Density Planner: Minimizing Collision Risk in Motion Planning with Dynamic Obstacles using Density-based Reachability*

Laura Lützow[1,2], Yue Meng[2], Andres Chavez Armijos[3] and Chuchu Fan[2]

Abstract—**Uncertainty is prevalent in robotics. Due to measurement noise and complex dynamics, we cannot estimate the exact system and environment state. Since conservative motion planners are not guaranteed to find a safe control strategy in a crowded, uncertain environment, we propose a density-based method. Our approach uses a neural network and the Liouville equation to learn the density evolution for a system with an uncertain initial state. We can plan for feasible and probably safe trajectories by applying a gradient-based optimization procedure to minimize the collision risk. We conduct motion planning experiments on simulated environments and environments generated from real-world data and outperform baseline methods such as model predictive control and nonlinear programming. While our method requires offline planning, the online run time is 100 times smaller compared to model predictive control.**

The code and supplementary material can be found at `https://mit-realm.github.io/density_planner/`.

I. INTRODUCTION

Ensuring safety is crucial for autonomous systems. However, because of uncertainties such as measurement errors, external disturbances, and model errors, we cannot precisely predict the future state of the system or the environment, which complicates safety certification.

To deal with these uncertainties, there are two main directions in motion planning: Conservative approaches focus on robust safety by considering all possible situations via reachability analysis [1], planning a worst-case safe trajectory [2] or computing safe sets in which safety can be guaranteed for contained trajectories [3, 4]. However, in many situations, a guaranteed safe trajectory may not exist. Thus, other approaches seek trajectories that are safe with high probability [5, 6] by assuming linear dynamics or Gaussian distributions to propagate the uncertainties along the trajectories. Our method follows a similar philosophy as the latter approach but we handle nonlinear dynamics and the propagation of arbitrary initial state uncertainties by leveraging machine learning techniques.

[1] Department of Informatics, Technical University of Munich, Germany
[2] Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, USA
[3] Division of Systems Engineering, Boston University, USA

Recent work [7] solves the Liouville equation [8] with a neural network to learn the density distribution of reachable states. The follow-up work [9] conducts perturbation-based motion planning to ensure probabilistic safety. However, the method in [9] is computationally expensive and was not tested on realistic datasets. Inspired by this work, we design a differentiable framework to plan for feasible trajectories which minimize the collision risk. Beyond the system settings in [9], we also model the uncertainty from the environment and estimate the collision probability in an efficient manner. The proposed planning process can be described as follows: First, we predict the evolution of the state distribution in time for a closed-loop system under a parameterized control policy using a neural network. Then, we model the environment uncertainty as a probability occupancy grid map. Finally, we use a gradient method to optimize the policy parameters by minimizing the collision risk.

We conduct experiments on both simulated and real-world data [10] where our method can outperform a standard and a tube-based model predictive control (MPC) with a lower failure rate and lower online runtime. Furthermore, our method can be used for nonlinear system dynamics and uncertain initial states following an arbitrary probability distribution and can find plausible trajectories even when the worst-case collision is inevitable.

Our contributions are threefold: (1) To the best of our knowledge, we are the first to propose a density-based, differentiable motion planning procedure for nonlinear systems with state and environment uncertainties. (2) We provide an efficient method to compute the collision probability for closed-loop dynamics and an effective gradient-based algorithm for probabilistic safe trajectory planning. (3) We test our approach on simulated and real-world testing cases where we outperform state-of-the-art motion planning methods.

II. RELATED WORKS

Motion planning under uncertainties: Motion planning aims to find a trajectory that minimizes some cost function, fulfills the kinematic constraints, and avoids collisions [11]. Environment uncertainties can be treated as occupancy maps [12, 13, 14] where probability navigation functions [15], chance-constrained RRT [16], or constrained optimization techniques [17] are applied to ensure safety. However, most of these methods

assume that the initial state is precisely known, but this is often not given in the real world due to sensor noise. Thus, it is desirable to consider the whole initial state distribution for motion planning as it is done by density-based planning methods.

