State Estimation and Belief Space Planning Under Epistemic Uncertainty for Learning-Based Perception Systems

Keiko Nagami and Mac Schwager, *Member, IEEE*

***Abstract*—Learning-based models for robot perception are known to suffer from two distinct sources of error: aleatoric and epistemic. Aleatoric uncertainty arises from inherently noisy training data and is easily quantified from residual errors during training. Conversely, epistemic uncertainty arises from a lack of training data, appearing in out-of-distribution operating regimes, and is difficult to quantify. Most existing state estimation methods handle aleatoric uncertainty through a learned noise model, but ignore epistemic uncertainty. In this work, we propose: (i) an epistemic Kalman filter (EpiKF) to incorporate epistemic uncertainty into state estimation with learned perception models, and (ii) an epistemic belief space planner (EpiBSP) that builds on the EpiKF to plan trajectories to avoid areas of high epistemic and aleatoric uncertainty. Our key insight is to train a generative model that predicts measurements from states, "inverting" the learned perception model that predicts states from measurements. We compose these two models in a sampling scheme to give a well-calibrated online estimate of combined epistemic and aleatoric uncertainty. We demonstrate our method in a vision-based drone racing scenario, and show superior performance to existing methods that treat measurement noise covariance as a learned output of the perception model.**

***Index Terms*—Aerial systems: Perception and autonomy, planning under uncertainty, deep learning for visual perception.**

I. Introduction

WE CONSIDER a common architecture for vision based navigation in which a learning-based perception model ingests raw RGB images and outputs pseudo-measurements of the robot's state vector [1], [2], [3]. Such perception models can be trained with an error covariance output head to quantify their own uncertainty. The pseudo-measurement and covariance estimate are then fed into a filtering module to give a state estimate for downstream planning and control as done in [1]. Our research is motivated by two key shortcomings in this architecture: (i) learning covariance as an output of a perception model fails to quantify epistemic uncertainty, and (ii) learning-based perception models do not allow for projecting uncertainty into the future, posing a significant challenge to planners that reason about future uncertainty, risk, or the belief space.

Quantifying the epistemic uncertainty of learned perception models is challenging, as the behavior of the learned model on out of distribution (OOD) data is generally difficult to predict or analyze. Furthermore, it is unclear how to incorporate a learned perception module into a trajectory planner, since images at future states in the plan are not yet available to pass through the perception model. To overcome these problems, we train a generative model in conjunction with the perception model. The generator takes as input the robot's state vector and outputs a predicted image of what the robot would see at that state. This is the inverse of the perception model, which takes in an image, and outputs a predicted state of where the agent took the image from. We find that the more dissimilar the test data are from the training data, the farther the composition of the two models departs from the identity map. This signal is helpful in quantifying epistemic uncertainty in the learned perception model, without requiring access to the training data, the training residuals, or any other information inherent to the training process that is typically not available at runtime.

We use the composition of the image generator and perception module in a sampling scheme to obtain an online covariance estimate for the pseudo-measurement, and apply this to the update step of a Gaussian belief filter, which we call the Epistemic Kalman Filter (EpiKF). We then incorporate this filter within an optimization-based belief space planning algorithm to plan trajectories that balance efficiency with state estimation uncertainty, which we call the Epistemic Belief Space Planner (EpiBSP). The EpiBSP naturally avoids areas of high sensor noise (aleatoric uncertainty), as well as areas with sparse training data (epistemic uncertainty), without having access to the training data itself. Fig. 1 demonstrates the use of our generative model denoted $g_\phi(\cdot)$ and perception model $p_\theta(\cdot)$ to give a measurement uncertainty signal for both the current and predicted future states. Throughout this work we illustrate concepts of uncertainty using the prototypical "light-dark" problem found in belief space planning and POMDPs [4]. Regions of the state

Manuscript received 10 November 2023; accepted 23 March 2024. Date of publication 10 April 2024; date of current version 22 April 2024. This letter was recommended for publication by Associate Editor T. L. Molloy and Editor H. Kurniawati upon evaluation of the reviewers' comments. This work was supported in part by DARPA under Grant HR001120C0107 and in part by NASA University Leadership initiative under Grant 80NSSC20M0163. *(Corresponding author: Keiko Nagami.)*

The authors are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305 USA (e-mail: knagami@stanford.edu; schwager@stanford.edu).

Code and video can be found at https://msl.stanford.edu/projects/epistemic_planning_filtering.

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3387139, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3387139

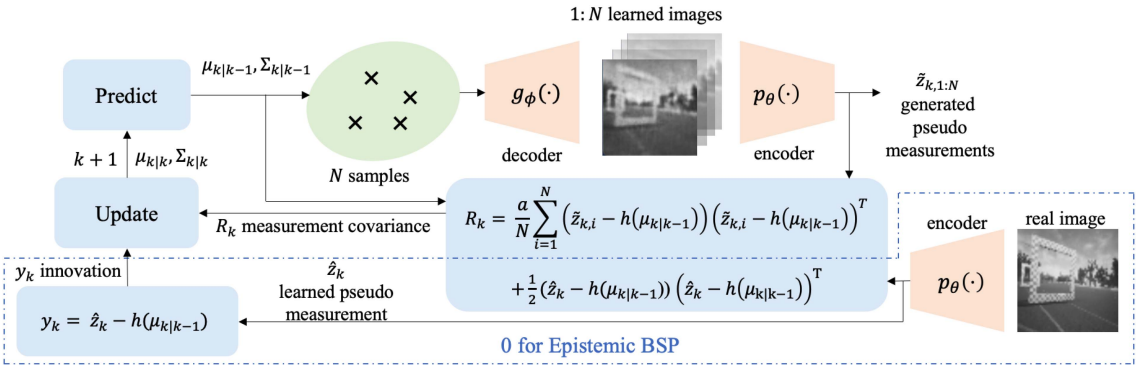


Fig. 1. In both the EpiKF and EpiBSP, N measurements are sampled from the predicted distribution, and passed through the neural network generator $g_\phi(\cdot)$. This produces N predicted images that are forwarded to the neural network perception model $p_\theta(\cdot)$. The resulting output of predicted pseudo-measurements are used to compute the measurement covariance matrix, $\mathbf{R}_k$. In the EpiKF, we use the real image taken from the robot for both the measurement covariance and for the innovation in the belief update. To calculate $\mathbf{R}_k$, we use a weighting factor $a = 1$ in the EpiBSP, since there is no access to the real image, while we use $a = \frac{1}{2}$ in the EpiKF to distribute the weighting evenly across the two terms in the equation.

