Uncertainty-Aware Reinforcement Learning for Collision Avoidance

Gregory Kahn*, Adam Villaflor*, Vitchyr Pong*, Pieter Abbeel*†, Sergey Levine*
*Berkeley AI Research (BAIR), University of California, Berkeley
†OpenAI

***Abstract*—Reinforcement learning can enable complex, adaptive behavior to be learned automatically for autonomous robotic platforms. However, practical deployment of reinforcement learning methods must contend with the fact that the training process itself can be unsafe for the robot. In this paper, we consider the specific case of a mobile robot learning to navigate an a priori unknown environment while avoiding collisions. In order to learn collision avoidance, the robot must experience collisions at training time. However, high-speed collisions, even at training time, could damage the robot. A successful learning method must therefore proceed cautiously, experiencing only low-speed collisions until it gains confidence. To this end, we present an uncertainty-aware model-based learning algorithm that estimates the probability of collision together with a statistical estimate of uncertainty. By formulating an uncertainty-dependent cost function, we show that the algorithm naturally chooses to proceed cautiously in unfamiliar environments, and increases the velocity of the robot in settings where it has high confidence. Our predictive model is based on bootstrapped neural networks using dropout, allowing it to process raw sensory inputs from high-bandwidth sensors such as cameras. Our experimental evaluation demonstrates that our method effectively minimizes dangerous collisions at training time in an obstacle avoidance task for a simulated and real-world quadrotor, and a real-world RC car. Videos of the experiments can be found at https://sites.google.com/site/probcoll.**

I. Introduction

Policy search via reinforcement learning holds the promise of automating a wide range of decision making and control tasks in safety-critical domains, ranging from self-driving vehicles to drones. However, many reinforcement learning algorithms experience failures at training time, which can be catastrophic in safety-critical domains. Other reinforcement learning algorithms ensure safety by assuming complete state and environment knowledge at training time; however, these assumptions often severely restrict the feasibility of real-world robot deployment. Developing reinforcement learning algorithms that reason about perception and control in unknown environments, understand uncertainty, and explore safely is crucial to deploying reinforcement learning algorithms on safety-critical systems.

One of the central challenges in reinforcement learning is that a robot can only learn the outcome of an action by executing the action itself. Consider a robot learning to navigate an unknown environment while avoiding collisions. This scenario seemingly presents a quandary: the robot needs to learn how to avoid collisions in order to achieve the desired task, but to learn how to avoid collisions, the robot must experience (possibly catastrophic) collisions during training. The robot can overcome this quandary by first experiencing gentle collisions in order to learn about the environment; once the robot is confident about the environment, the robot can avoid catastrophic failures in the future. Central to this approach is that the robot must be able to reason about its own uncertainty because these catastrophic failures are likely to occur in novel scenarios.

Fig. 1: **Uncertainty-aware collision prediction model for collision avoidance**: A quadrotor and an RC car are tasked with navigating in an unknown environment. How should the robots navigate while avoiding collisions? We propose a model-based reinforcement learning approach in which the robot learns a collision prediction model by experiencing collisions at low speed, which is unlikely to damage the vehicle. We formulate a velocity-dependent collision cost that uses collision prediction estimates and their associated uncertainties to enable the robot to only experience safe collisions during training while still approaching the desired task performance.

Consider an example scenario in which an autonomous drone is learning to fly in an obstacle-rich building. If the drone encounters a novel scenario, the drone will likely crash because the novel scenario is not contained within the training distribution of the reinforcement learning algorithm policy. However, by reasoning about its own policy's uncertainty, the drone can safely interact with the environment and avoid catastrophic failures while also increasing the diversity of its training distribution.

To realize this kind of safe, uncertainty-aware navigation in unknown environments, we propose a model-based learning approach in which the robot learns a collision prediction model and uses estimates of the model's uncertainty to adjust its navigation strategy. By using a speed-dependent collision cost together with uncertainty-aware collision estimates, our navigation strategy naturally chooses to move cautiously when uncertainty is high so as to experience only harmless low-speed collisions, and increases speed only in regions where the confidence of the prediction model is high.

Our main contribution is an uncertainty-aware collision prediction model that enables a robot to learn how to accomplish a desired task in an unknown environment while only experiencing gentle collisions. The collision prediction

model takes as input the current robot observation and a sequence of controls, computes the probability of a collision occurring along with an estimate of its uncertainty, and outputs a speed-dependent collision cost. The speed-dependent collision cost is a function of the model and its uncertainty, which enables the robot to automatically avoid catastrophic high-speed collisions by acting cautiously in novel situations. We use a deep neural network for the collision prediction model, which allows the model to cope with raw, high-dimensional sensory inputs. To obtain uncertainty estimates from the neural network, we leverage uncertainty estimation methods for discriminatively trained neural networks based on a combination of bootstrapping [5] and dropout [28, 7]. A model-based reinforcement learning algorithm then gathers samples using the neural network collision prediction model, which are aggregated and used to further improve the collision prediction model. Our empirical results demonstrate that a robot equipped with our uncertainty-aware neural network collision prediction model experiences substantially fewer dangerous collisions during training while still learning to achieve the desired task. We present an evaluation of our method with various parameter settings for both a simulated and real-world quadrotor, and a real-world RC car (Fig. 1), and demonstrate that our method offers a favorable tradeoff between training-time collisions and final task performance compared to baseline approaches that do not explicitly reason about uncertainty.

II. Related Work

In this work, we investigate how model-based reinforcement learning for robot collision avoidance can be made safe and reliable at both training and test time. Reinforcement learning has been applied to a wide range of robotic problems, ranging from locomotion and manipulation to autonomous helicopter flight [14, 4]. Model-free methods have been particularly popular due to their simplicity and favorable computational properties [24]. However, model-based methods are generally known to be more sample-efficient [3]. In this work, we adopt a model-based approach and learn an uncertainty-aware collision avoidance model; however, similar uncertainty estimation techniques could be extended also to model-free methods.

Several model-based robotic learning algorithms have been proposed that explicitly reason about uncertainty [3, 26]. Uncertainty estimates have been used to perform both risk-averse and risk-seeking, optimistic exploration [19]. The role of uncertainty estimation in our work is to avoid unsafe actions at training time until the model has gained sufficient confidence, which is largely orthogonal and complementary to prior work that seeks to improve exploration in order to accelerate learning. Combining these two directions is a promising direction for future work.

Uncertainty-aware model-based reinforcement learning has been explored in previous work using Bayesian models [25, 1]. While our work is similar in the overall aim, one of the central goals of our method is to directly process raw inputs from high-bandwidth sensors such as cameras, which necessitates the use of rich and expressive models, such as deep neural networks. Uncertainty estimation for deep neural networks is substantially more challenging, since these models are inherently discriminative. Recent work has proposed to use a Bayesian formulation of neural networks based on dropout [8], as well as to use the bootstrap for exploration [22], but not, to the best of our knowledge, for uncertainty estimation for the purpose of safety. In this work, we demonstrate that combining both dropout and bootstrap can yield actionable uncertainty estimates for reinforcement learning tasks.

There is much prior work on safe robot control for safety-critical systems such as autonomous cars [29], legged robots [31], and quadrotors [20, 30, 9]. A number of recent works have sought to address the question of safety for learning-based robotic systems. Methods based on reachability provide appealing theoretical guarantees, but cannot cope with rich sensory input and are often difficult to scale to high-dimensional systems [23, 17, 10]. Several works have suggested using discriminative models, including neural networks, to learn safety predictors [2]. These methods generally take the approach of training a model to predict whether an unsafe action will occur, and reverting to a hand-designed safety controller if such a potential failure is detected. Our method offers two advantages over this approach. First, by directly estimating model uncertainty, we do not rely on a discriminative safety estimator. This approach is preferred in environments where the model might encounter previously unseen inputs because a discriminative safety estimator cannot provide meaningful predictions for completely novel inputs; in short, the discriminative safety estimator may erroneously conclude that an unsafe environment is safe. In contrast, a statistical uncertainty prediction such as bootstrapping is more likely to estimate high uncertainty in novel environments. Secondly, our approach does not assume the existence of a manually designed safety control, but instead naturally reverts to more cautious exploratory behavior in the presence of uncertainty. This makes the approach more automated, and does not require a safety mechanism that can recover from arbitrary unsafe situations.