Density-based planning: Many density-based planning approaches assume that the initial state follows a Gaussian distribution. Under linear dynamics, the mean and covariance of the state distribution can be easily propagated through time [18], and the related optimal control problems can be solved by convex programming [19]. There exist several publications dealing with the covariance steering problem for nonlinear systems [20, 21, 22], but these methods are not directly applicable to non-Gaussian cases.

Controlling arbitrary initial distributions is closely related to optimal transport theory [23]. The work [24, 25] finds the optimal policy by solving the Hamilton-Jacobi-Bellman (HJB) equation. However, this method can only be used for full-state feedback linearizable systems and the convergence cannot be guaranteed. In [26], a primal-dual optimization is proposed to iteratively estimate the density and update the policy with the HJB equation, but this approach requires a time-consuming state discretization and cannot generalize to high-dimensional systems.

The closest work to ours is [9], which estimates the density for nonlinear systems and arbitrary initial distributions with neural networks (NN). With a perturbation method and nonlinear programming, a valid control policy is searched. However, the planning algorithm is very near-sighted, the collision-checking procedure is still computationally expensive, and no optimality statements can be claimed. Our approach overcomes these drawbacks by utilizing a more efficient collision computation method and a gradient-based optimization algorithm to minimize the collision probability.

III. PRELIMINARIES

Density prediction: First, we consider an autonomous system $\dot{\mathbf{x}} = f(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^n$ is the system state with initial density distribution $\rho_0(\mathbf{x})$ and $\rho(\mathbf{x}, t)$ is the density of $\mathbf{x}$ at time t. According to [8], $\rho(\mathbf{x}, t)$ follows the Liouville equation (LE):

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (f \cdot \rho) = 0, \tag{1}$$

where $\nabla \cdot (f \cdot \rho) = \sum_i^n \left(\frac{\partial}{\partial \mathbf{x}_i} f_i(\mathbf{x}) \rho(\mathbf{x}, t) \right)$. Following [26], we know that the density at the future state $\Phi(\mathbf{x}_0, t)$ is

$$\rho(\Phi(\mathbf{x}_0, t), t) = \rho_0(\mathbf{x}_0) \underbrace{\exp\left(- \int_0^t \nabla \cdot f\left(\Phi(\mathbf{x}_0, \tau)\right) d\tau\right)}_{g(\mathbf{x}_0, t)},$$
$$\tag{2}$$

where Φ denotes the flow map of the system. By solving Eq. (2), which can be accelerated by approximating

$g(\mathbf{x}_0, t)$ with a neural network [27], and using interpolation methods, the whole density distribution $\rho(\cdot, t)$ at time t can be estimated.

Problem formulation: In this work, we want to define an optimal control policy that steers an autonomous system to a goal state while minimizing the collision probability with dynamic obstacles. The state and input constraints must be satisfied, and the system dynamics can be nonlinear in the states.

In contrast to standard motion planning problems, we consider an uncertain initial state following an arbitrary but known probability density function. During the execution of the control policy, the state can be measured and used for closed-loop control.

IV. DENSITY-BASED MOTION PLANNING

In this section, we explain the proposed algorithm. First, we show how the density distribution of the closed-loop dynamics can be predicted given a parameterized control policy. Based on this, we provide an algorithm to compute the collision probability. Next, we introduce the cost function to evaluate the policy parameters. Lastly, we describe the gradient-based optimization method.

A. Density Estimation for Controlled Systems

A controlled system $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ with policy $\mathbf{u} = \pi(\mathbf{x}; \mathbf{p})$ and parameters $\mathbf{p}$ can be treated as an autonomous system with dynamics $f_{\mathbf{p}}(\mathbf{x})$, and hence, its density can be computed with Eq. (2).

Although our approach can adapt to arbitrary control policies, we choose $\mathbf{p}$ to parameterize a reference trajectory that is tracked by a tracking controller. By utilizing contraction theory [28] for the controller synthesis, the tracking error between the system and the reference trajectory can be upper-bounded and formal convergence guarantees can be provided. In this paper, the contraction controller is learned with the algorithm from [29] which requires the system to be control-affine. However, the remaining parts of the density planner do not need this special system structure such that the planner can be applied to arbitrary system dynamics when replacing the controller.