space that have high measurement uncertainty are shown as dark, while those with low measurement uncertainty appear as light. Agents planning under uncertainty have to choose longer trajectories that move through the light region in order to obtain a sharper state estimate before proceeding to the goal in the dark region. In this letter we delineate two distinct types of "light-dark" that arise in problems with learning-based perception modules. Aleatoric uncertainty corresponds to the classical notion of sensor noise found in "light-dark" problems, while epistemic uncertainty arises from a lack of training data in some operating regimes, and therefore represents a different kind of "light-dark" uncertainty in planning with learned models.

In summary, our main contributions are:

1) We obtain an epistemic uncertainty signal by composing a learned measurement generator with a learned robot perception model.
2) We introduce the EpiKF, a Gaussian belief filter that uses the uncertainty signal to quantify aleatoric and epistemic uncertainty in the robot's state estimate.
3) We embed the EpiKF within a belief space planner (the EpiBSP) to plan robot trajectories to avoid both high measurement noise and out-of-distribution regimes.
4) We demonstrate superior performance to models with learned covariance estimates in a high-fidelity 6DoF drone racing simulation study.

II. Related Work

A. Neural Network OOD Detection and Quantification

A common approach to anticipating both aleatoric and epistemic uncertainty from neural networks is using ensemble methods. These methods use multiple neural network models, where the disagreement across the ensemble of models is used to determine the level of uncertainty. Applications of ensembles are seen in [5], where they are used to estimate uncertainty in dynamics models, as well as in [6], [7], and [2], where ensembles are used for vision-based tasks. However, ensemble methods require training of multiple neural networks. Alternative approaches include the methods in [3], [8], [9], and [10]. In [8] the authors use a matrix sketching technique offline to identify the directions in the weight space that most impact the training predictions, and use this information online by measuring how much perturbations to these directions affect the predictions for new data. In [3] the authors focus on real-time out of distribution detection for new image samples by training a classifier, which is used in a Martingale to detect distribution shifts with high confidence. In [9] and [10], the authors use conformal inference to produce prediction sets to describe the output probability distributions of black-box models like neural networks. While these methods, as well as the ensemble methods, are helpful in determining the reliability of a single measurement taken at the current time, they are unsuitable for our problem, since they cannot anticipate measurement uncertainty at future states as required in a belief space planner. This motivates our approach of training an image generator in conjunction with the perception model.

B. Active Perception

Active perception or perception-aware planning methods generally require a metric to define the quality of the perception system's performance at future states. In using Visual-Inertial Odometry for perception, this can be done by using the positions of feature points. Methods that use this approach focus largely on aleatoric uncertainty, where the perception system's performance is tied to the amount of features in frame, as done in [11], [12], and [13], as well as their velocities or errors, as is done in [14], and [15]. Similarly, [16] uses a method that iteratively improves paths to acquire visibility of unknown regions in the environment. Other methods in active perception have also used neural networks as part of the planning and control pipelines [17], [18]. In [17], the authors use imitation learning and reinforcement learning to train a policy with a reward on perception quality. In [18], the authors use imitation learning to train a network that produces collision free trajectories. However, these methods of active-perception and perception aware planning, do not handle epistemic uncertainty, which is crucial in learned perception systems.

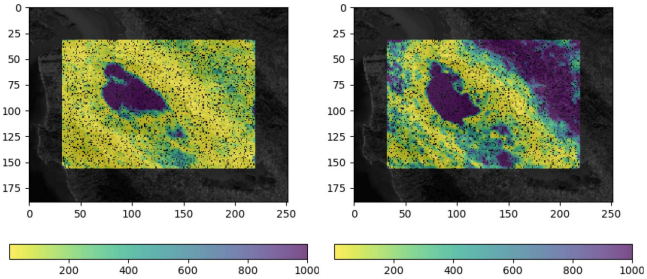
C. Belief Space Planning

Belief space planning considers the evolution of the whole belief (represented as a Gaussian, a histogram, or a set of samples) along future trajectories, and optimizes plans to balance trajectory efficiency with information richness. A common approach is to formulate the problem as a Partially Observable Markov Decision Process (POMDP) [19] and to precompute a value function across the belief space that is used to make decisions for planning. However, with continuous belief states, actions, and observations, the problem is typically intractable. An alternative solution is to use sampling based approaches, as done in [13], [20], [21], [22], and [23], where potential trajectories are sampled, and the total cost of the trajectories are calculated and compared. Another method is to use local trajectory optimization approaches, as used in [24], [25], and [26] where gradient based methods are used to optimize for trajectories that are able to reach a goal while also maintaining high confidence in the state estimates. The authors of [4] do this online with a Linear Quadratic Regulator using belief states and belief dynamics. However, since the measurements to be taken at future states are not yet known, [4] assumes known measurement uncertainties and maximum likelihood observations. Still other methods to perform belief space planning exist. In [27], the authors model continuous time controls and discrete time observations as a hybrid system to formulate an optimal control problem that is solved using Sequential Action Control (SAC) [28]. In [29], the authors generate a set of open loop trajectories and compute funnels for these trajectories using sum-of-squares programming. The funnels are then used at run-time by sequentially composing them. None of these methods consider the epistemic uncertainty that arises with learned perception models.