We use deep neural networks to estimate the probability of collision from raw sensory inputs. Combining deep networks with reinforcement learning has been an active area of research in recent years, with applications to video game playing [18], control of simulated robots [27, 16], and manipulation [15]. However, most of these applications focus on task complexity or learning speed, rather than explicitly considering uncertainty and safety during training. Prior work has considered safety at training time by using model-predictive control (MPC) with ground truth state information [11]. In contrast, our work does not assume any access to ground truth state, which is advantageous for real-world deployment.

III. Preliminaries

Our goal is to control a mobile robot, such as a quadrotor or a car, attempting to navigate an unknown environment. The task may be formally defined in terms of states $\mathbf{x}$, actions $\mathbf{u}$, dynamics $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$, and observations $\mathbf{o}$. We use $\mathcal{M}$ to represent the environment, including any potential obstacles.

We assume the robot's objective is encoded as a scalar cost function of the form

$$\mathcal{C}(\mathbf{x}_t, \mathbf{u}_t, \mathcal{M}) = \mathcal{C}_{\text{TASK}}(\mathbf{x}_t, \mathbf{u}_t) + \mathbb{1}_{\text{COLL}(\mathbf{x}_t, \mathcal{M})}\mathcal{C}_{\text{COLL}}(\mathbf{x}_t).$$

That is, the cost consists of an obstacle-independent task term $\mathcal{C}_{\text{TASK}}(\mathbf{x}_t, \mathbf{u}_t)$, which might include, for instance, flying to a desired position or in a desired direction, as well as an obstacle-dependent collision cost, which is given by the product of an indicator for collision $\mathbb{1}_{\text{COLL}(\mathbf{x}_t, \mathcal{M})}$, which is the only term that depends on the environment, and a collision cost $\mathcal{C}_{\text{COLL}}(\mathbf{x}_t)$ that may, for instance, penalize high-speed collisions more than relatively harmless low-speed collisions.

In a fully observable environment where $\mathcal{M}$ is known, the collision indicator can be evaluated exactly, and the problem can be solved by a standard optimal control method, such as the receding-horizon model-predictive control (MPC) approach we use in this work. In receding-horizon MPC, the robot solves an optimal control problem of the form

$$\min_{\mathbf{u}_t, \ldots, \mathbf{u}_{t+H}} \sum_{h=0}^{H} \mathcal{C}(\mathbf{x}_{t+h}, \mathbf{u}_{t+h}, \mathcal{M}) \text{ s.t. } \mathbf{x}_{t+h+1} = f(\mathbf{x}_{t+h}, \mathbf{u}_{t+h})$$

at each time step, it executes the action $\mathbf{u}_t$, advances to time step $t+1$, and repeats the optimization, effectively performing replanning at each time step. In this work, we assume that the dynamics, which might correspond, for instance, to the equations of motion of a quadrotor, are known at least approximately in advance. We instead focus on estimation of the cost, which depends on the unknown environment $\mathcal{M}$. If the environment is unknown and the indicator $\mathbb{1}_{\text{COLL}(\mathbf{x}_t, \mathcal{M})}$ cannot be estimated exactly, we can attempt instead to evaluate the probability of a collision using sensor observations, such as LIDAR or camera images. In this case, we can approximate the collision indicator according to

$$\mathbb{1}_{\text{COLL}(\mathbf{x}_{t+h}, \mathcal{M})} \approx P(\text{COLL}_{t+h} | \mathbf{x}_t, \mathbf{u}_{t:t+h} \mathbf{o}_t).$$

That is, we can estimate the probability of collision at a future time step $t+h$ based on the current state $\mathbf{x}_t$, the sequence of actions $\mathbf{u}_{t:t+h}$ that we intend to take, and the current observation $\mathbf{o}_t$, which might be used to deduce where the obstacles are located and thereby estimate the probability of collision, without prior knowledge about the environment.

In practice, we will slightly simplify the problem by predicting the probability of a collision at *any* time step h within the MPC horizon H. This approximation is not required, but yields a somewhat simpler model that we found performed equally well in practice, especially for relatively short-horizon MPC problems where $\mathcal{C}_{\text{COLL}}(\mathbf{x}_{t+H})$ doesn't change much over the MPC horizon. In this case, the full approximate cost at time $t+h$ evaluated using observation at time step t is given by

$$\mathcal{C}(\mathbf{x}_{t+h}, \mathbf{u}_{t+h}) \approx \mathcal{C}_{\text{TASK}}(\mathbf{x}_{t+h}, \mathbf{u}_{t+h}) + P_\theta(\text{COLL} | \mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t) \mathcal{C}_{\text{COLL}}(\mathbf{x}_{t+H}),$$

where we parameterize the probability of collision by model parameters θ, which corresponds to a class of parameteric conditional models. In our case, we present $P_\theta(\text{COLL} | \mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)$ with a neural network that outputs the parameter of a Bernoulli random variable, as we will discuss in Section IV-C. Our goal now is to learn the probability of collision model P_θ in such a way that avoids catastrophic failures (i.e., high-speed collisions) at both training and test time. However, for the robot to be able to act appropriately in novel situations, the robot must be able to reason about the uncertainty of the collision prediction model P_θ, as we will discuss in the next section.

IV. Uncertainty-aware Collision Prediction

The core component of our approach is an uncertainty-aware collision prediction model P_θ. Training this collision prediction model from experience presents a dilemma: the robot must first experience collisions in order to learn how to avoid collisions. We formulate a speed-dependent collision cost that uses uncertainty-aware collision estimates, resulting in the robot exploring cautiously when uncertainty is high and moving faster when uncertainty is low. This naturally arising behavior enables the robot to learn about collisions without experiencing catastrophic failures, and subsequently use these safe collision experiences to act more aggressively in the future.

An example application domain and desired application of the uncertainty-aware collision prediction model is the following: consider a quadrotor navigation task in which the objective is to fly fast and avoid collisions in an unknown environment. The quadrotor seeks to learn a collision prediction model that takes as input an image and a sequence of velocity commands and outputs the probability of collision. Initially, the quadrotor flies conservatively because the speed-dependent collision cost favors low-speed actions due to high uncertainty estimates of the collision prediction model. While flying conservatively, the quadrotor experiences safe collisions. These safe collisions, coupled with the associated images, are used to train the collision prediction model; the collision prediction model then learns how to associate images and velocity commands with the likelihood of colliding. As the algorithm continues and the collision prediction model uncertainty becomes low enough, the speed-dependent collision cost will favor high-speed flight.

A. Collision Prediction with Uncertainty

The collision prediction model P_θ takes as input the current state $\mathbf{x}_t$ and observation $\mathbf{o}_t$, a sequence of H controls $\mathbf{u}_{t:t+H}$, and outputs the probability the robot experiences a collision within the horizon. We formulate P_θ as a discriminative model using the logistic function $\text{L}(y) = 1/(1+\exp(-y))$, so that

$$P_\theta(\text{COLL} | \mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t) = \text{L}\big(\mathbb{E}[f_\theta(\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)]\big).$$

Here, $f_\theta(\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)$ is a random variable that corresponds to the real-valued output of our stochastic discriminatively trained model, which in our case corresponds to a modified neural network model that can produce uncertainty estimates. In general a variety of alternative models, including stochastic Bayesian models, could be used. Under this model, we can also define a risk-averse collision estimator $\widetilde{P}_\theta(\text{COLL} | \mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)$, given by

$$\widetilde{P}_\theta(\text{COLL} | \mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t) = \text{L}\big(\mathbb{E}[f_\theta(\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)] + \lambda_{\text{STD}} \sqrt{\text{Var}[f_\theta(\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)]}\big), \quad (1)$$

where λ_{STD} is a non-negative user-defined scalar and f_θ is scalar-valued function of the current state, observation, and a sequence of controls.

