrotor picks average speed values $\bar{v}^* = 0.55$m/s and $\bar{v}^* = 0.30$m/s respectively. We did not repeat this experiment for payload disturbance 50g as the quadrotor was able to move with the maximum average speed in the previous case and hence moving the window closer to the goal can only improve the situation. Lastly, we moved the window to the $x_w = 1.0$m position and repeated first the tests with payload disturbances 200g, and 400g. The quadrotor was able to move with $\bar{v}^* = 0.80$m/s and $\bar{v}^* = 0.35$m/s respectively, which are higher than previous (a) and (b) cases, as expected. We also tested with a payload disturbance of 520g (magenta curve), which is higher than the largest payload in training set, and the quadrotor was able to move with 0.30m/s average speed and safely pass the window. With the dashed blue curve in Fig. 12(c), we show the trajectory of the quadrotor moving with the maximum average speed value without using our approach and we demonstrate that the system would have crashed with the wall below the window as expected. These results are also summarized in Table I.

VII. Conclusions and Future Work

In this work, we have presented a fast runtime planning, recovery and learning framework for safe navigation of autonomous systems with unknown payload disturbance. We have leveraged Gaussian Process regression theory to estimate the payload disturbance and the deviation of the



(a) Results for 50g, 200g, 400g, and 550g payloads and window located at $x_w = -0.5$m.

(b) Results for 200g and 400g payloads and window located at $x_w = 0.0$m.

(c) Results for 200g, 400g and 520g payloads and window located at $x_w = 1.0$m.

Fig. 12. Experiment results with three different window positions and varying payloads.

(a) Results with average speed 0.25m/s to pass the window at $x_w = -0.5$m.

(b) Results with average speed 0.30m/s to pass the window at $x_w = 0.0$m.

(c) Results with average speed 0.35m/s to pass the window at $x_w = 1.0$m.

Fig. 13. Overlapped sequence of snapshots for the quadrotor carrying a 400g payload.

TABLE I. EXPERIMENT RESULTS.

	(a) x_w = -0.5m		(b) x_w = 0.0m		(c) x_w = 1.0m	
payload (g)	**payload est. (g)**	**speed (m/s)**	**payload est. (g)**	**speed (m/s)**	**payload est. (g)**	**speed (m/s)**
50	136.74	1.25	-	-	-	-
200	293.89	0.50	276.95	0.55	276.78	0.80
400	443.87	0.25	421.29	0.3	479.85	0.35
520	-	-	-	-	540.62	0.30
550	632.21	0.20	-	-	-	-

system from the desired behavior and to adapt the speed of the planned trajectory. The model used for prediction is also constantly updated at runtime with the data acquired online to improve the future decision making. Currently we are working on including different types of online disturbances like wind and failures in the system and provide assurance during the recovery phase.

ACKNOWLEDGMENTS

This work is based on research sponsored by DARPA under Contract No. FA8750-18-C-0090.

REFERENCES

[1] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec 2017, pp. 1517–1522.

[2] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[3] S. Singh, A. Majumdar, J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5883–5890.

[4] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.

[5] V. Rubies-Royo, D. Fridovich-Keil, S. Herbert, and C. J. Tomlin, "A classification-based approach for approximate reachability," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 7697–7704.

[6] C. J. Ostafew, A. P. Schoellig, T. D. Barfoot, and J. Collier, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking," *Journal of Field Robotics*, vol. 33, no. 1, pp. 133–152, 2016.

[7] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with gaussian processes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 491–496.

[8] C. D. McKinnon and A. P. Schoellig, "Learning multimodal models for robot dynamics online with a mixture of gaussian process experts," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 322–328.

[9] ——, "Learn fast, forget slow: Safe predictive learning control for systems with unknown and changing dynamics performing repetitive tasks," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2180–2187, April 2019.

[10] E. Yel and N. Bezzo, "Fast run-time monitoring, replanning, and recovery for safe autonomous system operations," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019, pp. 1661–1667.

[11] P. J. Cruz and R. Fierro, "Cable-suspended load lifting by a quadrotor uav: hybrid model, trajectory generation, and control," *Autonomous Robots*, vol. 41, no. 8, pp. 1629–1643, 2017.

[12] K. Sreenath, N. Michael, and V. Kumar, "Trajectory generation and control of a quadrotor with a cable-suspended load - a differentially-flat hybrid system," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 4888–4895.

[13] S. Dai, T. Lee, and D. S. Bernstein, "Adaptive control of a quadrotor uav transporting a cable-suspended load with unknown mass," in *53rd IEEE Conference on Decision and Control*, Dec 2014, pp. 6149–6154.

[14] C. E. Rasmussen, "Gaussian processes for machine learning." MIT Press, 2006.

[15] N. Bezzo, K. Mohta, C. Nowzari, I. Lee, V. Kumar, and G. Pappas, "Online planning for energy-efficient and disturbance-aware uav operations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 5027–5033.

[16] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 2520–2525.

[17] C. E. Rasmussen and H. Nickisch, "Gaussian processes for machine learning (gpml) toolbox," *The Journal of Machine Learning Research*, vol. 11, pp. 3011–3015, Dec. 2010.