Our work differs significantly from all of these methods in the way we quantify measurement noise at future states in the plan. We use two neural networks to define our measurement uncertainty. The first is a generative neural network, similar to those in [30] and [31]. This network is trained to produce an image given a state vector. We then train a separate neural network perception model that maps from image to state using the same dataset or a dataset from the same distribution. When we compose these models, we find that the error between the input state to the generator and the output state from the perception model gives a signal that is well-correlated with epistemic uncertainty. A similar pipeline is used in [32], however the authors in that paper apply a generator for neural network verification of a controller, not for estimating epistemic uncertainty. The use of this anticipated measurement uncertainty enables us to formulate an epistemic belief space planner (EpiBSP) built on an epistemic Kalman filter (EpiKF), to account for degraded state estimation performance in the regions of high epistemic and aleatoric uncertainty.

III. Model Formulation

We consider a problem where a robot collects image measurements taken by an onboard camera as it navigates through its environment. A pretrained neural network perception model is used to map the input images to a learned pseudo-measurement



(a) Ground Truth Error (b) Predicted Error

Fig. 2. Heat maps comparing the ground truth perception error and generated image perception error. The predicted error tends to over-estimate the true perception error, giving conservative filter and planner performance.

$\hat{\mathbf{z}}$. This vector is used in a Gaussian belief filter to produce an estimate of the mean and covariance of the state, which is passed to a belief space planner that is applied in a model-predictive control (MPC) loop. We assume that the system has known dynamics with Gaussian distributed process noise. We define $\mathbf{s}$ as the ground truth state vector, $\mathbf{z}$ as the ground truth pseudo-measurement, $f(\cdot)$ as the known dynamics function, and $h(\cdot)$ as the known measurement function that maps $\mathbf{s}$ to $\mathbf{z}$. Since the underlying ground truth states are unknown, the belief states are modeled as Gaussian distributions, parameterized by a mean, $\boldsymbol{\mu}$, and covariance $\boldsymbol{\Sigma}$. Our approach is visualized in Fig. 1.

A. Perception System

In our approach, we require two neural network models, trained in a supervised fashion on the same dataset (or data of the same distribution) of image and pseudo-measurement pairs. The first maps from ground truth image inputs $\mathcal{I}$ to learned pseudo-measurements $\hat{\mathbf{z}}$, and the second maps from ground truth pseudo-measurements $\mathbf{z}$ to learned images $\hat{\mathcal{I}}$ as follows

$$\hat{\mathbf{z}} = p_\theta(\mathcal{I}) \tag{1}$$

$$\hat{\mathcal{I}} = g_\phi(\mathbf{z}). \tag{2}$$

The neural network model parameters for the perception and generator are defined by θ and ϕ, respectively. We compose the two networks together in the estimation and planning stages as

$$p_\theta(g_\phi(\mathbf{z})) = \tilde{\mathbf{z}}, \tag{3}$$

where $\tilde{\mathbf{z}}$ is the generated pseudo-measurement. We find the pseudo-measurement reconstruction error $(\tilde{\mathbf{z}} - \mathbf{z})$ is highly predictive of combined epistemic and aleatoric uncertainty in both of these learned models. This captures epistemic uncertainty because the two learned models become less accurate at predicting their outputs as their respective inputs stray farther from the training data. A toy example of this is shown in Fig. 2. We train both networks in a 2D map environment created from a satellite image from [33], where the pseudo-measurement $\mathbf{z}$ is a pixel coordinate in the 2D map image, and the training images are pixel patches of the map centered at the corresponding pseudo-measurement. We compute the error between the pixel coordinates of the ground truth labels, and the associated output

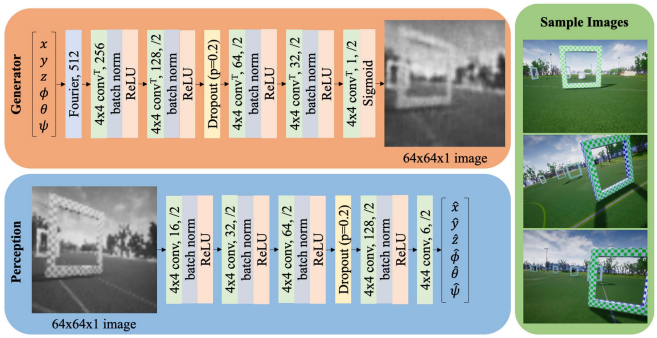


Fig. 3. Generator and Perception neural networks. Kernel size, number of channels, and stride length are shown for each convolutional layer and transposed convolutional layer. Sample RGB images from the environment are included.

of the perception model when the ground truth image is used as input to the network. The resulting errors are shown in Fig. 2(a). We then use the ground truth pixel coordinates as input to the generator, produce a generated image, and pass through the perception model. The resulting errors are shown in Fig. 2(b).

These errors compound in the composition of the models, leading to a conservative estimate of the perception error. When $\mathbf{z}$ is close to the training data, any reconstruction error is due to inherent randomness in the training data (aleatoric uncertainty), which makes an exact reconstruction impossible. Far from the training data, the predicted error incorporates both epistemic and aleatoric sources of uncertainty. Crucially, this combined uncertainty signal is obtained *without having access to an image* $\mathcal{I}$. All existing OOD detection/epistemic uncertainty quantification methods (neural network ensembles, matrix sketching, and martingale techniques, etc.) to our knowledge require access to an image $\mathcal{I}$, rendering them unsuitable for predictive planning.

Fig. 3 shows the architectures of our two models. Specifically, for the generator model we use a Fourier Encoding technique similar to that of [34], which we find to be crucial to obtaining suitable generated images. For all networks, a Huber loss function is used to train with the labeled data.