The risk-averse collision prediction model $\widetilde{P}_\theta$ accounts for uncertainty using the variance of the function f_θ: the larger the variance of f_θ, the less certain the underlying stochastic model is about the probability of collision. The standard model P_θ ignores this uncertainty, while the risk-averse model $\widetilde{P}_\theta$ uses the uncertainty to produce a conservative guess about the collision probability. Note that we use the variance of the sigmoid pre-activation value f_θ, since sigmoid probabilities are always in the range $[0, 1]$. Our goal is to increase the conservative estimate of collision if the model f_θ is uncertain (has high variance). However, if we use the sigmoid values, we might systematically underestimate the uncertainty. For example, imagine that the expected value of f_θ is a large negative number. Then, even if the variance is very large, the sigmoid expectation will be zero, which means that the sigmoid variance will be low. This is because the tails of the sigmoid flatten any variance in the model, making it invisible in situations where the mean prediction is close to 0 or 1. The hyperparameter λ_{STD} allows us to set how conservative the risk-averse model $\widetilde{P}_\theta$ should be, which allows the user to make intuitive tradeoffs between safety and task completion.

B. Velocity-Dependent Collision Cost

Based on the previously defined risk-averse model, we can now formulate a collision cost that will naturally favor slow, cautious exploration in regions of high uncertainty. The particular cost that we use has the form

$$\mathcal{C}_{\text{COLL}}(\mathbf{x}_t) = \lambda_{\text{COLL}} \|\text{VEL}_t\|^2, \tag{2}$$

where VEL_t is the robot velocity at time t and λ_{COLL} is a non-negative user-defined scalar that weights the relative importance of $\mathcal{C}_{\text{COLL}}$ versus $\mathcal{C}_{\text{TASK}}$. The full cost is then approximated using the risk-averse collision prediction model, according to

$$\begin{aligned}\mathcal{C}(\mathbf{x}_{t+H}, \mathbf{u}_{t+H}) \approx & \mathcal{C}_{\text{TASK}}(\mathbf{x}_{t+H}, \mathbf{u}_{t+H}) + \\ & \widetilde{P}_\theta(\text{COLL}|\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)\mathcal{C}_{\text{COLL}}(\mathbf{x}_{t+H}),\end{aligned} \tag{3}$$

With $\widetilde{P}_\theta$ and $\mathcal{C}_{\text{COLL}}$ defined, let us now confirm that Eqn. 3 will naturally favor cautious behavior when the collision prediction model is uncertain, and favor more aggressive behavior when the collision prediction model is confident. If the risk-averse collision prediction probability $\widetilde{P}_\theta$ is large, the robot is encouraged to move slowly in order to minimize $\mathcal{C}_{\text{COLL}}$. The collision prediction probability P_θ is large when $\mathbb{E}[f_\theta] + \sqrt{\text{Var}[f_\theta]}$ is large, which occurs whenever the model predicts a collision (i.e., $\mathbb{E}[f_\theta] \gg 0$) or when the model is uncertain (i.e., $\text{Var}[f_\theta] \gg 0$). On the other hand, if the risk-averse collision prediction probability is small, corresponding to a confident no-collision prediction, the robot can focus on minimizing $\mathcal{C}_{\text{TASK}}$ and move at fast speeds. The collision prediction probability $\widetilde{P}_\theta$ is small when $\mathbb{E}[f_\theta] + \sqrt{\text{Var}[f_\theta]}$ is small, which occurs when the model predicts no collision (i.e., $\mathbb{E}[f_\theta] \ll 0$) and the model is certain (i.e., $\text{Var}[f_\theta] \approx 0$).

C. Neural Network Collision Prediction Model

In order to be able to predict collisions from rich, high-dimensional sensory inputs, such as cameras or LIDAR measurements, we will use deep neural networks to estimate the probability of a collision. In the case of a standard deterministic, discriminatively trained neural network, f_θ would represent the pre-activation values in the network at the last layer, while P_θ is obtained by applying a sigmoidal nonlinearity to the pre-activations. Such a network can be trained on prior trajectories experienced by the robot simply by slicing all prior data into subsequences of length H, and inputting the states $\mathbf{x}_t$, observations $\mathbf{o}_t$, and the concatenated sequence of controls $\mathbf{u}_{t:t+H}$ into the model. The probability of collision labels are binary values recorded by the robot indicating whether a collision occurred, and we can obtain the label for each subsequence simply by checking whether a collision occurred between time steps t and $t + H$. The network can then be trained using standard stochastic gradient descent (SGD) with a cross-entropy loss on the final sigmoid output.

While such a model can provide accurate predictions about collision probability in regions of the environment close to the training data, it is inherently discriminative and deterministic. Such a deterministic model does not provide an estimate of its variance, and therefore is not by itself suitable for risk-averse collision prediction.

D. Estimating Uncertainty with Neural Networks

Standard predictive neural network models are trained discriminatively, which means that, even though the network might achieve a high accuracy on samples drawn from the same distribution as the training data, it is very difficult to predict how the network would behave on data drawn from a different distribution. While it is possible to train a neural network model that outputs a mean and a variance as its prediction [2], this model is not in general guaranteed to output high variances for unfamiliar inputs because the network is by definition trained only on the datapoints that are in the training set. Indeed, such a method for estimating variance is only effective at estimating the inherent noise in the data, and the variance estimates are not a meaningful indication of the model's own uncertainty about its predictions. To produce accurate uncertainty estimates for data that is outside of the training distribution, we must explore techniques that go beyond direct discriminative training. In order to obtain accurate uncertainty estimates from our model, we use two techniques: bootstrapping and dropout.

Bootstrapping: Bootstrapping [5, 6] is a simple and effective method of estimating model uncertainty using resampling that can be used with any discriminatively trained model. Given a dataset $\mathcal{D}$, B new datasets $\mathcal{D}^{(b)}$ are sampled with replacement from $\mathcal{D}$ such that $|\mathcal{D}^{(b)}| = |\mathcal{D}|$. Then, instead of training a single model $\mathcal{M}$ on the entire dataset $\mathcal{D}$, B different models $\mathcal{M}^{(b)}$ are trained on the datasets $\mathcal{D}^{(b)}$. The output prediction and uncertainty estimates are the sample mean and standard deviation of the outputs from the population of models.

The intuition behind bootstrapping is that, by generating multiple populations (using sampling with replacement) and

Algorithm 1 Neural net training with bootstrapping and dropout

input: dataset $\mathcal{D} = \{\mathbf{x}_t^{(i)}, \mathbf{u}_{t:t+H}^{(i)}, \mathbf{o}_t^{(i)}\}$, neural network model NN
for $b = 1$ to B **do**
  Sample a dataset of subsequences $\mathcal{D}^{(b)}$ from the full dataset $\mathcal{D}$ with replacement
  Initialize neural network $\text{NN}^{(b)}$ with random weights
  for number of SGD iterations **do**
    Sample datapoint $(\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)$ from $\mathcal{D}^{(b)}$
    Sample $\text{NN}_d^{(b)}$ by masking the units in $\text{NN}^{(b)}$ using dropout
    Run forward pass on $\text{NN}_d^{(b)}$ using $(\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)$
    Run backward pass on $\text{NN}_d^{(b)}$ to get gradient $g_d^{(b)}$
    Update model $\text{NN}^{(b)}$ parameters using $g_d^{(b)}$
  end for
end for

training one model per population, the models will agree in high-density areas of the population (i.e., low uncertainty regions) and disagree in low-density areas of the population (i.e., high uncertainty regions). This intuition is backed with theoretical guarantees [13]. However, for time- and resource-constrained applications such as robotics, usually only a limited number of bootstraps can be used, which often leads to inaccurate estimates of the model uncertainty.