Analog to [27], we train an NN to approximate g and Φ from Eq. (2) using the initial state $\mathbf{x}_0$, the prediction time t_k, and the control parameters $\mathbf{p}$ as inputs.

Assuming that the initial density ρ_0 at state $\mathbf{x}_0$ is known, the density $\rho(\Phi(\mathbf{x}_0, t_k), t_k)$ can be predicted.

B. Computation of the Collision Probability

In this paper, we assume the probabilistic predictions for the evolution of a 2D environment are given as occupancy maps. The environment is evenly split into $C_x \times C_y$ grid cells along x and y directions. $P_{occ}(c_x, c_y, t_k)$ denotes the probability that cell (c_x, c_y) is occupied by an obstacle at time t_k. Obstacles can be road users, lanes, or static objects, and their respective occupancy probabilities can be generated by off-the-shelf environment

predictors as in [30, 31, 32]. The discretization of the environment is illustrated in Fig. 1 as a coarse grid map for visualization purposes.
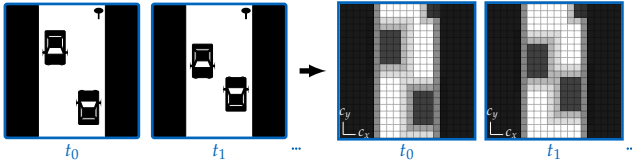

Fig. 1: Discretization of the environment.

We can compute the density distribution of the position of our ego vehicle as follows: First, we randomly sample initial states $\{\mathbf{x}^{(i)}(0)\}_{i=1}^{N}$ from the support of the given initial density distribution and predict the density $\{\hat{\rho}(\mathbf{x}^{(i)}(t_k), t_k)\}_{i=1}^{N}$ along the reference trajectory $\mathbf{p}$ with the trained NN. To compute the occupation probabilities of the ego vehicle $P_{ego}(c_x, c_y, t_k)$ for every cell (c_x, c_y) and timestep t_k, we assign to each predicted sample position $\hat{\mathbf{x}}^{(i)}(t_k)$ a corresponding cell $(c_x^{(i)}, c_y^{(i)})$ on the occupancy map. The densities of all samples falling into the same cell are averaged to represent the intermediate occupancy logits. Finally, the occupancy logits are normalized on the xy-plane to transform to the occupancy probability of the ego vehicle P_{ego}. The collision probability on (c_x, c_y) can be computed as:

$$P_{coll}(c_x, c_y, t_k) = P_{occ}(c_x, c_y, t_k) \cdot P_{ego}(c_x, c_y, t_k). \quad (3)$$

By integrating $P_{coll}(c_x, c_y, t_k)$ over the x and y dimensions, we can derive the overall collision probability at time t_k. For simplicity, we do not consider the physical dimensions of the system when predicting the future density distribution but assume a point shape.

C. Cost Function Formulation

After computing the collision probability for a given reference trajectory $\mathbf{p}$, we seek to find the reference trajectory that reaches a target goal and stays within the valid state space while minimizing the collision probability and the control effort. On that account, we create a differentiable cost function as a convex combination of four sub-objectives with weighting factors α to evaluate trajectory $(\mathbf{x}(\cdot), \mathbf{u}(\cdot))$:

$$J(\mathbf{x}(\cdot), \mathbf{u}(\cdot)) = \alpha_G J_G + \alpha_I J_I + \alpha_B J_B + \alpha_C J_C. \quad (4)$$

J_G penalizes the distance from the final state of the state trajectory to the goal state, weighted by the final density of the trajectory,

$$J_G = \rho(\mathbf{x}(t_N), t_N) ||\mathbf{x}(t_N) - \mathbf{x}_G||^2_{\mathbf{Q}_G}, \quad (5)$$

where t_N is the final time, $\mathbf{x}_G$ is the goal state for the ego vehicle to reach and $\mathbf{Q}_G$ is a weighting matrix. Similarly, J_I penalizes the control effort,

$$J_I = \sum_{k=0}^{N-1} ||\mathbf{u}(t_k)||^2_{\mathbf{Q}_I}. \quad (6)$$

The term J_B aims to keep the state trajectory within the valid state space and is defined as