IV. Epistemic Kalman Filter

As detailed in the previous sections, the perception system takes an image input and outputs a learned pseudo-measurement of the state, $\hat{\mathbf{z}}$. With this vector, we filter the state with an Extended Kalman Filter (EKF), using the standard predict and update steps,

$$\text{Predict:} \quad \boldsymbol{\mu}_{k+1|k} = f(\boldsymbol{\mu}_{k|k}, \mathbf{u}_k) \tag{4}$$

$$\boldsymbol{\Sigma}_{k+1|k} = \mathbf{F}_k \boldsymbol{\Sigma}_{k|k} \mathbf{F}_k^T + \mathbf{Q} \tag{5}$$

$$\text{Update:} \quad \mathbf{y}_{k+1} = \hat{\mathbf{z}}_{k+1} - h(\boldsymbol{\mu}_{k+1|k}) \tag{6}$$

$$\mathbf{S}_{k+1} = \mathbf{H}_{k+1} \boldsymbol{\Sigma}_{k+1|k} \mathbf{H}_{k+1}^T + \mathbf{R}_{k+1} \tag{7}$$

$$\mathbf{K}_{k+1} = \boldsymbol{\Sigma}_{k+1|k} \mathbf{H}_{k+1}^T \mathbf{S}_{k+1}^{-1} \tag{8}$$

$$\boldsymbol{\mu}_{k+1|k+1} = \boldsymbol{\mu}_{k+1|k} + \mathbf{K}_{k+1} \mathbf{y}_{k+1} \tag{9}$$

$$\boldsymbol{\Sigma}_{k+1|k+1} = (\mathbf{I} - \mathbf{K}_{k+1} \mathbf{H}_{k+1}) \boldsymbol{\Sigma}_{k+1|k}, \tag{10}$$

where subscript k indicates the time index, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean and covariance of the estimated state, $\mathbf{u}$ is the control input, $\mathbf{H}$ is the Jacobian of the measurement function, $\mathbf{F}$ is the Jacobian of the dynamics equation, $\mathbf{I}$ is an identity matrix, $\mathbf{Q}$ is the process noise covariance matrix, and $\mathbf{R}$ is the measurement noise covariance matrix.

In our problem setup, we assume that the dynamics equation, process noise, and measurement equation are known, and that we receive the learned pseudo-measurement from a neural network perception model. This necessitates a definition of the measurement noise covariance $\mathbf{R}$ in (7). We propose to obtain $\mathbf{R}$ by first sampling pseudo-measurements from the prediction distribution, passing these sampled vectors through the generator network to produce learned images, and then passing these through the perception network to obtain sampled generated pseudo-measurement vectors,

$$\mathbf{z}_{k,i} \sim \mathcal{N}(h(\boldsymbol{\mu}_{k|k-1}), \mathbf{H}_k \boldsymbol{\Sigma}_{k|k-1} \mathbf{H}_k^T), \quad \forall i = 1:N \tag{11}$$

$$\tilde{\mathbf{z}}_{k,i} = p_\theta(g_\phi(\mathbf{z}_{k,i})). \tag{12}$$

This sampling procedure is done for N samples, where i is the index of the sample. With the sampled generated pseudo-measurements $\tilde{\mathbf{z}}_{k,i}$, we compute their empirical covariance matrix, and combine it with the learned pseudo-measurement of the true image from the predicted mean,

$$\mathbf{R}_k = \frac{1}{2N} \sum_{i=1}^{N} (\tilde{\mathbf{z}}_{k,i} - h(\boldsymbol{\mu}_{k|k-1}))(\tilde{\mathbf{z}}_{k,i} - h(\boldsymbol{\mu}_{k|k-1}))^T + \frac{1}{2} (\hat{\mathbf{z}}_k - h(\boldsymbol{\mu}_{k|k-1}))(\hat{\mathbf{z}}_k - h(\boldsymbol{\mu}_{k|k-1}))^T. \tag{13}$$

This matrix $\mathbf{R}_k$ is then used as the measurement noise covariance matrix in the EKF equations. This overall pipeline is shown in Fig. 1. With this formulation we are able to get a measurement uncertainty matrix that reflects the predicted performance of the neural network perception model at the estimated state in the state space, incorporating both aleatoric and epistemic sources of uncertainty.

V. Epistemic Belief Space Planning

In this section we formulate a belief space planner as a trajectory optimization problem, informed by the measurement uncertainty at future states from the EpiKF described above. The trajectory optimization problem we seek to solve takes the following form,

$$\underset{\mathbf{u}_{k=1:K-1}}{\text{minimize}} \quad c_1 \text{tr}(\boldsymbol{\Sigma}_K) + c_2 \bar{\boldsymbol{\mu}}_K^T \mathbf{C}_s \bar{\boldsymbol{\mu}}_K + c_3 \mathbf{U}^T \mathbf{C}_u \mathbf{U} \tag{14}$$

$$\text{s.t. } \boldsymbol{\mu}_{k+1|k} = f(\boldsymbol{\mu}_{k|k}, \mathbf{u}_k), \quad \forall k = 1:K-1 \tag{15}$$

$$\boldsymbol{\Sigma}_{k+1|k} = \mathbf{F}_k \boldsymbol{\Sigma}_{k|k} \mathbf{F}_k^T + \mathbf{Q} \tag{16}$$

$$\mathbf{y}_{k+1} = \tilde{\mathbf{z}}_{k+1} - h(\boldsymbol{\mu}_{k+1|k}) \tag{17}$$

$$\mathbf{S}_{k+1} = \mathbf{H}_{k+1} \boldsymbol{\Sigma}_{k+1|k} \mathbf{H}_{k+1}^T + \mathbf{R}_{k+1} \tag{18}$$

$$\mathbf{K}_{k+1} = \boldsymbol{\Sigma}_{k+1|k} \mathbf{H}_{k+1}^T \mathbf{S}_{k+1}^{-1} \tag{19}$$

$$\boldsymbol{\mu}_{k+1|k+1} = \boldsymbol{\mu}_{k+1|k} + \mathbf{K}_{k+1} \mathbf{y}_{k+1} \tag{20}$$

$$\boldsymbol{\Sigma}_{k+1|k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1})\boldsymbol{\Sigma}_{k+1|k}, \tag{21}$$

