IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

MonoMPC: Monocular Vision Based Navigation With Learned Collision Model and Risk-Aware Model Predictive Control

Basant Sharma, *Graduate Student Member, IEEE*, Prajyot Jadhav, Pranjal Paul, *Graduate Student Member, IEEE*, K.Madhava Krishna, *Member, IEEE*, and Arun Kumar Singh

***Abstract*—Navigating unknown environments with a single RGB camera is challenging, as the lack of depth information prevents reliable collision-checking. While some methods use estimated depth to build collision maps, we found that depth estimates from vision foundation models are too noisy for zero-shot navigation in cluttered environments. We propose an alternative approach: instead of using noisy estimated depth for direct collision-checking, we use it as a rich context input to a learned collision model. This model predicts the distribution of minimum obstacle clearance that the robot can expect for a given control sequence. At inference, these predictions inform a risk-aware MPC planner that minimizes estimated collision risk. We proposed a joint learning pipeline that co-trains the collision model and risk metric using both safe and unsafe trajectories. Crucially, our joint-training ensures well calibrated uncertainty in our collision model that improves navigation in highly cluttered environments. Consequently, real-world experiments show reductions in collision-rate and improvements in goal reaching and speed over several strong baselines.**

***Index Terms*—Vision-based navigation, planning under uncertainty, motion and path planning, collision avoidance.**

I. Introduction

NAVIGATION in unknown environments using a monocular RGB camera is well-suited for lightweight robotic platforms like aerial or compact ground robots, where size, weight, and power constraints make LiDARs impractical. Vision-only setups reduce hardware complexity and energy usage, supporting longer missions and broader deployment in cost-sensitive settings. However, monocular navigation remains fundamentally challenging due to the absence of direct depth perception, which hampers reliable collision-checking essential for safe operation in cluttered environments.

Recent progress in vision foundation models like DepthAnything [1], [2] and ZoeDepth [3] has enabled depth prediction from a single RGB image, opening new avenues for monocular navigation. Prior work has used these depth estimates to build local 3D or collision maps for planning [4]. However, depth estimates from existing models [1], [2], [3] are often too noisy or inconsistent for reliable collision-checking, particularly in close-proximity scenarios where small errors can result in collisions (see Fig. 1). In zero-shot settings, where the robot must generalize to unseen environments, these models fall short as standalone solutions and we further discuss these observation in Section IV.

Contributions: This work introduces a principled framework for autonomous navigation that, for the first time, directly reasons about uncertainty from estimated depth. At the core of our approach is a learned collision model that predicts a distribution over the minimum obstacle clearance for a planned trajectory. By conditioning this model on depth estimates, we can quantify collision risk at inference time. This risk is then integrated into a risk-aware Model Predictive Control (MPC) planner, which optimizes for actions that are both safe and goal-oriented.

A core innovation of our approach lies in appropriately transferring the noise in depth estimation to the actual downstream collision risk estimate. This is achieved by developing a joint learning pipeline that co-trains the collision model and the risk metric using both safe and un-safe trajectories. Our approach ensures that during the training process, the probabilistic collision model is aware of how its predictions impact downstream risk. This in turn, provides crucial supervision signal during training that regularizes the variance of the collision model and enables safe yet non-conservative behavior.

We validate our approach on real-world hardware, showing significant navigation gains over two baseline groups. The first, including the ROS stack and MonoNav [4], builds occupancy maps from noisy depth estimates, often causing unsafe or overly conservative behavior (Fig. 1). The second, NoMaD [5]-a diffusion-based end-to-end policy that directly regresses robot trajectory based on the input RGB image-lacks the precision needed for cluttered scenes. We also present additional statistical results validating the accuracy and statistical consistency of our learned collision model.

Received 24 July 2025; accepted 8 November 2025. Date of publication 8 December 2025; date of current version 16 December 2025. This article was recommended for publication by Associate Editor T. Do and Editor A. Bera upon evaluation of the reviewers' comments. This work was supported by in part by Estonian Research Council under Grant PSG753, in part by European Union and Estonian Research Council under Project TEM-TA101, and in part by SekMO Program 2021-2027.1.01.23-0419 co-funded by European Union. *(Corresponding author: Basant Sharma.)*

Basant Sharma and Prajyot Jadhav are with the Institute of Technology, University of Tartu, 50090 Tartu, Estonia (e-mail: basantsharma1990@gmail.com, basant.sharma@ut.ee).

Pranjal Paul and K.Madhava Krishna are with the International Institute of Information Technology, Hyderabad 500032, India.

Arun Kumar Singh is with the Institute of Technology, University of Tartu, 50090 Tartu, Estonia, and also with Estonian Aviation Academy, 61707 Tartu maakond, Estonia (e-mail: aks1812@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3641112, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3641112