$$J_B = \sum_{k=0}^{N} \rho(\mathbf{x}(t_k), t_k) \Big(||\mathbf{x}(t_k) - \mathbf{x}_{min}||^2_{\mathbf{Q}_{min}(\mathbf{x}(t_k))}$$
$$+ ||\mathbf{x}(t_k) - \mathbf{x}_{max}||^2_{\mathbf{Q}_{max}(\mathbf{x}(t_k))} \Big) \quad (7)$$

where $\mathbf{x}_{min}$ is the minimum and $\mathbf{x}_{max}$ is the maximum bound for the state space, and $\mathbf{Q}_{min}(\mathbf{x}(t_k))$ and $\mathbf{Q}_{max}(\mathbf{x}(t_k))$ are diagonal matrices whose diagonal elements are nonzero only when the corresponding state exceeds the maximum or falls below the minimum bound, respectively.

The last term, J_C, describes the cost for high collision probabilities. Due to the non-differentiability of the implemented binning method for computing P_{ego}, the collision probability cannot be minimized directly by a gradient-based optimization algorithm. Instead we compute a differentiable collision risk for $\mathbf{x}(t_k)$ as follows:

1) We approximate the gradients of P_{occ} between adjoining cells in x and y direction denoted by G_x and G_y, respectively.
2) Following the negative gradient, we can compute the desired position for sample $\mathbf{x}(t_k)$:

$$c_{x,\text{des}}(t_k) = c_x(t_k) - \beta G_x(c_x, c_y, t_k) \quad (8)$$
$$c_{y,\text{des}}(t_k) = c_y(t_k) - \beta G_y(c_x, c_y, t_k), \quad (9)$$

where β is the step size, and c_x and c_y are the x and y position of $\mathbf{x}(t_k)$ in grid coordinates, respectively.
3) After transforming the desired grid position to the position in real-world coordinates $(p_{x,\text{des}}, p_{y,\text{des}})$, we can compute the collision risk cost as

$$J_C = \sum_{k=0}^{N} P_{coll}(\mathbf{x}(t_k)) \Big(\big(p_x(t_k) - p_{x,\text{des}}(t_k)\big)^2$$
$$+ \big(p_y(t_k) - p_{y,\text{des}}(t_k)\big)^2 \Big), \quad (10)$$

where the collision probability for the reference trajectory is computed as $P_{coll}(\mathbf{x}(t_k)) = P_{occ}(\mathbf{x}(t_k), t_k) \rho(\mathbf{x}(t_k), t_k)$. The minimization of J_C yields positions $\mathbf{x}(t_k)$ with low occupancy probability, and consequently, decreases the collision probability of the trajectory.

Finally, the overall cost function can be minimized by gradient descent methods such as ADAM [33].

D. Optimization Approach

We propose the following two-step procedure for optimizing the reference trajectory:

Initialization: Since the optimization of Eq. (4) for an uncertain initial state is computationally complex and due to its non-convexity can get stuck in local minima, we wish to find a suitable initial guess first without performing any density predictions. This can be done by assuming an exactly given initial state which

is chosen as the mean of the initial density distribution and optimizing a large number of reference trajectories starting from there. The process is given as follows:

First, we randomly sample M parameter sets $\{\mathbf{p}^{(i)}\}_{i=1}^{M}$ from the admissible policy parameter space and recover the corresponding reference trajectories starting at the predefined initial state. Next, we calculate the cost with Eq. (4) for each parametrization by assuming that the system follows the corresponding reference trajectory accurately. We update the parameters $\{\mathbf{p}^{(i)}\}_{i=1}^{M}$ with gradient descent until a certain number of iterations is reached. Since we do not consider the uncertainty of the initial state, we assume that the density of each state is equal to one.