Dropout: Dropout [7] is, by comparison, a computationally cheap method to improve uncertainty estimates. Dropout is commonly used to reduce overfitting in neural networks by randomly dropping units from the neural network during training [28]. Specifically, a given unit with dropout is set to 0 with probability p and left as its original value with probability $1 - p$ during training. Dropout prevents units from co-adapting (and thus overfitting) too much because different units are sampled for each forward pass, which effectively samples a new, but related, network during each step of training. Given a neural network $\text{NN}^{(b)}$, dropout in effect constructs a new randomized version of this network $\text{NN}_d^{(b)}$ by sampling independent Bernoulli random variables to act as masks on each neuron.

When dropout is used to reduce overfitting, it is only applied during training in order to force the units in the network to cope with stochastic removal of other units. In order to achieve high accuracy at test time, the dropout regularization is removed and all network weights are scaled by p to compensate for the increased level of activation. However, Gal and Ghahramani [7] showed that dropout can be used to obtain uncertainty estimates at test time by calculating the sample mean and standard deviation of multiple stochastic forward passes of the neural network *using dropout*. In this way, dropout can be viewed as an economical approximation to an ensemble method (such as bootstrapping) in which each sampled dropout mask corresponds to a different model. However, dropout underestimates the uncertainy because it acts roughly as a variational lower bound [7].

Algorithm 2 RL with Risk-Averse Collision Estimates

Initialize empty dataset $\mathcal{D}$
Initialize collision prediction model $\widetilde{P}_\theta$
for iter=1 to max_iter **do**
  Sample trajectories $\{\tau_i\}$ using MPC with cost $\mathcal{C}$
  Add samples $\{\tau_i\}$ to $\mathcal{D}$
  Train $\widetilde{P}_\theta$ using $\mathcal{D}$ (Alg. 1)
end for

Neural Networks with Bootstrapping and Dropout: Alg. 1 provides an overview of training neural networks with bootstrapping and dropout. From an initial dataset, multiple datasets are resampled with replacement, along with corresponding neural network model instantiations. While performing stochastic gradient descent on each bootstrap, different units are dropped each time a forward pass occurs; the gradient calculated by backpropagation is then used to update that specific bootstrap model's parameters.

At test time, we can evaluate the mean and variance of the ensemble by performing multiple forward passes on each network $\text{NN}^{(b)}$ using multiple instantiations of the dropout process, corresponding to $\text{NN}_d^{(b)}$. The random function $f_\theta(\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)$ then corresponds to sampling a network, sampling a dropout process, and evaluating the output. Thus, using neural networks with bootstrapping and dropout, we can estimate $\mathbb{E}[f_\theta(\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)]$ and $\text{Var}[f_\theta(\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)]$ for use in the risk-averse model $\widetilde{P}_\theta(\text{COLL}|\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)$.

E. Reinforcement Learning with Risk-Averse Collision Estimation

Alg. 2 provides an overview of how the uncertainty-aware collision prediction model is used in a model-based reinforcement learning algorithm. Each iteration of the algorithm, the cost function $\mathcal{C}$ is formed using the current uncertainty-aware collision prediction model $\widetilde{P}_\theta$. The model predictive controller then samples trajectories using cost $\mathcal{C}$. These sample trajectories are aggregated into a dataset containing all previous sampled trajectories. Then $\widetilde{P}_\theta$ is trained on the dataset according to Alg. 1 and the next iteration begins.

V. Experiments

We present simulated and real-world experiments to evaluate our uncertainty-aware collision prediction model, as well as our proposed model-based RL algorithm. We compare different settings for the parameters in our model, as well as evaluate its performance against a model-based approach that directly estimates the probability of collision, without explicitly accounting for uncertainty. Videos of the experiments can be found at https://sites.google.com/site/probcoll/.

Our collision prediction model $\widetilde{P}_\theta(\text{COLL}|\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)$ is a fully connected neural network with two layers with 40 ReLU [21] hidden units each. The activation of the last layer, which outputs the collision probability, is a sigmoid (see Eqn. 1). The model inputs are the concatenation of $\mathbf{x}_t, \mathbf{u}_{t:t+H}$ and $\mathbf{o}_t$. We trained the network using ADAM [12] and a standard

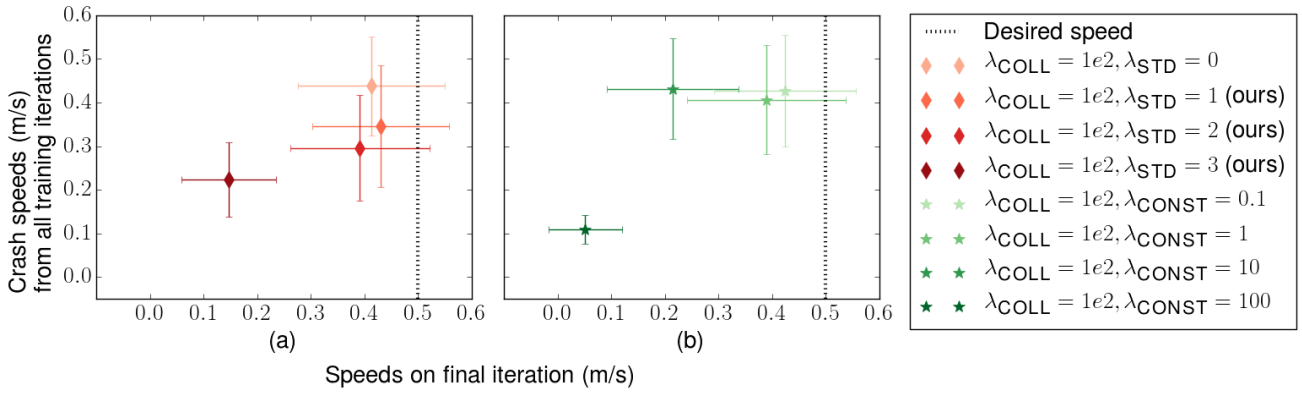

Fig. 2: **Comparison of safety versus task performance in simulation**: We investigated the effect of changing parameters in Eqn. 1 on crash speeds experienced during all training iterations (y-axis) versus the desired objective of flying forward at 0.5 m/s (x-axis). (a) shows the effect of changing λ_{STD} for our uncertainty-aware approach. (b) shows the effect of changing λ_{CONST} for a conservative baseline, in which the uncertainty in Eqn. 1 is replaced by a constant. Compared to the conservative baseline approach in (b), (a) shows our uncertainty-aware approach and its parameters can effectively trade off between safety and performance.

cross-entropy loss. For uncertainty estimation, the simulation experiments used 50 bootstraps and a dropout ratio of 0.2, while the real-world experiments used 5 bootstraps (due to real-time constraints) and a dropout ratio of 0.05.

At each time step, the receding-horizon MPC planner chooses among a set of fixed action sequences of horizon length H by evaluating cost $\mathcal{C}$ on each action sequence, and executes the first action of the minimal cost action sequence.

A. *Quadrotor experiments*

The simulated and real-world quadrotors have the same states, controls, and observations. We use a high-level representation of the quadrotor in which the control $\mathbf{u} \in \mathbb{R}^2$ is the commanded planar linear velocity, and therefore we assume the state $\mathbf{x}$ is estimated such that this level of control is feasible. However, we do not provide the state $\mathbf{x}$ as input to the collision prediction model. The observation $\mathbf{o} \in \mathbb{R}^{256}$ is a 16 by 16 grayscale image. The set of action sequences considered by the MPC planner at each time step consists of 190 straight-line, constant-velocity trajectories at various angles and speeds.