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

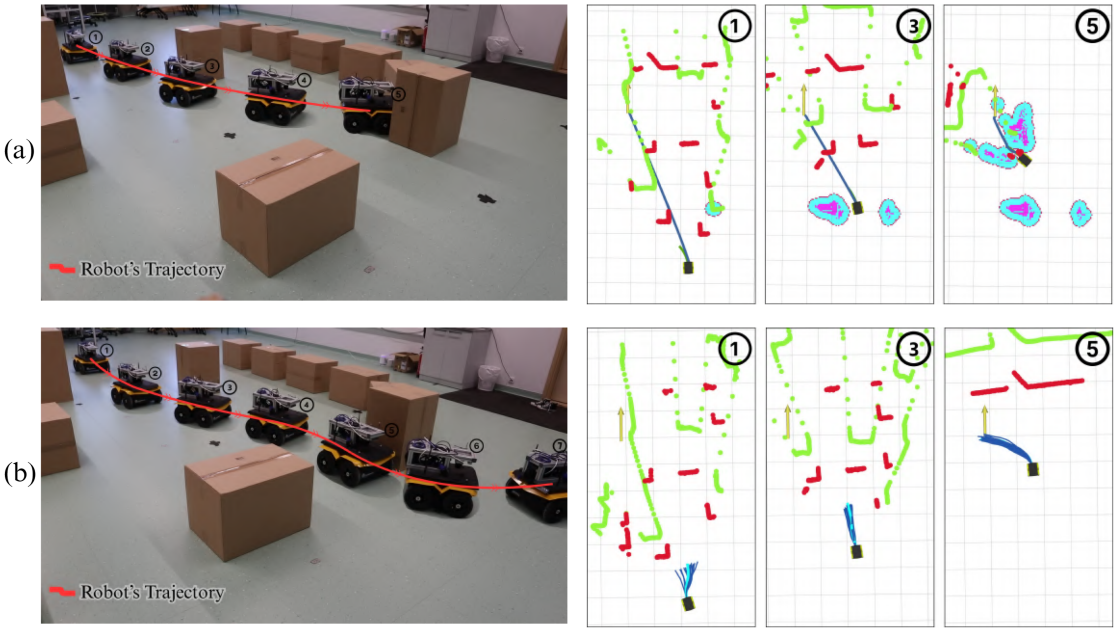

Fig. 1. Monocular navigation in cluttered environments using ROSNAV (top) vs. our approach (bottom). ROSNAV constructs cost maps directly from the estimated point cloud (green) generated by DepthAnything [1], which deviates significantly from the ground-truth (red), leading to incorrect free-space detection (e.g., top row, panel 3) and collisions. In contrast, our method treats the estimated point cloud as a conditioning input to a learned probabilistic collision model, integrated with a risk-aware MPC framework. Snapshots across time steps are shown for both methods (corresponding time indices are labeled).

II. Related Works

In the following, we restrict our review to works that use only monocular camera for navigation and exclude those that rely on depth sensors or LiDARs.

End-to-End Approaches for Visual Navigation: A common approach in monocular visual navigation is to train end-to-end models that map RGB images directly to control outputs like velocity commands or waypoints. Many rely on supervised learning via behavioral cloning or imitation learning [6], [7], [8], [9], [10], [11]. While simple and appealing, these methods often generalize poorly, are sensitive to perception noise, and lack interpretability. They also omit explicit modeling of geometry, uncertainty, and constraints, making them data-hungry and hard to adapt. Recent methods like ViNT [12] and NoMaD [5] improve generalization by using image-conditioned diffusion models to generate trajectories from RGB input. However, they depend on high-quality supervision and do not model collision uncertainty.

Another class of approaches focuses on learning spatial representations from monocular input [4], [9], [10]. For example, [4] constructs a map from estimated depth on-the-fly and then performs planning over it. However, Section IV shows that errors in occupancy predictions due to noise in estimated depth proves detrimental in cluttered environments.

Deep reinforcement learning (DRL) has also been explored for visual navigation, particularly in unseen or partially known environments [13], [14], [15]. While DRL enables flexible end-to-end learning from experience, it typically requires extensive training in simulation, careful reward shaping, and offers limited safety guarantees.

Predictive Models for Visual Navigation: Approaches like [16], [17], [18] learn action and image conditioned predictive models to simulate visual dynamics and leverage them for downstream planning and control, rather than directly learning policies. [19] adopt a slightly different approach and propose a visual MPC controller that outputs a sequence of velocity commands based on the current image and a trajectory of subgoal images. Authors in [20] regress depth images and actions to collision probabilities which is then minimized within an MPC framework. Although theoretically, this approach can be extended to work with estimated depth, it is unclear how the associated noise will affect the efficiency of the overall pipeline. Moreover, we believe that predicting obstacle clearance distribution is easier than directly predicting collision probabilities, as the supervision data for the former can be easily obtained. Our design choice also allows us to use sophisticated statistical tools to capture collision risk from the clearance samples in an efficient manner.

Improvement Over State-of-the-Art: Unlike end-to-end pipelines, our approach explicitly incorporates safety constraints by learning an action-conditioned collision model, placing us closer to [16], [17], [18], [19]. However, in contrast to these cited works, we explicitly reason about uncertainty in collision prediction, and introduce a novel method to learn task-aware uncertainty. Moreover, we also learn optimal parameters of our risk metric to improve the efficacy of our approach. On the implementation side, most monocular navigation methods demonstrate results in simple environments like hallways. In contrast, we show, for the first time, robust and reliable navigation in cluttered settings typically requiring LiDAR or depth sensors.

III. Main Algorithmic Results

Symbols and Notations: We use small/upper case normal-font to represent scalars. The bold-face small fonts represent vectors while upper-case variants represent matrices.

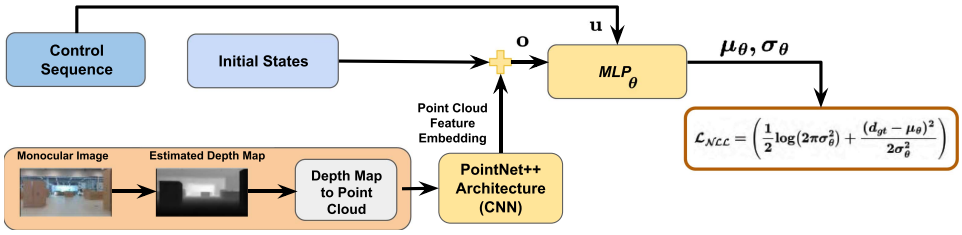

Fig. 2. Overview of baseline learning pipeline for our probabilistic collision model that predicts worst-case obstacle clearance along a trajectory. Given an RGB image and control sequence, we extract geometric features from the estimated point cloud using a pre-trained depth estimator and PointNet++. Combined with the initial robot state, these form the observation vector, which an MLP uses to predict the mean and variance of obstacle clearance. The learnable components (yellow blocks) are trained end-to-end using Gaussian negative log-likelihood loss.

A. Vision-Based Navigation as Risk-Aware Trajectory Optimization

We frame monocular vision-based collision avoidance and goal reaching as the following trajectory optimization.

$$\min_{\mathbf{u}} w_1 c(\mathbf{x}) + w_2 r(d(\mathbf{u},\mathbf{o})) + w_3 \left\|\mathbf{u}\right\|_2^2, \tag{1a}$$

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k),\ \mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} \forall k \tag{1b}$$

where $c(\cdot)$ represents the state-dependent cost function. The vector $\mathbf{x}_k$ represents the state of the robot at time-step k. The vector $\mathbf{x}$ is the concatenation of the states at different k. The function f represents the robot dynamics model. The control inputs are represented by $\mathbf{u}_k$. The vector $\mathbf{u}$ is formed by stacking $\mathbf{u}_k$ at different time-step k respectively. The scalar function d represents a learned probabilistic collision model, which takes as input the observations $\mathbf{o}$ and the control sequence $\mathbf{u}$ and outputs a distribution of worst-case obstacle clearance along the trajectory resulting from $\mathbf{u}$. We discuss the details of d in the subsequent sections. The first term c in (1a) minimizes the state cost, typically addressing path-following errors. The second term r captures collision-risk based on the predictions of d. The last term $\|\mathbf{u}\|_2^2$ in (1a) penalizes large control inputs, and the weights w_i tune the robot's risk-seeking behavior. Control bounds are enforced through (1b). We construct a MPC feedback loop by solving (1a)–(1b) in a receding horizon manner from the current state.