Given that we wish to guide the trajectories to the goal, we initialize α_B and α_C in Eq. (4) for all trajectories with zero. At each optimization iteration and for each trajectory pair, we check if the distance to the goal is smaller than a threshold. If this is true for one trajectory, the corresponding α_B is set to a nonzero value such that the state space constraints get enforced. For all trajectories where $\alpha_B \neq 0$, we perform another check: If the state space constraints are met, the collision cost will be considered by setting α_C to a nonzero value. This cost calculation procedure ensures that the trajectories are first steered to the goal, are next moved to the valid state space, and only then get optimized with respect to collisions. This procedure saves computation effort and leads to good trajectories much faster than when considering all cost terms from the beginning. Finally, we compare the costs of all optimized parameter sets and return the set $\mathbf{p}^*$ with the lowest cost.

Local optimization with density predictions: In the next step, we locally optimize the best parameter set $\mathbf{p}^*$ by taking the initial state uncertainties into account. First, we randomly sample S initial states $\{\mathbf{x}^{(i)}(0)\}_{i=1}^{S}$ from the given initial density distribution $\rho(\cdot, 0)$ and approximate their state trajectories $\mathbf{x}^{(i)}(\cdot)$ as well as their density $\rho(\mathbf{x}^{(i)}(\cdot), \cdot)$ with the neural density predictor when tracking reference trajectory $\mathbf{p}^*$.

Next, we compute the costs $J^{(i)}$ of all trajectories with Eq. (4) where all weights α are nonzero from the beginning. If the maximum number of optimization iterations is not reached, we calculate the gradient of the overall cost $\sum_{i=1}^{S} J^{(i)}$ and optimize $\mathbf{p}^*$.

V. EXPERIMENTS

While the proposed motion planning algorithm can be applied to all kinds of autonomous systems, we implement our method on a self-driving car in a congested uncertain environment. In this section, we introduce the model and problem for an autonomous-driving scenario. Additionally, we perform an ablation study on the proposed optimization method. Lastly, we compare our algorithm against three baseline methods in different environments. All experiments were conducted on an AMD Ryzen Threadripper 3990X (2.9GHz) with an
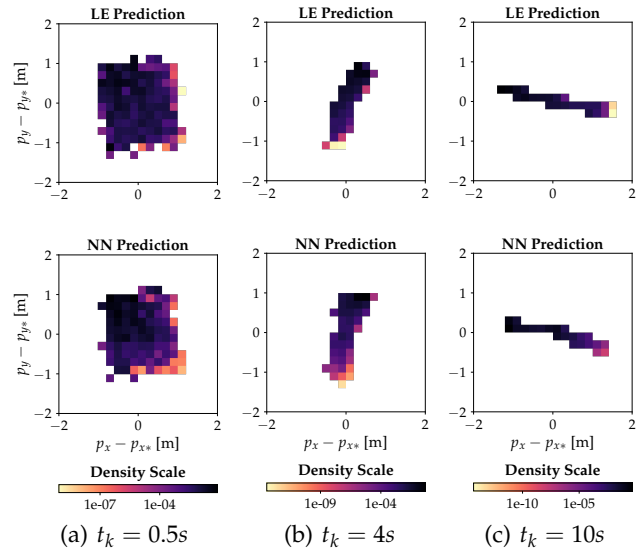
Fig. 2: Performance of the density NN.

NVIDIA A4000 16GB graphics card. The environments considered in the experiments are visualized in the supplementary video.

Model description: The system is described by a kinematic vehicle model [34] with input $\mathbf{u}(t) = [\omega(t), a(t)]^T$, where $\omega(t)$ is the angular velocity and $a(t)$ is the longitudinal acceleration. Additionally, to demonstrate the robustness of the proposed algorithm to disturbances, we included a constant sensor bias θ_{bias} for measuring the heading angle. The resulting vehicle dynamic model is defined as

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{p}_x(t) \\ \dot{p}_y(t) \\ \dot{\theta}(t) \\ \dot{v}(t) \\ \dot{\theta}_{\text{bias}}(t) \end{bmatrix} = \begin{bmatrix} v(t)\,\cos(\theta(t)) \\ v(t)\,\sin(\theta(t)) \\ \omega(t) \\ a(t) \\ 0 \end{bmatrix} \quad (11)$$

where $p_x(t)$ and $p_y(t)$ describe the position along the x-axis and y-axis respectively. $\theta(t)$ denotes the heading angle and $v(t)$ the longitudinal velocity. The control input $\mathbf{u}$ is generated by the learned contraction controller to track the reference trajectory on the basis of a biased state measurement $\hat{\mathbf{x}}(t) = [p_x(t),\ p_y(t),\ \theta(t) + \theta_{\text{bias}}(t),\ v(t),\ 0]^T$.