Simulated quadrotor: We first evaluate our uncertainty-aware collision prediction model in a simulated environment consisting of a cylindrical obstacle of radius 0.2m (Fig. 3). The objective $\mathcal{C}_{\text{TASK}}$ is to fly forward at 0.5 m/s, which is encoded as an ℓ^2 norm. The time horizon is $H = 6$ and each discrete time step corresponds to $\delta = 0.2$ seconds, therefore the planning horizon is δH seconds. At each time step, the quadrotor must decide on the sequence of actions using only the observation from a simulated monocular camera.

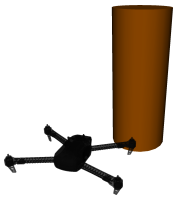

Fig. 3: Simulation

Fig. 2 compares safety versus task performance for different variants of Alg. 2. All experiments consist of 20 training iterations, with each iteration consisting of 20 on-policy rollouts from start states drawn from the same distribution. Each experiment was run 5 times with different random seeds.

First, we investigate the benefits of incorporating uncertainty into the cost by evaluating different values for λ_{STD} (Eqn. 1). Fig. 2a shows that, when not accounting for uncertainty (i.e., $\lambda_{\text{STD}} = 0$), the final task performance approaches the desired speed of 0.5 m/s. However, the quadrotor experiences high-speed collisions during training, as shown by the vertical axis. By accounting for uncertainty (i.e., $\lambda_{\text{STD}} > 0$), the quadrotor experiences lower speed collisions during training. The final task performance decreases if λ_{STD} is increased too much, which is expected: the more conservative the vehicle behaves during training, the longer it takes to learn the task. These results show that λ_{STD} allows the user to control their desired degree of risk during training and trade off safety against learning efficiency.

One reasonable question is whether accounting for uncertainty improves safety due to good uncertainty estimates, or simply because adding uncertainty to the collision probability simply makes the vehicle more cautious by penalizing high speeds. To answer this question, we compare our uncertainty-aware approach against a conservative baseline that replaces the uncertainty in Eqn. 1 with a constant λ_{CONST} (Fig. 2b). The experiments for $\lambda_{\text{CONST}} = 0.1, 1$, and 10 show no safety improvement, and also show decreased task performance compared to the baseline $\lambda_{\text{CONST}} = 0 \equiv \lambda_{\text{STD}} = 0$. The experiment for $\lambda_{\text{CONST}} = 100$ shows substantial safety improvement, but task performance is also substantially diminished. Compared to our uncertainty-aware approach with different settings of λ_{STD}, the baseline constant penalty approach with λ_{CONST} is ineffective at trading off between safety and performance, and always produces overly conservative motions. This indicates that uncertainty estimation is in fact reasoning about the vehicle's surroundings, rather than uniformly encouraging slower flight.

Another reasonable question to ask is whether simply increasing the collision cost λ_{COLL} induces safer training behavior. Our experimental results, included in the appendix, show that increasing λ_{COLL} does not lead to safer training behavior. Further simulation experiments and results are also provided in the appendix.

Real-world quadrotor: We evaluated our approach in a real-world environment consisting of a single obstacle, in which the objective is to fly around the obstacle (Fig. 1). Although the

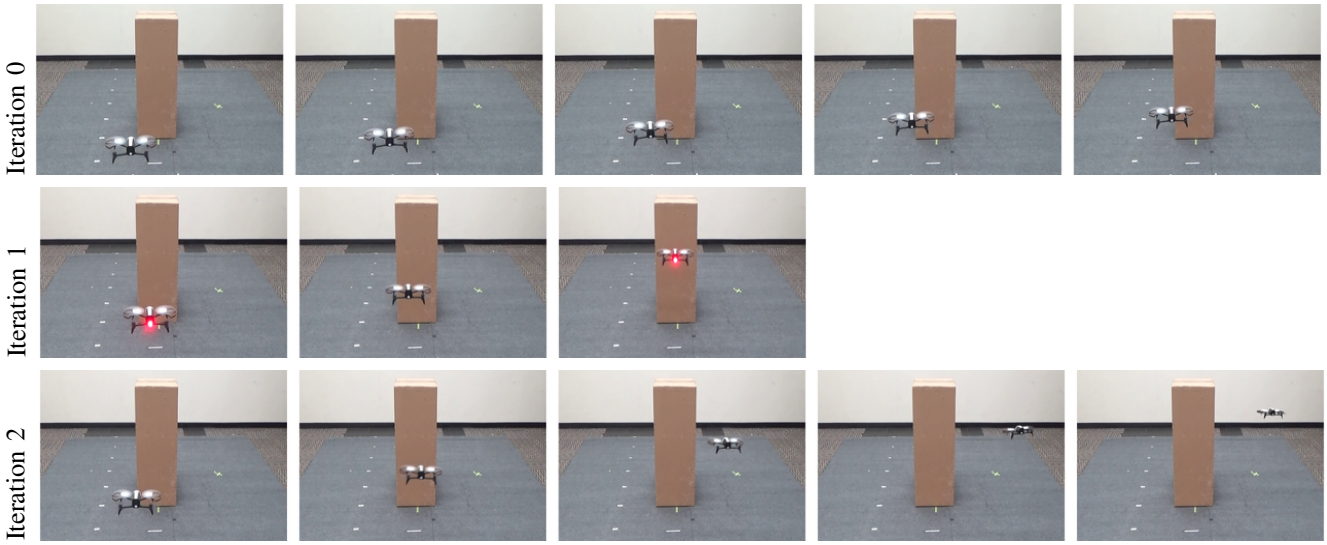

Fig. 4: **Real-world quadrotor experiments**: A Parrot Bebop 2 quadrotor learns to fly while avoiding obstacles using our uncertainty-aware reinforcement learning for collision avoidance algorithm (Alg. 2). Sample trajectories from the RL algorithm are shown above. On iteration 0, the quadrotor does not collide with the obstacle, but flies slowly. On iteration 1, the quadrotor flies faster, but collides with the obstacle. On iteration 2, the quadrotor avoids the obstacle while flying at high speed.

task of avoiding a single static obstacle is relatively simple, it is worth noting that the vehicle must perform this task entirely using real-world training data and only monocular images, while minimizing the number of collisions experienced during training. As such, the task is in fact quite challenging.

We ran our experiments using a Parrot Bebop 2 quadrotor. We used the ROS bebop_autonomy package, which allows the laptop to send linear velocity commands and receive the onboard images in real-time. The quadrotor's objective $\mathcal{C}_{\text{TASK}}$ is to fly forward at 1.6 m/s, which is encoded as an ℓ^2 norm. The time horizon $H = 3$ and each time step corresponds to $\delta = 0.5$ seconds.

All experiments consist of 5 training iterations, with each iteration consisting of 5 rollouts from 4 different initial positions. This experimental setup can be viewed in the online video. After each rollout, the quadrotor was manually reset to the next initial state. Note that this reset was solely done for minimizing experimental confounds for the purpose of evaluation, and is not a requirement of our approach. In principle, the vehicle could simply continue flying around the room and collecting data until good performance is achieved. Each experiment was initialized with 6 flight demonstrations provided by a human pilot. These demonstrations were the exact same for all experiments and consisted of 2 crashes and 4 successful flights around the obstacle. To prevent damage to the quadrotor, particularly for the baselines, a human pilot intervened if a crash was imminent; the algorithm therefore treated each intervention as a collision. Each experiment was run 5 times.

Fig. 4 shows images of our approach during the training process for an example experiment. In the beginning iterations, the quadrotor makes little progress and experiences collisions. As the RL algorithm progresses, the quadrotor is eventually able to fly around the obstacle at high speed.