B. Modeling Risk Through Chance-Constraints

We model the robot's footprint as a circular disk of radius d_o. Thus, we can define $h(d(\mathbf{u},\mathbf{o})) = -d(\mathbf{u},\mathbf{o}) + d_o \leq 0$ as the control and observation dependent collision-avoidance constraint. We define risk r in terms of so-called chance constraints. These have the general form of $P(h(d(\mathbf{u},\mathbf{o})) \geq 0) \leq \varepsilon$, where P represents probability and ε is some constant. Thus, we define risk as:

$$r(h(d(\mathbf{u},\mathbf{o}))) = P(h(d(\mathbf{u},\mathbf{o})) \geq 0) \tag{2}$$

Intuitively, our risk model captures the probability of collision constraints being violated for a given control sequence $\mathbf{u}$ and current observation $\mathbf{o}$. Moreover, the risk depends on the distribution characterized by the collision model d. Our aim is to first learn a suitable d and then minimize the associated risk using (1a)–(1b).

A key challenge in using risk r (2) is that the left-hand side does not have an analytical form if the predictive distribution of $d(\mathbf{u},\mathbf{o})$ departs significantly from Gaussian. Thus, in the next subsection, we present a tractable surrogate risk model which can produce the same effect as r.

C. Maximum Mean Discrepancy (MMD) as Parameterized Risk Metric

Let us define a constraint residual function as

$$\overline{h}(d(\mathbf{u},\mathbf{o})) = \max(0, h(d(\mathbf{u},\mathbf{o})) \tag{3}$$

In the deterministic scenario, driving $\overline{h}(d(\mathbf{u},\mathbf{o}))$ to zero will push $h(d(\mathbf{u},\mathbf{o}))$ to the feasible boundary. In the stochastic case, (3) maps $d(\mathbf{u},\mathbf{o})$ to a distribution of constraint residuals. Let $\overline{h}(d(\mathbf{u},\mathbf{o})) \sim p_{\overline{h}}$. Although we don't know the parametric form for $p_{\overline{h}}$, we can be certain that its entire mass lies to the right of $\overline{h} = 0$. Moreover, as $P(h(d(\mathbf{u},\mathbf{o})) \geq 0)$ approaches zero, $p_{\overline{h}}$ converges to a Dirac-Delta distribution p_δ [21]. In other words, one way of reducing risk is to minimize the difference between $p_{\overline{h}}$ and p_δ. We formulate this distribution matching through the lens of Maximum Mean Discrepancy (MMD) [22], [21] and use the following surrogate for the collision-risk proposed in [21], [23].

$$r \approx r_{MMD}^{emp} = \left\|\hat{\mu}[\overline{h}] - \hat{\mu}[\delta]\right\|_{\mathcal{H}}^2, \tag{4}$$

where, $\hat{\mu}[\overline{h}]$ is called the empirical RKHS embedding of $p_{\overline{h}}$. It is computed through the following expression

$$\hat{\mu}[\overline{h}] = \sum_{i=1}^{i=N} \frac{1}{N} K_\lambda({}^i\overline{h}, .), \tag{5}$$

where, ${}^i\overline{h} = \overline{h}({}^i d(\mathbf{u},\mathbf{o}))$ and ${}^i d$ are the samples drawn from the predictive distribution characterized by d. In a similar manner, $\hat{\mu}[\delta]$ is computed based on the N samples of δ drawn from p_δ.[1] For any given N, as $r_{MMD}^{emp} \to 0$, $p_{\overline{h}} \to p_\delta$. Thus our aim is to compute a control sequence $\mathbf{u}$ which leads to as low as possible r_{MMD}^{emp}.

The K_λ is a positive definite function in RKHS known as the kernel function with the "reproducing" property (6) [22], where $\langle\cdot\rangle$ represents the inner product. Throughout this letter, we have used Laplacian kernel for which λ represents the kernel width.

$$K_\lambda(\mathbf{z},\mathbf{z}') = \langle K_\lambda(\mathbf{z},\cdot), K_\lambda(\mathbf{z}',\cdot)\rangle_{\mathcal{H}} \tag{6}$$

Importance of Kernel Parameter: The kernel parameter λ critically controls the MMD metric, $r_{\text{MMD}}^{\text{emp}}$, by shaping the embeddings $\hat{\mu}[\overline{h}]$ and $\hat{\mu}[\delta]$. An overly large λ causes the embeddings to collapse to a constant, making them indistinguishable and

[1] We can approximate p_δ through a Gaussian $\mathcal{N}(0,\epsilon)$, with an extremely small covariance ϵ ($\approx 10^{-5}$).