where K is the terminal time index, $\bar{\boldsymbol{\mu}}_K$ is the difference between the mean of the estimate $\boldsymbol{\mu}_K$ and the goal state $\mathbf{s}_g$, $\mathbf{C}_s$ is a cost matrix on the goal state, $\mathbf{U}$ is the sequence of control inputs, $\mathbf{U} = \left[\mathbf{u_1^T}, \mathbf{u_2^T}, \ldots, \mathbf{u_{K-1}^T}\right]^T$, $\mathbf{C}_u$ is a cost matrix on the control action, c_1, c_2, and c_3 are constants, $\tilde{\mathbf{z}}$ is the generated pseudo-measurement, and $\mathbf{R}$ is the predicted measurement uncertainty. We use the trace of the covariance of the final belief state to quantify the uncertainty cost as a simplification of the uncertainty cost used in [4].

For the predicted measurement uncertainty, we follow a similar procedure to the measurement uncertainty in the EpiKF described in Section IV, where we sample from the predicted distribution, and pass predicted measurement vectors through the generator and perception networks to reproduce the measurement vector (11) and (12). However, the key challenge in the belief space planner is the lack of access to the collected image measurements from the robot. As such, we use only the first term in (13) to define the measurement uncertainty in the belief space planner:

$$\mathbf{R}_k = \frac{1}{N}\sum_{i=1}^{N}(\tilde{\mathbf{z}}_{k,i} - h(\boldsymbol{\mu}_{k|k-1}))(\tilde{\mathbf{z}}_{k,i} - h(\boldsymbol{\mu}_{k|k-1}))^T. \tag{22}$$

Since there is no real image collected for the planner, we must also define $\tilde{\mathbf{z}}$ in (17). One option is to use a Maximum Likelihood Observation (MLO), where $\tilde{\mathbf{z}}_k = h(\boldsymbol{\mu}_{k|k-1})$. This is the assumption that is used for the planning procedure through the rest of the paper. However, an alternative approach can be used to generate a pseudo-measurement using the sampled pseudo-measurements from the predicted mean.

With this formulation, our belief state dynamics are a function of the neural network generator and perception model. To initialize our trajectories, we use IPOPT to generate a trajectory toward the goal state. From here, we use Pytorch [35] to define the sequence of control inputs as optimization variables, and use a gradient descent approach to minimize the cost function shown in (14)–(21). We use Pytorch so that we can compute the cost with respect to the control inputs through the neural network model queries.

VI. Simulation Experiments

We demonstrate this approach on an environment where a quadrotor flies through gates. We use Airsim Drone Racing Lab [36] as the simulation environment where all computation is run on an Ubuntu 20.04 OS desktop with an AMD Ryzen 9 5900X 12-Core Processor, 32 GB of RAM, and an NVIDIA GeForce RTX 3090 GPU. Our state dimension includes position $\mathbf{p}$, orientation $\boldsymbol{\theta}$, and velocity $\mathbf{v}$, relative to the gate center, the pseudo-measurement includes the position and orientation, and the control is a scalar thrust u_T and body rate input $\mathbf{u}_\omega$:

$$\mathbf{s} = \begin{bmatrix} x & y & z & \phi & \theta & \psi & v_x & v_y & v_z \end{bmatrix}^T = \begin{bmatrix} \mathbf{p}^T & \boldsymbol{\theta}^T & \mathbf{v}^T \end{bmatrix}^T \tag{23}$$

$$\mathbf{z} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T = \begin{bmatrix} \mathbf{p}^T & \boldsymbol{\theta}^T \end{bmatrix}^T \tag{24}$$

$$\mathbf{u} = \begin{bmatrix} u_t & u_p & u_q & u_r \end{bmatrix}^T = \begin{bmatrix} u_t & \mathbf{u}_\omega^T \end{bmatrix}^T, \tag{25}$$

where the control inputs are limited by minimum and maximum values. All data was collected by the authors in the simulation environment with first-person view image and pose pairs. The state of the quadrotor evolves with dynamics:

$$\dot{\mathbf{p}} = \mathbf{v} \tag{26}$$

$$\dot{\mathbf{v}} = \frac{u_T}{m}\mathbf{z}_b - g\mathbf{z}_w \tag{27}$$

$$\dot{\boldsymbol{\theta}} = \mathbf{T}\mathbf{u}_\omega, \tag{28}$$

where m is the mass of the quadrotor, $\mathbf{z}_b$ is the z unit vector in the body frame, g is the acceleration due to gravity, $\mathbf{z}_w$ is the z unit vector in the world frame, and $\mathbf{u}_\omega$ is a vector of the body rate control inputs. $\mathbf{T}$ is a matrix to map the body rates to rates of the euler angles,

$$\mathbf{T} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}. \tag{29}$$

We use this simplified 9 dimensional quadrotor model to maintain a state vector of position, orientation, and velocity as done in [15] and [37], instead of the full 12 dimensional state vector often found in quadrotor planning and control [38]. This is convenient in interfacing with a fast low-level feedback controller that regulates the body rotation rates to follow those commanded by this higher-level model. In belief space planning approaches, the dimensionality of the state-space grows as we must also account for the uncertainty of the state to represent the belief. Using this lower dimensional representation allows us to reduce the size of the state space for the computation in the trajectory optimization.

A. Perception Models

To demonstrate the performance of our method in the presence of epistemic and aleatoric uncertainty, we train a generator and perception network pair for two different sets of data. In the first, we hold out data in the negative x dimension of the gate, creating a region of epistemic uncertainty. In the second, we corrupt the quality of the image data by adding random Gaussian noise to the images in the negative x dimension, creating a region of aleatoric uncertainty. For both of these datasets, this creates a "dark" region of the state space where the measurement uncertainty should be higher than the regions where the models are trained on higher quality data. With these trained networks, we expect our planner to be able to compensate for both types of uncertainty by navigating through regions of higher certainty on the right side of the gate entrance. For ease of training, we compress and grayscale the images to 64 x 64 pixel images.