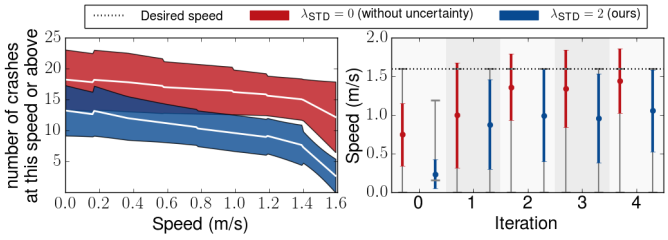

Fig. 5: **Comparison of safety versus task performance on a real-world quadrotor**: We investigated the effect of changing λ_{STD} in Eqn. 1 on crash speeds experienced during all training iterations (left) versus the desired objective of flying forward at 1.6 m/s (right) on a Bebop 2 quadrotor. For each value of λ_{STD}, results are combined from 5 complete experiments with the left plot displaying the mean and std and the right plot displaying the mean, std, and max/min. Increasing λ_{STD} leads to fewer crashes (left), but suboptimal performance (right).

Fig. 5 compares safety versus task performance when running our model-based RL algorithm (Alg. 2) without uncertainty ($\lambda_{\text{STD}} = 0$) and with uncertainty ($\lambda_{\text{STD}} = 2$). When accounting for uncertainty, the quadrotor experiences substantially fewer collisions, especially at higher speeds, but takes longer to approach the desired task performance.

B. Real-world RC car experiments

We evaluated our approach on an RC car (Fig. 8) in a simple obstacle avoidance task (Fig. 1). The car is parameterized by control $\mathbf{u} \in \mathbb{R}^2$ consisting of speed and steering angle and observation $\mathbf{o} \in \mathbb{R}^{576}$ consisting of a 32 by 18 grayscale image. We do not assume access to any underlying state $\mathbf{x}$.

Fig. 8: 1/10th scale RC car with a Logitech C920 Webcam and limit switch collision detectors.

The car's objective $\mathcal{C}_{\text{TASK}}$ is to drive at 1.2 m/s in any direction, which is encoded as an ℓ^2-norm. The time horizon was set to $H = 4$ and each discrete time step corresponds

Fig. 6: **Real-world RC car experiments**: An RC car learns to drive while avoiding obstacles using our uncertainty-aware reinforcement learning for collision avoidance algorithm (Alg. 2). A successful rollout is shown above.

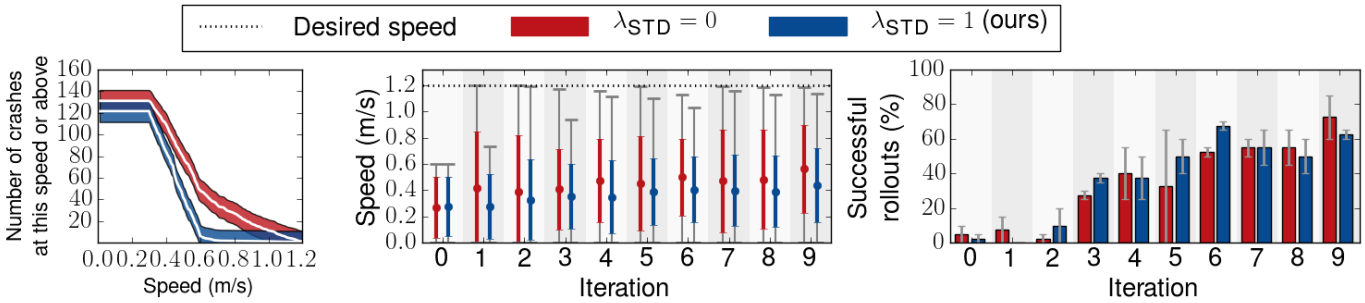

Fig. 7: **Comparison of safety versus task performance on a real-world RC car**: We investigated the effect of changing λ_{STD} in Eqn. 1 on crash speeds experienced during all training iterations (left) versus the task objective of driving at 1.2 m/s (middle) and the percentage of rollouts in which the RC car reached the end of the track (right). For each value of λ_{STD}, results are combined from 2 complete experiments. Our uncertainty-aware approach ($\lambda_{\text{STD}} = 1$) experiences 13% fewer crashes at speeds above 0.6 m/s (left) and comparable task performance (middle/right) compared to the baseline approach which does not account for uncertainty.

to $\delta = 0.5$ seconds. The set of action sequences considered by the MPC planner at each time step consists of 49 curving, constant-velocity trajectories at various steering angles and speeds.

All experiments consist of 10 training iterations, with each iteration consisting of 5 on-policy rollouts from 4 different initial states. Each rollout ended after either a collision or 10 time steps, therefore each experiment consists of approximately 15 minutes of real-world experience. After each rollout, the car was manually reset to the next initial state. No human demonstrations were used for initialization and each experiment was ran twice. Unlike in the quadrotor experiments, the car was allowed to collide at full speed and automatically registered collisions using limit switches mounted on the front of the car.

Fig. 6 shows images of our approach during the training process for an example experiment. Initially, the car is unable to avoid the obstacle and side walls, but eventually learns to avoid collisions.

Fig. 7 compares safety versus task performance when running our model-based RL algorithm (Alg. 2) without uncertainty ($\lambda_{\text{STD}} = 0$) and with uncertainty ($\lambda_{\text{STD}} = 1$). The final model-based planner for both approaches succeeds in navigating without colliding for almost 70% of the rollouts, which is a significant improvement over the initial policy. When accounting for uncertainty, the car experiences fewer high-speed collisions and achieves comparable speeds compared to when not accounting for uncertainty.

VI. Discussion and Future Work

We presented a model-based combined perception and control method for learning obstacle avoidance strategies that uses uncertainty estimates to automatically generate safe strategies. Our method is based on predicting the probability of collision conditioned on raw sensory inputs and a sequence of actions, using deep neural networks. This predictor can be used within a model-predictive control pipeline to choose actions that avoid collisions with high probability. In regions of high uncertainty, our risk-averse cost function naturally causes the robot to revert to a cautious low-speed strategy, without any explicit manual engineering of safety controllers or fail-safe mechanisms. We demonstrate our approach is safer compared to methods without uncertainty estimates in both a simulated and real-world quadrotor obstacle avoidance task, as well as a real-world RC car task.

Although our method produces cautious, uncertainty-aware behavior, it does not attempt to explicitly seek out successful strategies except through the MPC optimization. This can cause the algorithm to become stuck in bad local optima. For example, the suboptimal final performance of our approach in the real-world quadrotor experiments with $\lambda_{\text{STD}} = 2$ (Fig. 5). A promising direction of future work is to combine our method with optimistic—but still cautious—exploration strategies.

The success of our approach depends strongly on the accuracy of the uncertainty estimates. If the uncertainty estimates are overly optimistic, the robot may experience catastrophic failures. However, if the uncertainty estimates are overly pessimistic, the robot will be perpetually scared and the resulting policy will be suboptimal. This latter case may be another explanation for the suboptimal final performance of our uncertainty-aware approach in the real-world experiments (Fig. 5), therefore future work on developing new uncertainty estimators and characterizing their qualities is important for deploying RL algorithms on robotic systems.

Another promising direction for future work is to generalize our approach beyond collision prediction to other model-based reinforcement learning scenarios. The principle of uncertainty-aware prediction of future events can be readily applied to any feature of the environment, including the expected

cost, and exploring this extension to general reinforcement learning problems could produce effective and safe exploration techniques for a wide range of robotic scenarios.

VII. Acknowledgements

This research was funded in part by the Army Research Office through the MAST program, the National Science Foundation under IIS-1637443 and IIS-1614653, and the Berkeley Deep Drive consortium.

References

[1] F. Berkenkamp, A. Krause, and A. Schoellig. Bayesian Optimization with Safety Constraints: Safe and Automatic Parameter Tuning in Robotics. In *arXiv:1602.04450*, 2016.

[2] S. Daftry, S. Zeng, J. A. Bagnell, and M. Hebert. Introspective Perception: Learning to Predict Failures in Vision Systems. In *IROS*, 2016.

[3] M. Deisenroth and C. Rasmussen. PILCO: A Model-based and Data-Efficient Approach to Policy Search. In *ICML*, 2011.

[4] M. P. Deisenroth, G. Neumann, and J. Peters. A Survey on Policy Search for Robotics. In *Foundations and Trends in Robotics*, 2013.