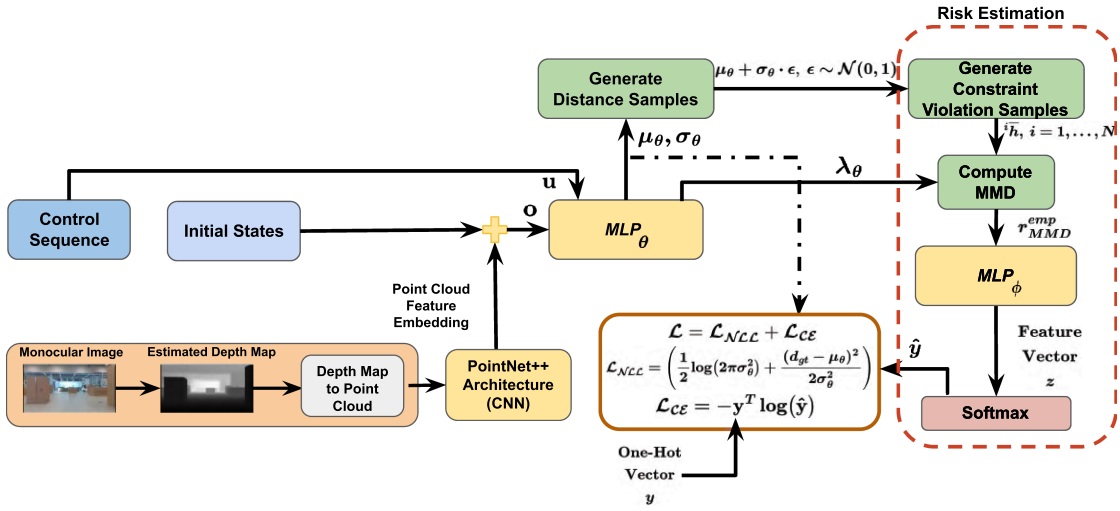

Fig. 3. Overview of our task-aware learning of probabilistic collision model. The previous baseline model of Fig. 2 had a weak supervision on predicted variance due to the absence of ground-truth uncertainty, often resulting in over- or underconfident predictions. To address this, we introduce downstream supervision via collision risk estimation. The observation vector and control sequence are passed through MLP_θ to predict the mean, variance, and a kernel parameter. Using the reparameterization trick, we generate obstacle clearance samples to compute constraint violations, forming an MMD-based risk representation. This is processed by MLP_ϕ, followed by a softmax layer. The learnable parts shown in yellow are trained end-to-end with Gaussian NLL and a cross-entropy loss.

insensitive to collision risk. Conversely, a λ that is too small makes the embeddings artificially distinct, leading to overly conservative behavior that flags open spaces as high-risk. To resolve this, our approach learns to predict an optimal λ directly from current observations and control sequences (Section III-E, Fig. 3).

D. Learning the Collision Model: The Baseline Approach

This section presents our approach for learning a probabilistic collision model $d(\mathbf{u}, \mathbf{o})$ which quantifies the worst-case obstacle clearance along a given trajectory. We model it as a Gaussian random variable with mean $\mu_{\boldsymbol{\theta}}(\mathbf{u}, \mathbf{o})$ and variance $\sigma_{\boldsymbol{\theta}}(\mathbf{u}, \mathbf{o})$, both of which are designed as feed-forward neural networks parameterized by weight $\boldsymbol{\theta}$.Our design choice captures observation noise in $\mathbf{o}$, primarily arising from depth or point-cloud estimates of the vision foundation model [24]. The detailed architecture is shown in Fig. 2. It consists of a pre-trained depth estimator that takes in RGB image and produces a depth-map. The depth-map is converted to a 2D point cloud and passed through PointNet++ [25] to get a feature embedding. We concatenate PointNet++ features with the initial state to get the full observation vector $\mathbf{o}$, which is then passed along with control sequence $\mathbf{u}$ to a Multi-Layer Perceptron (MLP) to get $\mu_{\boldsymbol{\theta}}(\mathbf{u}, \mathbf{o})$ and $\sigma_{\boldsymbol{\theta}}(\mathbf{u}, \mathbf{o})$. Both PointNet++ and MLP are trained in end-to-end manner using the following negative log-likelihood (NLL) function.

$$\mathcal{L}_{\mathcal{NLL}} = \left(\frac{1}{2}\log(2\pi\sigma^2_{\boldsymbol{\theta}}) + \frac{(d_{gt} - \mu_{\boldsymbol{\theta}})^2}{2\sigma^2_{\boldsymbol{\theta}}}\right), \tag{7}$$

where d_{gt} represents the ground-truth worst-case obstacle clearance along a given trajectory. We explain how these are obtained in Section IV.

E. Learning the Collision Model: The Down-Stream Task-Aware Approach

The core limitation of training with just the baseline NLL loss (7) is that the supervision on variance is rather weak since there is no ground truth variance available. Moreover, the training process is unaware of how its predictions are leveraged by the downstream task. Depending on how the training proceeds, the trained model can be either under-confident (high-variance) or overconfident(low-variance). For example, consider a scene where the actual d_{gt} signifies a collision-free scenario. That is $h(d_{gt}) = -d_{gt} + d_0 \leq 0$. Now, imagine that for this situation, the network of Fig. 2 predicts a $\mu_{\boldsymbol{\theta}}(\mathbf{u}, \mathbf{o})$ which is very close to d_{gt} but the variance is unreasonably higher. In such a case, the collision-free scenario will be incorrectly assigned high-risk due to uncertainty stemming from a higher variance.

The training process described in this section is designed to overcome the above described issues. Our main insight is that the supervision on variance can and should come from the downstream task of estimating collision risk based on the predicted $\sigma_{\boldsymbol{\theta}}$. To this end, we present the modified architecture in Fig. 3, which is obtained by adding a risk estimation head to the baseline architecture of Fig. 2. During training time, we use the predicted $\mu_{\boldsymbol{\theta}}(\mathbf{u}, \mathbf{o})$, $\sigma_{\boldsymbol{\theta}}(\mathbf{u}, \mathbf{o})$ to generate samples ${}^i d$ of the random variable $d(\mathbf{u}, \mathbf{o})$. We use the re-parameterization trick (8a) to generate samples in a differentiable manner. The samples ${}^i d$ should characterize zero risk if the associated d_{gt} is indeed collision-free. On the other hand, the samples should lead to a high risk value if d_{gt} is insufficient for collision avoidance. We can interpret this requirement as a two class classification problem conditioned on the samples ${}^i d$ and implicitly on the predicted variance $\sigma_{\boldsymbol{\theta}}(\mathbf{u}, \mathbf{o})$. The risk estimation head of Fig. 3 performs exactly this role.

The risk estimation head estimates collision constraint violations from ${}^i d$ (8b). Using these violations and the predicted kernel width $\lambda_{\boldsymbol{\theta}}$, the empirical MMD risk r^{emp}_{MMD} is computed. This risk is then passed through an MLP to yield a 2D feature vector $\mathbf{z}$, which a softmax layer normalizes to $\hat{\mathbf{y}}$. The ground-truth label for $\hat{\mathbf{y}}$ is given by $\mathbf{y}$ and is computed through (8c)

$$^i d = \mu_{\boldsymbol{\theta}} + \sigma_{\boldsymbol{\theta}} \cdot {}^i\epsilon,\ {}^i\epsilon \sim \mathcal{N}(0,1)\ \forall i = 1,\ldots,N \tag{8a}$$

$$^i h(^i d) = -{}^i d + d_o,\ {}^i\overline{h}({}^i d) = \max(0, {}^i h({}^i d)) \tag{8b}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Algorithm 1: Sampling-Based Optimizer to Solve (1a)–(1b).