B. Learned Measurement Uncertainty Baseline

We compare our method to an alternative approach of learning the measurement uncertainty of images by training a neural network to output the diagonal of a covariance matrix associated

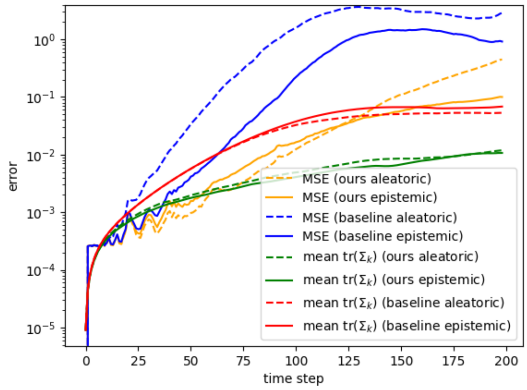


Fig. 4. EpiKF and baseline covariance statistics. Each filter is run on open loop trajectories 1000 times initialized in high uncertainty regions in aleatoric and epistemic uncertainty. We plot the average mean-squared error and mean covariance of the rollouts for each time step in the trajectory.

with the image input. The baseline we compare to is inspired by [1], where the uncertainty associated with a collected measurement is learned by the same neural network that predicts the pose for the image,

$$(\hat{\mathbf{z}}, \hat{\mathbf{R}}) = p_\psi(\mathcal{I}) \tag{30}$$

$$(\tilde{\mathbf{z}}, \tilde{\mathbf{R}}) = p_\psi(g_\phi(\mathbf{z})). \tag{31}$$

We use this baseline network to compare against our approach in both the EpiKF and the EpiBSP.

C. Epistemic Kalman Filter

To isolate the performance of the EpiKF, we roll forward the quadrotor dynamics with process noise and collect images along a trajectory where the quadrotor flies toward a gate located at the origin. We then run our EpiKF on this trajectory from an uncertain mean state estimate, using the collected images. We run the filter 1000 times from initial mean state estimates drawn randomly from a Gaussian distribution, and apply this in the aleatoric and epistemic uncertainty cases, initialized in "dark" and "light" regions.

We apply the same 1000 rollouts using a baseline filtering approach, where we use the baseline network, $p_\psi(\cdot)$ described in Section VI-B. The measurement uncertainty is an output of this model, $\hat{\mathbf{R}}$ in (30), using the real image collected as input. Resulting sample belief trajectories are shown in Fig. 5. When the trajectories are initialized in the "light" regions of the state space, the mean of the estimates for both the baseline and our method are close to the ground truth trajectory. The main differences are seen when the trajectories are initialized in the "dark" region of the state space. Our method is able to maintain an estimate with a mean that is closer to the ground truth trajectory, while the baseline deviates significantly.

We additionally demonstrate how our method is able to better approximate the uncertainty in the estimate. To do this, we take the average across the 1000 filtering rollouts of the trace of the covariance of the predicted belief states along the trajectory. We then compare this to the mean squared error of the estimated mean from the ground truth trajectory for the 1000 rollouts. These comparisons are shown in Fig. 4, where our method is able to more accurately represent the actual deviation from the ground truth. Averaged over 100 steps, our method runs with 0.00382 ± 0.0007 seconds.

TABLE I
EPISTEMIC BELIEF SPACE PLANNER VS. BASELINE

Training Data Uncertainty	Aleatoric	Epistemic
Our Method	0.2180 ± 0.1195	0.3350 ± 0.1358
Baseline	0.2518 ± 0.1476	0.4973 ± 0.2281

D. Epistemic Belief Space Planner

In this section, we demonstrate the use of the EpiBSP. We first initialize trajectories in aleatoric and epistemic uncertainty cases, where the starting pose is in the higher uncertainty region, and the goal pose is 2 meters in front of the gate. To initialize these trajectories we use the Interior Point optimization (IPOPT) solver in the Casadi library [39] including the control limits and dynamics in (26)–(28). We construct a cost function described in (14)–(21) as a function of the control sequence $\mathbf{u}_{k=1:K-1}$, and use gradient descent using automatic differentiation tools in Pytorch to iteratively optimize the trajectory [35]. The initial and optimized trajectories are shown in Fig. 6.

In both cases, our method optimizes the trajectory to reach the "light" region of state space before reaching the goal. We additionally compare our method to a baseline, where we use the neural network described in Section VI-B. Because real images are unavailable in the planning stage, we use the image generator to pass through sampled pseudo-measurements to generated learned images that are passed through the neural network baseline. This results in N generated pseudo-measurements for each point in the trajectory, $\tilde{\mathbf{z}}$, and their associated predicted covariances, $\tilde{\mathbf{R}}$ from (31). The average of these N covariance outputs are used as the measurement uncertainty at that belief state.

In Table I we show that the final optimized trajectory in our method is able to produce a lower uncertainty cost. We calculate this uncertainty cost by running the EpiKF on each trajectory 100 times with initial mean estimates randomly sampled from a Gaussian distribution, and compute the mean squared error between the final trajectory state and the final state estimate at time step K. This metric is meant to represent the first cost term in (14), where the trace of the covariance of the state estimate at time step K is penalized. Run for 2000 iterations, our method takes 55.66 minutes to compute in the epistemic case. While these times for the belief space planner are long, these trajectories would be computed offboard before flight, while onboard compute would be reserved for trajectory tracking and the EpiKF.