[5] B. Efron and R. Tibshirani. The Jackknife, the Bootstrap and Other Resampling Plans. In *SIAM*, 1982.

[6] B. Efron and R. Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[7] Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *ICML*, 2016.

[8] Y. Gal, R. Mcallister, and C. Rasmussen. Improving PILCO with Bayesian Neural Network Dynamics Models. In *Data-Efficient Machine Learning workshop, ICML*, 2016.

[9] J. Gillula and C. Tomlin. Guaranteed Safe Online Learning via Reachability: Tracking a Ground Target using a Quadrotor. In *ICRA*, 2012.

[10] J. Gillula and C. Tomlin. Reducing Conservativeness in Safety Guarantees by Learning Disturbances Online: Iterated Guaranteed Safe Online Learning. In *RSS*, 2012.

[11] G. Kahn, C. Zhang, S. Levine, and P. Abbeel. PLATO: Policy Learning using Adaptive Trajectory Optimization. In *ICRA*, 2017.

[12] D.P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

[13] A. Kleiner, A. Talwalkar, P. Sarkar, and M. I. Jordan. The Big Data Bootstrap. In *ICML*, 2012.

[14] J. Kober, J. A. Bagnell, and J. Peters. In *Reinforcement learning in robotics: A survey*, volume 32, pages 1238–1274, 2013.

[15] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-End Training of Deep Visuomotor Policies. 2016.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *arXiv:1411.0247*, 2015.

[17] A. Majumdar and R. Tedrake. Funnel Libraries for Real-Time Robust Feedback Motion Planning. In *arXiv:1601.04037*, 2016.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and Riedmiller M. Playing Atari with Deep Reinforcement Learning. In *Workshop on Deep Learning, NIPS*, 2013.

[19] T. Moldovan and P. Abbeel. Safe Exploration in Markov Decision Processes. In *ICML*, 2012.

[20] M. Mueller and R. D'Andrea. Relaxed hover solutions for multicopters: Application to algorithmic redundancy and novel vehicles. *The International Journal of Robotics Research*, page 0278364915596233, 2015.

[21] V. Nair and G. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML*, 2010.

[22] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep Exploration via Bootstrapped DQN. In *NIPS*, 2016.

[23] T. Perkins and A. Barto. Lyapunov Design for Safe Reinforcement Learning. In *JMLR*, 2002.

[24] J. Peters and S. Schaal. Policy Gradient Methods for Robotics. In *IROS*, 2006.

[25] C. Richter, W. Vega-Brown, and N. Roy. Bayesian Learning for Safe High-Speed Navigation in Unknown Environments. In *ISRR*, 2015.

[26] J. Schneider. Exploiting Model Uncertainty Estimates for Safe Dynamic Control Learning. In *NIPS*, 1997.

[27] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust Region Policy Optimization. In *ICML*, 2015.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutidnov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In *JMLR*, 2014.

[29] C. Urmson and et. al. Autonomous Driving in Urban Environments: Boss and the Urban Challenge. In *Journal of Field Robotics*, 2008.

[30] M. Watterson and V. Kumar. Safe receding horizon control for aggressive MAV flight with limited range sensing. In *IROS*, 2015.

[31] P. Wieber. Viability and Predictive Control for Safe Locomotion. In *IROS*, 2008.

APPENDIX

We present additional results for the simulated quadrotor described in Section V. We compare the effect of varying the values of λ_{COLL} and λ_{STD} on safety (Fig. 9) and task performance (Fig. 10). Fig. 11 provides a more detailed analysis of the baseline conservative approach presented in Fig. 2.

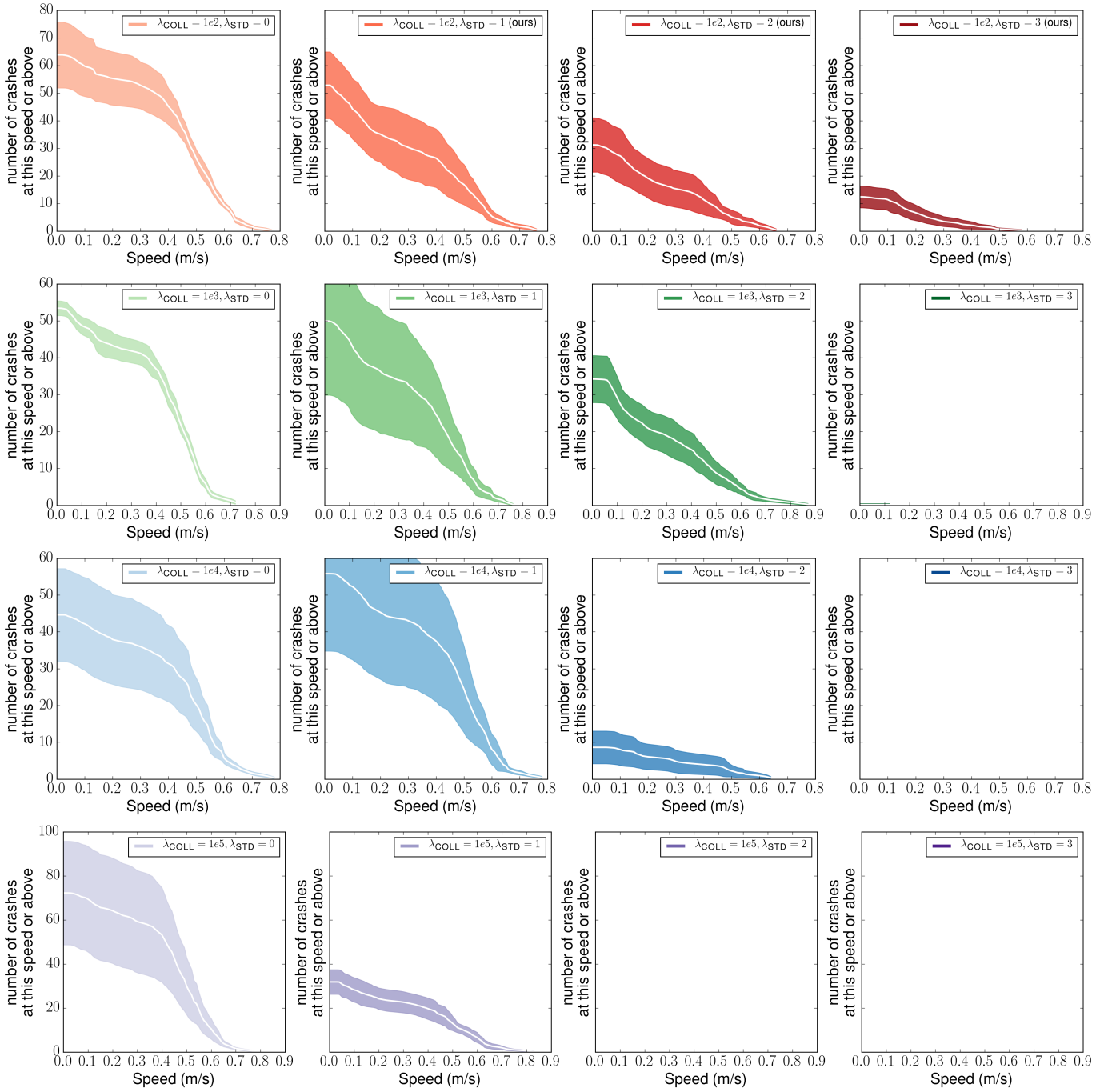

Fig. 9: **Safety comparison for different values of λ_{COLL} and λ_{STD}**: Each plot shows the number of training crashes at a given speed or above for a specific setting of λ_{COLL} and λ_{STD} averaged over 5 experiments. Each row corresponds to a fixed value for λ_{COLL} and each column corresponds to a fixed value for λ_{STD}. Examining the rows show that increasing λ_{STD} leads to fewer collisions, and of the collisions that do occur they are at lower speed. Examining the first column shows that increasing λ_{COLL} and not accounting for uncertainty does not lead to fewer collisions.