1 M = Maximum number of iterations
2 Initiate mean ${}^{m}\boldsymbol{\nu}, {}^{m}\boldsymbol{\Sigma}$, at iteration $m = 0$ for sampling control inputs $\mathbf{u}$; Given observation vector $\mathbf{o}$, trained neural network $\boldsymbol{\theta}$(Fig. 3)
3 **for** $m = 1, m \leq M, m++$ **do**
4 Initialize $CostList$ = []
5 Draw batch of n control sequences $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_q, ..., \mathbf{u}_n)$ from $\mathcal{N}({}^{m}\boldsymbol{\nu}, {}^{m}\boldsymbol{\Sigma})$
6 Query collision model $\forall \mathbf{u}_q$: $(\mu_{\boldsymbol{\theta},q}, \sigma_{\boldsymbol{\theta},q}, \lambda_{\boldsymbol{\theta},q})$ = $\text{MLP}_{\boldsymbol{\theta}}(\mathbf{u}_q, \mathbf{o})$ // $\text{MLP}_{\boldsymbol{\theta}}$ denotes the learned model whose output $\mu_{\boldsymbol{\theta},q}, \sigma_{\boldsymbol{\theta},q}, \lambda_{\boldsymbol{\theta},q}$ denotes the predicted mean, predicted variance and the kernel parameter, respectively, for each control sequence $\mathbf{u}_q$
7 Compute N distance samples ${}^{i}d_q \sim \mathcal{N}(\mu_q, \sigma_q), \forall i = 1, \ldots, N$ and subsequently N constraint violation samples ${}^{i}\overline{h}_q$. Repeat this process $\forall q = (1, 2, \ldots, n)$ // ${}^{i}d_q$ denotes i-th obstacle clearance sample corresponding to the q-th control sequence. Similarly for ${}^{i}\overline{h}_q$
8 Compute $\hat{\mu}_q[{}^{i}\overline{h}_q]$ over the constraint violation samples ${}^{i}\overline{h}_q$ through (5) and subsequently $r^{emp}_{MMD,q}$. Repeat this $\forall q = (1, 2, \ldots, n)$ // Compute the MMD-based surrogate for collision risk for each q
9 Generate state trajectories $\mathbf{x}_q(\mathbf{u}_q), \forall q = (1, 2, \ldots, n)$
10 $ConstraintEliteSet \leftarrow$ Select top n_c batch of $\mathbf{u}_q, \mathbf{x}_q$ with lowest $r^{emp}_{MMD,q}$ // We sort control sequences $\mathbf{u}_q$ and trajectories $\mathbf{x}_q$ by lowest $r^{emp}_{MMD,q}$ values.
11 Define $c_q = w_1 c(\mathbf{x}_q) + w_2 r^{emp}_{MMD,q} + w_3 \left\|\mathbf{u}_q\right\|_2^2$
12 $cost \leftarrow c_q, \forall q$ in the $ConstraintEliteSet$
13 append each computed $cost$ to $CostList$
14 $EliteSet \leftarrow$ Select top n_e samples of $(\mathbf{u}_q, \mathbf{x}_q)$ with lowest cost from $CostList$.
15 $({}^{m+1}\boldsymbol{\nu}, {}^{m+1}\boldsymbol{\Sigma}) \leftarrow$ Update distribution based on $EliteSet$
16 **end**
17 **return** *Control inputs* $\mathbf{u}_q$ *and* $\mathbf{x}_q$ *corresponding to lowest cost in the* $EliteSet$

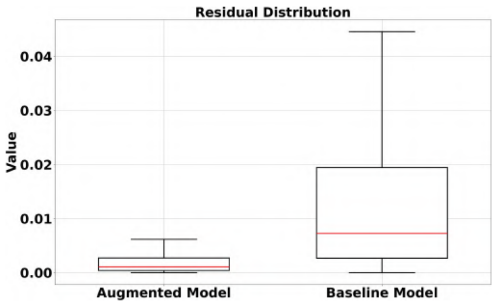

Fig. 4. Box plot of minimum residuals between sampled worst-case obstacle clearances and ground truth. The augmented model achieves lower median error and reduced variability compared to the baseline, indicating sharper and more consistent alignment with ground-truth worst-case obstacle clearances.

$$\mathbf{y} = \begin{cases} [0,1]^T, & \text{if } -d_{gt} + d_o \leq 0 \\ [1,0]^T, & \text{if } -d_{gt} + d_o > 0 \end{cases} \tag{8c}$$

$$\mathcal{L} = \mathcal{L}_{\mathcal{NLL}} + \mathcal{L}_{\mathcal{CE}} \; \mathcal{L}_{\mathcal{CE}} = -\mathbf{y}^T \log(\hat{\mathbf{y}}) \tag{8d}$$

The architecture of Fig. 3 is trained with a combination of NLL and a cross-entropy loss (8d). The latter acts as an implicit task-aware regularization on the predicted $\sigma_{\boldsymbol{\theta}}(\mathbf{u}, \mathbf{o})$.

F. Overall Approach

Algorithm 1 presents our overall approach wherein we solve (1a)–(1b) using the trained neural worst-case obstacle clearance model as our collision predictor. We adopt a sampling-based optimizer based on [26], wherein a batch of control sequences are drawn from a distribution which is gradually refined across iterations.

The algorithm initializes the sampling distribution (Line 2) and samples control inputs (Line 5), which are passed through a trained neural network (Line 6) to predict the mean $\mu_{\boldsymbol{\theta},q}$ and variance $\sigma_{\boldsymbol{\theta},q}$ associated with the distribution of obstacle clearances, as well as the kernel hyperparameter $\lambda_{\boldsymbol{\theta},q}$. These predictions generate N distance samples ${}^{i}d_q$, constraint violations ${}^{i}\overline{h}_q$ (Line 7), and subsequently $r^{emp}_{MMD,q}$ (Line 8). State trajectories $\mathbf{x}_q(\mathbf{u}_q)$ are then generated using the robot dynamics (Line 9). The n_c lowest-risk samples form the $ConstraintElliteSet$ (Line 10), from which costs are evaluated and stored (Lines 11–13). Finally, the top n_e samples form the $ElliteSet$ (Line 14) used to update the sampling distribution (Line 15) following [26].