E. Full Track Simulation

To showcase both the EpiKF and EpiBSP working together, we apply both to a case where a drone is flying through a race-track with multiple gates. We use the generator and perception

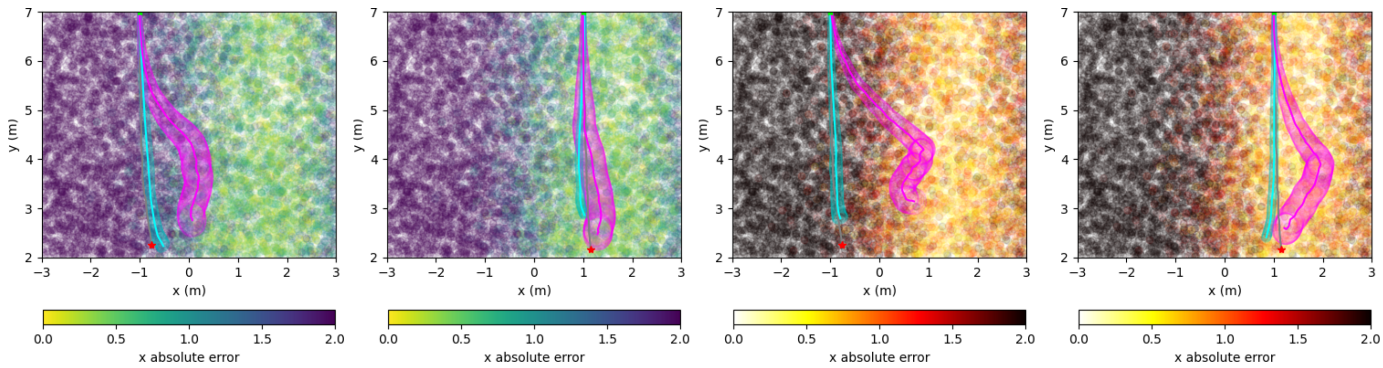


Fig. 5. Top-down view of sample rollout trajectories applying the EpiKF (blue) and baseline approach (pink) in which the model is trained to output an approximate measurement covariance. The left two figures show the performance of both methods in the presence of epistemic uncertainty while the right two show aleatoric uncertainty. In all cases, the heat map for x absolute error is computed by querying poses in the environment, passing them through the generator and perception models, and plotting the magnitude of the x-component of the absolute error $|\mathbf{z} - \tilde{\mathbf{z}}|$. The first and third demonstrate the performance of the filter in "dark" regions, and the second and fourth show performance in "light" regions. In all cases, the EpiKF has lower estimation errors, and better-calibrated estimate covariance.

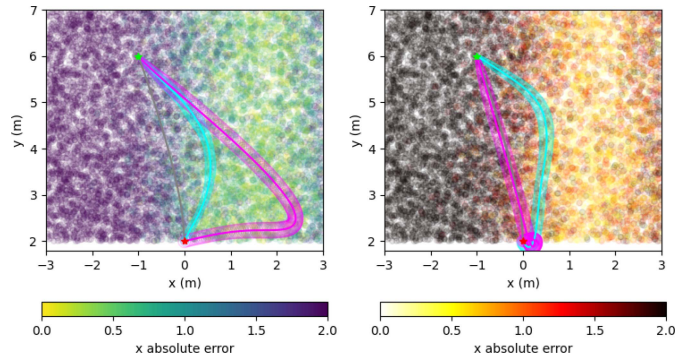


Fig. 6. Top-down view of trajectories after optimizing for 2000 iterations from the initial trajectory using the EpiBSP (blue) and the baseline (pink). The plot on the left showcases both methods in the epistemic uncertainty case, and on the right the aleatoric uncertainty case. In both cases the EpiBSP bends the trajectory into the low uncertainty region, while still finding an efficient trajectory to the goal. The baseline planner fails to bend the trajectory in to the light region under aleatoric uncertainty (right), and bends too far under epistemic uncertainty (left). The heat maps of x absolute error are computed in the same way as those in Fig. 5.

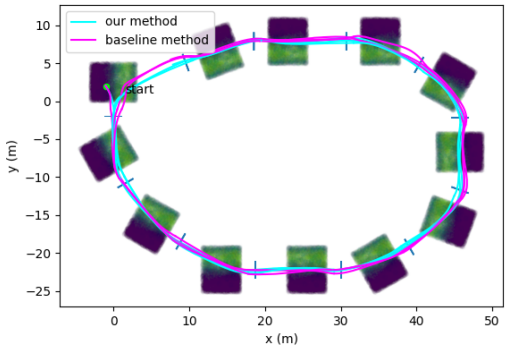


Fig. 7. Trajectories of our method and the baseline comparison method through the Airsim racetrack for three laps.

TABLE II
COMPARISON RESULTS FOR FULL RACETRACK

	Gate Collisions	Error to Ground Truth	Total Time
Our Method	0	0.3074 ± 0.2315	2 min 35.94 sec
Baseline	1	0.4047 ± 0.4453	2 min 38.68 sec

neural networks trained with epistemic uncertainty, where data from one side of the gate is withheld. We apply our methods when the drone is within 7 meters and 2 meters in the y dimension of the local frame of the next oncoming gate. Outside of this band, we provide noisy GPS and orientation measurements. This is to prevent poor measurements that arise from seeing the next gate, and losing vision of the current gate. For planning, we apply the EpiBSP by solving for a trajectory at each gate transition. Once the trajectory is optimized, we track this trajectory in a model-predictive control (MPC) loop. The racetrack consists of 12 gates, and we loop through the track three times. The resulting trajectory is shown in Fig. 7.

We additionally compare our method to a baseline similar to that of [1], where we use the neural network baseline described in Section VI-B for estimation. For planning, we use the initialized trajectory from the IPOPT solver. The trajectory for three laps is also shown in Fig. 7. To quantify the performance of our method, we compare the number of gate collisions, error of the state estimate mean to the ground truth, and the total number of time step taken to reach the goal. These metrics are shown for our method and the baseline in Table II. We find that our method is able to complete three laps with fewer collisions, and maintain a lower error from the ground truth than the baseline approach.