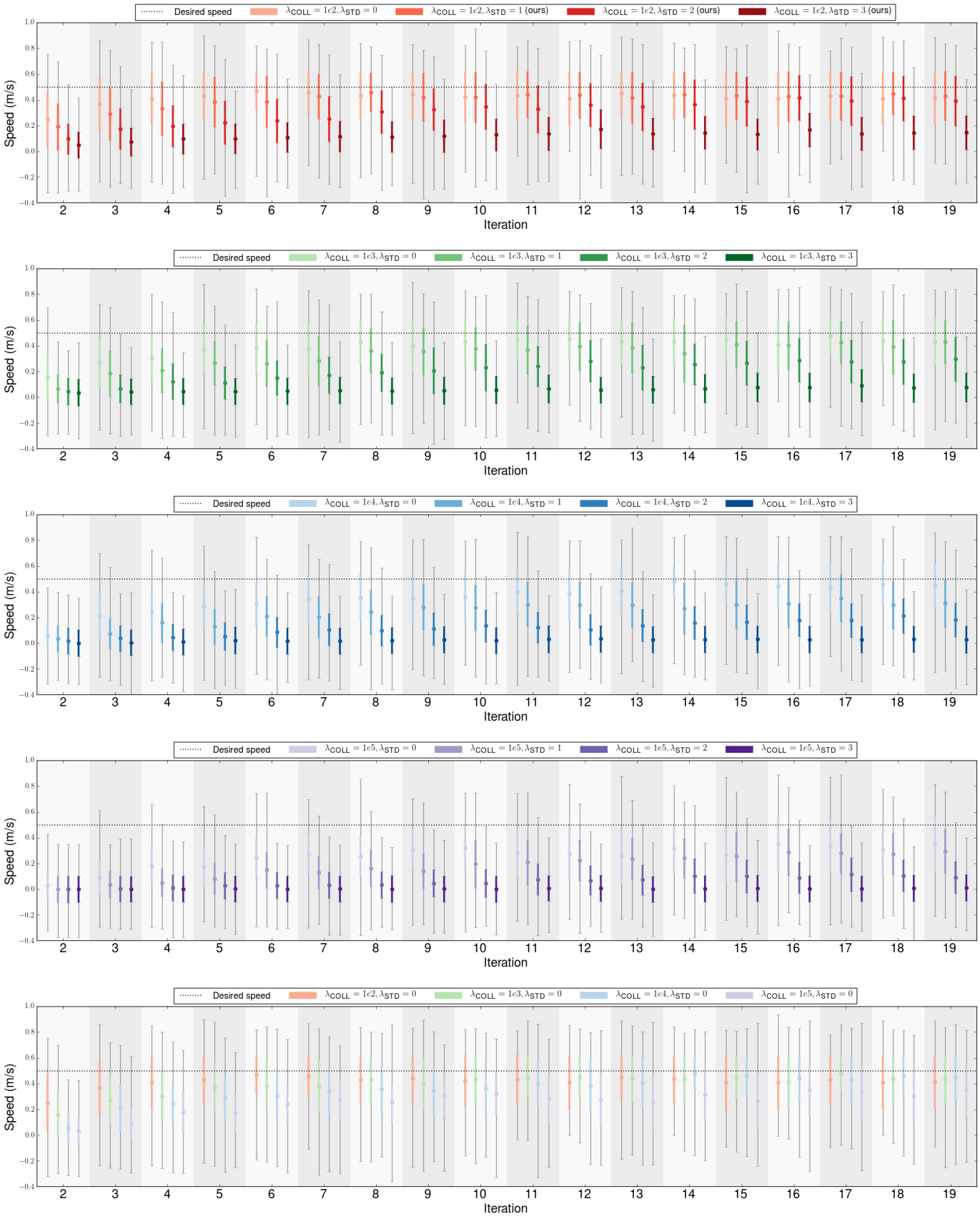

Fig. 10: **Comparison of task performance for different values of λ_{COLL} and λ_{STD}**: Each plot shows the task performance at each iteration of the RL algorithm. Results from each setting of λ_{COLL} and λ_{STD} were averaged from 5 experiments. The top four plots show that higher λ_{STD} results in slower progress towards achieving the optimal task performance. Furthermore, for large values of λ_{STD}, the final performance never reaches the optimal task performance. The bottom plot shows that when not accounting for uncertainty, λ_{COLL} has no significant effect on the final task performance.

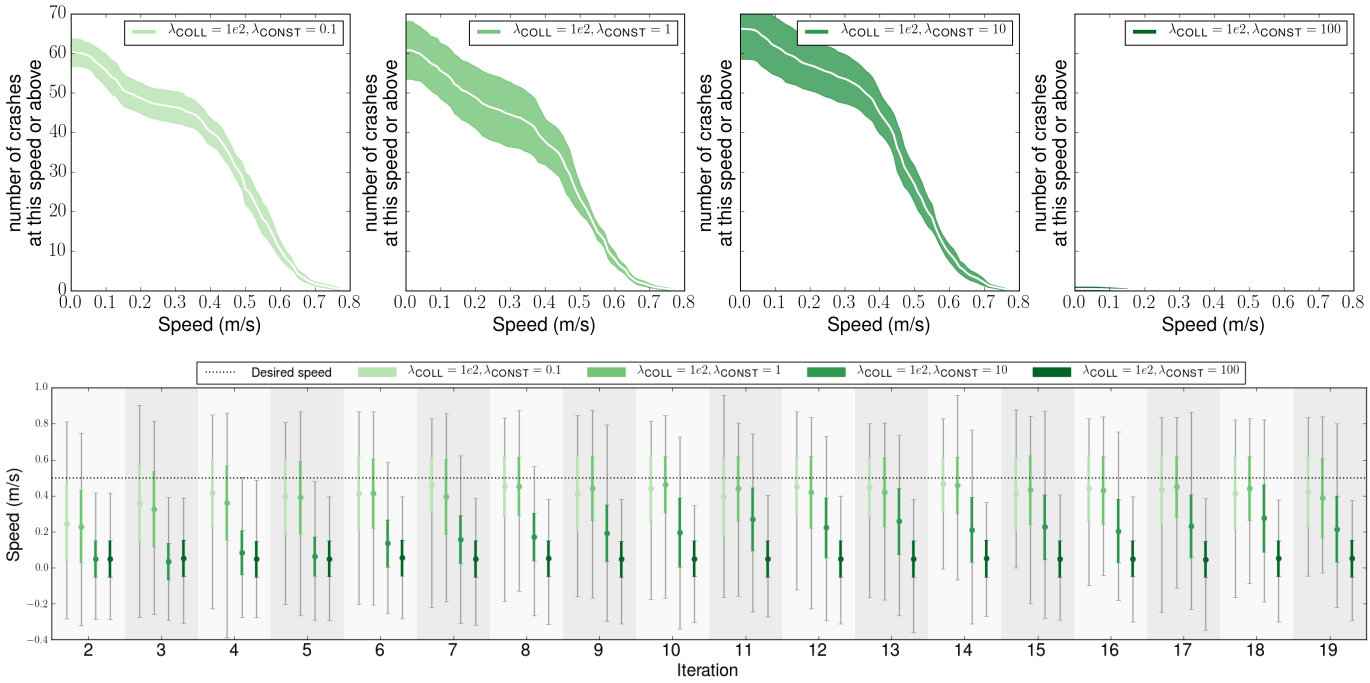

Fig. 11: **Comparison of safety versus task performance with a conservative approach**: Further analyzing the data presented in Fig. 2, these plots show the effect of changing λ_{CONST} for a conservative baseline, in which the uncertainty in Eqn. 1 is replaced by a constant. The effect of increasing λ_{CONST} on the final task performance (bottom) is similar to the effect of increasing λ_{STD} in our uncertainty-aware approach, however increasing λ_{CONST} does not necessarily increase safety (top).