IV. Validation and Benchmarking

A. Implementation Details

We implemented Algorithm 1 in Python using Jax [27] as the GPU-accelerated linear algebra back-end. The state cost consists of a running goal cost and has the following form:

$$c(\mathbf{x}) = \sum_k (x_k - x_f)^2 + (y_k - y_f)^2 \tag{9}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

where x_f, y_f denote the end-point coordinates. We used Algorithm 1 in a receding horizon manner from the current state to create a MPC feedback loop.

Data Collection: We collected two datasets: one via teleoperation and another using Algorithm 1-first with ground-truth LiDAR for collision avoidance to train the initial collision model and subsequently using Algorithm 1 with the learned collision model to collect further refinement data. Data were recorded at 10 Hz, with each sample consisting of:

- A standardized estimated point cloud of shape 300×2 obtained from Depth-Anything-V2 [1],
- Robot state $\mathbf{x} = (x, y, \psi, v, \omega)$, where (x, y) is position, ψ is heading, and v, ω are the linear and angular velocities (sampled uniformly from $\mathcal{U}[-1, 1]$ and clipped to $[0, 1]$ m/s and $[-1, 1]$ rad/s, respectively),
- A control sequence $\mathbf{u} = (\mathbf{v}, \boldsymbol{\omega})$ where $\mathbf{v}, \boldsymbol{\omega}$ are the concatenation of the linear and angular velocities, respectively, at different timesteps,
- Ground-truth clearance d_{gt} from LiDAR (see IV-A). This is the privileged information used during training.

Control Sequence Generation: Each control sequence consists of 50 steps over a $T = 5$ s horizon with sampling time of 0.1 s. Control sequences $\mathbf{u} = (\mathbf{v}, \boldsymbol{\omega})$ are generated via a uniform distribution in $[-1.0, 1.0]$ with $\mathbf{v}$ clipped to $[0, 1]$ m/s. The robot state evolves via:

$$x_{k+1} = x_k + v_k \cos(\psi_k)\Delta t,\ y_{k+1} = \quad y_k + v_k \sin(\psi_k)\Delta t$$
$$\psi_{k+1} = \psi_k + \omega_k \Delta t \tag{10}$$

Ground-Truth Distance Computation: For each trajectory resulting from the j-th control sequence $\mathbf{u}_j = (\mathbf{v}_j, \boldsymbol{\omega}_j)$, $\forall j = 1, \ldots, N_p$, the minimum distance to the LiDAR point cloud $\mathcal{P}_{\text{LiDAR}}$ is:

$$d_{gt,j} = \min_{k,\, \mathbf{p} \in \mathcal{P}_{\text{LiDAR}}} \left\| \begin{bmatrix} x_{k,j} \\ y_{k,j} \end{bmatrix} - \mathbf{p} \right\|_2 \tag{11}$$

Training Set-up: The probabilistic collision model (Section III-E) is trained in Equinox [28] on an RTX 5090 desktop. Inputs include estimated depth, vehicle states, and control sequences. The outputs are the mean and variance of obstacle clearance distribution. We collected 80 k point-cloud instances. In each of them, we used $N_p = 50$ control sequences with randomized robot initial state to generate trajectories along which we computed the ground-truth worst-case obstacle clearance. This yielded a total of $\approx$ 6M training points(point-cloud times N_p times random initial states). The total training time was $\approx$2.5 h, following the procedure in Sections. III-D–III-E.

Benchmarking Setup: We evaluate our approach using a Clearpath Jackal robot equipped with an Intel RealSense D435i RGB-D camera (RGB-only input, 69° horizontal FOV). For visualization and analysis, LiDAR data was recorded from a SICK TIM551 2D sensor. The system ran on ROS Noetic with an NVIDIA RTX 3080 GPU laptop connected to the Jackal over a local network. Depth estimation was performed in real time using TensorRT-accelerated Depth-Anything-V2 [1]. Experiments were conducted in indoor environments with static cardboard boxes to simulate clutter.

TABLE I
VALIDATION METRICS FOR THE LEARNED COLLISION MODEL

Method	UR (%) (lower is better)	CR (%) (lower is better)	ES (lower is better)	ECE over PIT (lower is better)
Alg. 1 with r^{emp}_{MMD}-based Augmented Model (Fig. 3)	**1.3**	**5.36**	**0.0236**	**0.0893**
Alg. 1 with Baseline Model (Fig. 2)	2.97	30.88	0.0917	0.1142

TABLE II
NAVIGATION WITH AND WITHOUT TASK-AWARE TRAINING

Method	% Collisions	% Stuck	Avg. Speed (m/s)	Max. Speed (m/s)
Alg. 1 with r^{emp}_{MMD}-based Augmented Model (Fig. 3)	**6.6**	**0**	0.32	**1.02**
Alg. 1 with Baseline Model(Fig. 2)	28.3	**0**	0.35	0.94
Deteministic	41.5	**0**	**0.52**	1.01

B. Validation of the Learned Collision Model

In this subsection, we validate the task-aware learning of the collision model (Fig. 3) and compare with the baseline approach (Fig. 2).

Predictive Accuracy: We analyze residuals between sampled worst-case obstacle clearances and ground truth. For each test case, we record the minimum absolute residual between samples from the predictive distribution and the ground truth, indicating whether any sample captures the true worst-case clearance. As shown in Fig. 4, the task-aware augmented model of Fig. 3 yields lower median residuals and a tighter inter-quartile range than the baseline (Fig. 2), indicating more consistent and reliable predictions.

Distributional Metrics: We evaluate three distributional metrics - Energy Score (ES), Probability Integral Transform (PIT), and Expected Calibration Error (ECE). The ES [29] measures both accuracy and sharpness, with lower values indicating predictions closer to the ground truth. Perfect calibration yields uniformly distributed PIT values [30] on [0, 1], while deviations from uniformity are summarized by ECE [31], where ECE ≈ 0 denotes ideal calibration.