VII. CONCLUSION

In this letter we demonstrate a novel approach to quantifying neural network uncertainty, and use this metric to anticipate regions of high uncertainty with belief space planning and Kalman filtering. We found that our method is able to avoid regions of both aleatoric and epistemic uncertainty to produce trajectories where the perception model is more likely to succeed. In future work, real-time belief space planning trajectories can be prioritized by applying a sampling based planning approach to reduce computation time that is induced by the automatic differentiation

through the perception networks required for the presented local optimization approach, which would enable real-world experiments. Additionally, improved generator models could be applied, so that predicted images closer represent the training distribution. This could be done by using tools to generate neural radiance fields (NeRF) [40].

ACKNOWLEDGMENT

This article solely reflects the opinions and conclusions of its authors and not any NASA entity.

REFERENCES

[1] E. Kaufmann et al., "Beauty and the beast: Optimal methods meet learning for drone racing," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 690–696.
[2] A. Loquercio, M. Segu, and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 3153–3160, Apr. 2020.
[3] R. Luo et al., "Online distribution shift detection via recency prediction," 2022, *arXiv:2211.09916*.
[4] R. Platt Jr., R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," in *Robotics: Science and Systems VI*. Cambridge, MA, USA: MIT Press, 2011.
[5] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 4754–4765.
[6] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 6402–6413.
[7] F. K. Gustafsson, M. Danelljan, and T. B. Schon, "Evaluating scalable Bayesian deep learning methods for robust computer vision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 1289–1298.
[8] A. Sharma, N. Azizan, and M. Pavone, "Sketching curvature for efficient out-of-distribution detection for deep neural networks," in *Proc. Uncertainty Artifi. Intell.*, PMLR, 2021, pp. 1958–1967.
[9] A. Angelopoulos, S. Bates, J. Malik, and M. I. Jordan, "Uncertainty sets for image classifiers using conformal prediction," in *Proc. Int. Conf. Learning Representations*, 2021.
[10] Y. Romano, M. Sesia, and E. Candes, "Classification with valid and adaptive coverage," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 3581–3591.
[11] L. Bartolomei, L. Teixeira, and M. Chli, "Perception-aware path planning for UAVs using semantic segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5808–5815.
[12] M. Jacquet, G. Corsini, D. Bicego, and A. Franchi, "Perception-constrained and motor-level nonlinear MPC for both underactuated and tilted-propeller UAVs," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 4301–4306.
[13] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart, "Motion-and uncertainty-aware path planning for micro aerial vehicles," *J. Field Robot.*, vol. 31, no. 4, pp. 676–698, 2014.
[14] X. Wu, S. Chen, K. Sreenath, and M. W. Mueller, "Perception-aware receding horizon trajectory planning for multicopters with visual-inertial odometry," *IEEE Access*, vol. 10, pp. 87911–87922, 2022.
[15] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-aware model predictive control for quadrotors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–8.
[16] B. Zhou, J. Pan, F. Gao, and S. Shen, "RAPTOR: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1992–2009, Dec. 2021.
[17] Y. Song, K. Shi, R. Penicka, and D. Scaramuzza, "Learning perception-aware agile flight in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 1989–1995.
[18] J. Tordesillas and J. P. How, "Deep-PANTHER: Learning-based perception-aware trajectory planner in dynamic environments," *IEEE Robot. Automat. Lett.*, vol. 8, no. 3, pp. 1399–1406, Mar. 2023.
[19] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1, pp. 99–134, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000437029800023X
[20] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *Int. J. Robot. Res.*, vol. 28, no. 11-12, pp. 1448–1465, 2009.
[21] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 723–730.
[22] J. van Berg Den, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 895–913, 2011.
[23] Z. Sunberg and M. Kochenderfer, "Online algorithms for POMDPs with continuous state, action, and observation spaces," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2018, vol. 28, pp. 259–263.
[24] J. van Berg Den, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *Int. J. Robot. Res.*, vol. 31, no. 11, pp. 1263–1278, 2012.
[25] M. Rafieisakhaei, S. Chakravorty, and P. Kumar, "T-LQG: Closed-loop belief space planning via trajectory-optimized LQG," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 649–656.
[26] K. Sun and V. Kumar, "Belief space planning for mobile robots with range sensors using iLQG," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1902–1909, Apr. 2021.
[27] H. Nishimura and M. Schwager, "SACBP: Belief space planning for continuous-time dynamical systems via stochastic sequential action control," *Int. J. Robot. Res.*, vol. 40, no. 10-11, pp. 1167–1195, 2021.
[28] A. R. Ansari and T. D. Murphey, "Sequential action control: Closed-form optimal control for nonlinear and nonsmooth systems," *IEEE Trans. Robot.*, vol. 32, no. 5, pp. 1196–1214, Oct. 2016.
[29] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *Int. J. Robot. Res.*, vol. 36, no. 8, pp. 947–982, 2017.
[30] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learn. Representations*, 2014.
[31] I. Goodfellow et al., "Generative adversarial NETS," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, vol. 27, pp. 2672–2680.
[32] S. M. Katz, A. L. Corso, C. A. Strong, and M. J. Kochenderfer, "Verification of image-based neural network controllers using generative models," *J. Aerosp. Inf. Syst.*, vol. 19, no. 9, pp. 574–584, 2022.
[33] Copernicus Sentinel-2 ESA, Accessed on: Mar. 7, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Bay_Area_by_Sentinel-2,_2019-03-11_(big_version).jpg
[34] M. Tancik et al., "Fourier features let networks learn high frequency functions in low dimensional domains," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 7537–7547.
[35] A. Paszke et al., "PyTrch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 8026–8037.
[36] R. Madaan et al., "Airsim drone racing lab," in *Proc. NeurIPS Competition Demonstration Track*, 2020, pp. 177–191.
[37] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1294–1310, Dec. 2015.
[38] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 620–626, Apr. 2018.
[39] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi–A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, pp. 1–36, 2019.
[40] B. Mildenhall, P.P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NERF: Representing scenes as neural radiance fields for view synthesis," *Commun. ACM*, vol. 65, no. 1, pp. 99–106, 2021.