Additionally, to evaluate safety, we compute four outcomes on the test-set: *Correctly Safe (CS)*, *Correctly Unsafe (CU)*, *Unsafe (U)*, and *Conservative (C)*. We define safety-critical rates of Unsafe (UR) and Conservative (CR):

$$\text{UR} = \text{U}/(\text{U} + \text{CU}) \times 100\%, \quad \text{CR} = \text{C}/(\text{C} + \text{CS}) \times 100\%$$

UR measures dangerous failures where unsafe states are overlooked, while CR captures over-conservatism where safe states are misclassified. For each outcome, risk r^{emp}_{MMD} is computed from predicted clearances and compared against the ground-truth risk from d_{gt}.

Table I shows that the r^{emp}_{MMD}-based augmented collision model (Fig. 3) surpasses the baseline (Fig. 2) across all metrics: UR ↓50%, CR ↓80%, ES ↓74%, and ECE ↓20%. These gains stem from task-aware variance supervision aligning uncertainty with collision risk, unlike the NLL-only baseline(Fig. 2), which produces mis-calibrated and unreliable predictions. The same trend appears in navigation results (Table II), where the task-aware model reduces collisions by 4 times. Table II also shows performance with deterministic collision model that performs the worst.

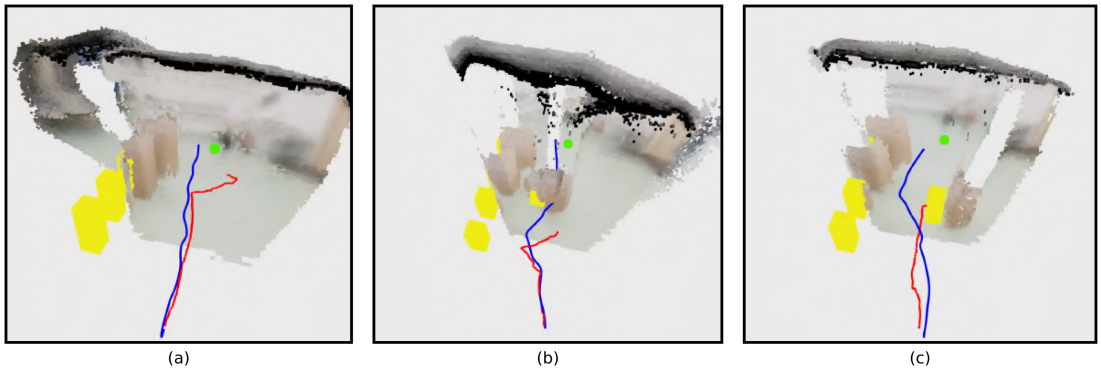

Fig. 5. Comparison between our approach (blue) and MonoNav [4] (red); goal in green. The noise in the estimated depth translates to erroneous 3D occupancy maps-the yellow cuboids (ground-truth obstacles) do not align with their reconstructed point clouds resulting in MonoNav getting stuck or colliding (b-c). In contrast, our approach is able to avoid the yellow cuboids based on the noisy estimated depth. We do not use MonoNav's 3D reconstruction; trajectories are overlaid solely for visualization and comparison.

TABLE III
COMPARISON WITH ROS NAVIGATION STACK

Method	% Collisions	% Stuck	Avg. Speed (m/s)	Max. Speed (m/s)
Alg. 1 with r^{emp}_{MMD}-based Augmented Model	**6.6**	**0**	0.32	1.02
ROSNAV (0.5 m/s)	25	51.6	0.38	0.59
ROSNAV (1 m/s)	48.3	48.3	**0.73**	**1.15**

TABLE IV
COMPARISON WITH MONONAV

Method	Easy Setting		Moderately Difficult Setting	
	% Collisions	% Stuck	% Collisions	% Stuck
Alg. 1 with r^{emp}_{MMD}-based Augmented Model	**0**	**0**	**0**	**0**
MonoNav [4]	**0**	**0**	20	80

TABLE V
COMPARISON WITH NOMAD

Method	% Collisions	% Stuck	Avg. Speed (m/s)	Max. Speed (m/s)
Alg. 1 with r^{emp}_{MMD}-based Augmented Model	**8.3**	**0**	0.28	**1.05**
NoMaD [5]	81.6	**0**	**0.46**	0.58

C. Comparison With ROS Navigation Stack (ROSNAV)

We benchmark our method against a classical planning stack from ROSNAV, which uses global planning followed by Dynamic Window Approach (DWA) for local control on costmaps built from estimated point clouds. To ensure a fair comparison with our planner's constraints (Algorithm 1), we evaluate ROSNAV at two speed limits: 0.5 m/s and 1.0 m/s. This classical approach fails mainly due to its inability to handle noise in the estimated depth. Offsets between estimated and true point clouds lead to false free space and collisions (Fig. 1(a)(3-4)), while noisy estimates inflate costmaps near goals, causing planning failures. These results show that such complex, dynamic errors cannot be corrected by simple heuristics. Table III quantifies these trends: ROSNAV often collides or fails at higher speeds, whereas our r^{emp}_{MMD}-based planner cuts collisions by up to 7 times and eliminates planning failures.

D. Comparison With MonoNav [4]

To highlight the benefits of our uncertainty-aware framework, we benchmark it against MonoNav [4], a baseline that plans directly on 3D reconstructions from estimated depth. In a series of quadcopter navigation experiments (40 total) through increasingly cluttered 2.5D environments (Fig. 5), we demonstrate the critical failure modes of ignoring depth uncertainty.

As shown in Fig. 5, noise in MonoNav's depth perception causes flawed reconstructions-tolerable in sparse scenes (a) but fatal in cluttered ones, leading to entrapment (b) or collision (c). Its simplistic primitive-based planner further limits adaptability to such errors. In contrast, our method explicitly incorporates depth uncertainty into risk-aware trajectory optimization, enabling successful navigation even in complex scenes. The executed trajectory (blue) consistently avoids all ground-truth obstacles (yellow cuboids), as confirmed by Table IV and the accompanying video showing smoother, faster motions than MonoNav.

E. Comparison With End-to-End Model NoMaD [5]

NoMaD [5] is a foundational model for image-based goal-directed navigation. To focus on its obstacle avoidance capabilities, we evaluate it in simplified scenarios where the goal lies directly ahead and is visible from the start position, requiring only obstacle avoidance. We fine-tuned NoMaD using tele-operation data (∼10k training points) and evaluated it in a test environment very similar to its training setup, giving it the best possible conditions to perform reliably.

Table V shows that NoMaD completes very few runs without collision, consistent with prior results [4], which report over 50% collision rates even in simpler hallway environments. Our settings were more complex, leading to even higher failure rates.

F. Computation Time

On an RTX 3080 laptop, planning runs in 0.01 s with 0.02 s for parallel depth estimation. On a Jetson Orin, planning time is 0.065 s, thus demonstrating the feasibility of deploying our system on mobile robots with real-time re-planning.

V. Conclusions and Future Work

In this work, we showed that depth estimates from vision foundation models are unreliable for direct use in autonomous navigation. For example, occupancy maps built from them are error-prone, causing failures in cluttered scenes (e.g., ROS Navigation Stack, MonoNav [4]). On the other hand, end-to-end vision-based methods such as NoMaD [5] lack explicit constraint handling and also perform poorly. We address these limitations by formulating monocular vision-based navigation as a risk-aware planning problem over a learned probabilistic collision model. Our training pipeline enforces well-calibrated uncertainty, improving downstream risk estimation and planning performance. The proposed approach operates in real-time on both commodity laptops and Jetson Orin, with successful demonstrations on ground and aerial platforms.

A limitation of our approach is the lack of temporal memory, as it ignores past images or point-clouds. This can cause oscillatory behavior in cluttered environments. Future work will focus on incorporating temporal information to mitigate this issue. Future work also includes extending our collision model to dynamic environments with time-varying collision models and accelerating planning through imitation learning. We also plan to explore end-to-end frameworks where collision model and action policy are jointly trained through demonstration data.

References

[1] L. B. Yang, Z. Kang, Z. Huang, X. Zhao, J. Xu Feng, and H. Zhao, "Depth anything v2," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, pp. 21875–21911. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/26cfdcd8fe6fd75cc53e92963a656c58-Paper-Conference.pdf

[2] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth anything: Unleashing the power of large-scale unlabeled data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 10371–10381.

[3] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, "ZoeDepth: Zero-shot transfer by combining relative and metric depth," 2023,. *arXiv:2302.12288*.

[4] N. Simon and A. Majumdar, "MonoNav: MAV navigation via monocular depth estimation and reconstruction," in *Proc. Symp. Exp. Robot.*, 2023, pp. 415–426.

[5] A. Sridhar, D. Shah, C. Glossop, and S. Levine, "NoMaD: Goal masked diffusion policies for navigation and exploration," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 63–70.

[6] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *Proc. 2018 IEEE Int. Conf. Robot. Automat.*, 2018, pp. 4693–4700.

[7] A. Loquercio, A. I. Maqueda, C. R. del- Blanco, and D. Scaramuzza, "DroNet: Learning to fly by driving," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 1088–1095, Apr. 2018.

[8] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Proc. Conf. Robot Learn.*, 2020, pp. 66–75.

[9] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural SLAM," in *Proc. Int. Conf. Learn. Representations*, 2020.

[10] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological SLAM for visual navigation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 12875–12884.

[11] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image," in *Proc. Robot. Sci. Syst.*, Cambridge, Massachusetts, Jul. 2017.

[12] D. Shah et al., "ViNT: A foundation model for visual navigation," in *Proc. 7th Annu. Conf. Robot Learn.*, 2023, pp. 711–733.

[13] Y. Zhu et al., "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3357–3364.

[14] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 5129–5136.

[15] S. Lian and F. Zhang, "TDAnet: Target-directed attention network for object-goal visual navigation with zero-shot ability," *IEEE Robot. Automat. Lett.*, vol. 9, no. 9, pp. 8075–8082, Sep. 2024.

[16] A. Bar, G. Zhou, D. Tran, T. Darrell, and Y. LeCun, "Navigation world models," in *Proc. Comput. Vis. Pattern Recognit. Conf.*, 2025, pp. 15791–15801.

[17] N. Hansen, H. Su, and X. Wang, "TD-MPC2: Scalable, robust world models for continuous control," in *Proc. Int. Conf. Learn. Representations*, 2024.

[18] P. Wu, A. Escontrela, D. Hafner, K. Goldberg, and P. Abbeel, "Daydreamer: World models for physical robot learning," in*Proc. Conf. Robot Learn.*, 2022, pp. 2226–2240.

[19] N. Hirose, F. Xia, R. Martıń-Martıń, A. Sadeghian, and S. Savarese, "Deep visual MPC-policy learning for navigation," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3184–3191, Oct. 2019.

[20] M. Jacquet and K. Alexis, "N-MPC for deep neural network-based collision avoidance exploiting depth images," in *Proc. 2024 IEEE Int. Conf. Robot. Automat.*, 2024, pp. 13536–13542.

[21] B. Sharma and A. K. Singh, "Trajectory optimization under stochastic dynamics leveraging maximum mean discrepancy," *IEEE Robot. Automat. Lett.*, vol. 10, no. 6, pp. 6079–6086, Jun. 2025.

[22] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *J. Mach. Learn. Res.*, vol. 13, pp. 723–773, 2012.

[23] B. Sharma and A. K. Singh, "MMD-OPT : Maximum mean discrepancy based sample efficient collision risk minimization for autonomous driving," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 19051–19068, 2025.

[24] 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, pp. 4759–4770.

[25] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet : Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5105–5114. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf

[26] M. Bhardwaj et al., "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation," in *Proc. Conf. Robot Learn.*, 2022, pp. 750–759.

[27] J. Bradbury et al., "JAX: Composable transformations of python NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

[28] P. Kidger and C. Garcia, "Equinox: Neural networks in JAX via callable PyTrees and filtered transformations," in *Proc. Differentiable Program. Workshop Neural Inf. Process. Syst.*, 2021.

[29] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *J. Amer. Stat. Assoc.*, vol. 102, no. 477, pp. 359–378, 2007, doi: 10.1198/016214506000001437.

[30] F. X. Diebold, T. A. Gunther, and A. S. Tay, "Evaluating density forecasts," National Bureau of Economic Research, Inc, *NBER Tech. Work. Papers 0215*, Oct. 1997. [Online]. Available: https://ideas.repec.org/p/nbr/nberte/0215.html

[31] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using Bayesian binning," in *Proc. AAAI Conf. Artif. Intell.*, 2015, pp. 2901–2907. [Online]. Available: https://api.semanticscholar.org/CorpusID:6292807