Next, we train the NN to predict the density along the reference trajectory. For this purpose, we define an architecture with seven fully connected layers and 150 neurons. A sample of the density predictions is shown in Fig. 2 where the NN predictions are compared against the solution of the LE as defined by Eq. (2). While the prediction of 10s-trajectories of the state and the density starting from 500 initial states takes on average 11.6s when integrating the system dynamics and using LE, the NN decreases this time to 0.3s.

Lastly, to plan the reference trajectory, we define the following weighting matrices:

$$\mathbf{Q}_G = \text{diag}([1, 1, 0, 0, 0]), \ \mathbf{Q}_I = \text{diag}([1, 1]),$$

where diag() denotes a diagonal matrix. The cost factors α in Eq. (4) are chosen such that the resulting cost gradient is sensible to J_G and J_C but is dominated by J_B when the trajectory is out-of-bounds. By tuning α_G and α_C, we can adjust the trade-off between low collision probabilities and goal-directed behavior. Furthermore, we clip the gradient if its absolute value becomes too big to ensure numerical stability.

Trajectory optimization: To analyze the performance of the reference trajectory computation, we compare the proposed gradient-based optimization method with a sampling-based and a search-based approach. For this purpose, we generate environments with a random number of uncertain stationary and dynamic obstacles. Similarly, the obstacle sizes, positions, uncertainties, and velocities are chosen randomly. Furthermore, we define the following criteria for the evaluation of the aforementioned methods:

- The *failure rate* is the percentage of environments where the considered method did not find a solution within five minutes. Failure cases are not considered when computing the average computation time or the cost scores.
- The collision risk increase (*CRI*), goal cost increase (*GCI*), and input cost increase (*ICI*) quantify the average increment of the collision risk, the goal cost, and the input cost when compared to the best possible method in each environment.
- The *computation time* is averaged over all evaluations.

The results are illustrated in Fig. 3. It can be seen that the proposed gradient-based approach achieves better performance in almost every criterion as compared to the other two methods. In each scenario, the gradient-based method was able to provide a valid solution while the sampling-based method failed in 35% of the environments. In addition, the distance from the final state of the planned trajectory to the goal was on average 0.2m as compared to the search-based (1.4m) and the sampling-based method (2.8m). Additionally,
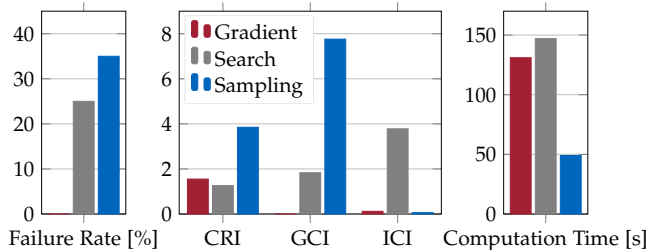

Fig. 3: Comparison of the optimization methods.

the collision risk of the gradient-based approach is comparatively small in most environments with an acceptable computation time.

Baseline methods: To evaluate the performance of the proposed density planner, we compare it against other collision-minimizing motion planning approaches.

Specifically, we define two receding horizon MPC algorithms for the given motion planning task.

The first implementation corresponds to an *standard MPC*, denoted as *M0*, which does not consider any disturbances or model uncertainties. The corresponding cost function is defined as:

$$
J_{\text{MPC}} = \alpha_I \sum_{k=h}^{h+H-1} ||\mathbf{u}(t_k)||_{\mathbf{Q}_I}^2 + \alpha_G ||\mathbf{x}(t_{h+H}) - \mathbf{x}_G||_{\mathbf{Q}_G}
$$
$$
+ \alpha_C \sum_{k=h}^{h+H} \left(P_{coll}(\mathbf{x}(t_k), t_k) \right)^2, \tag{12}
$$

where H denotes the prediction horizon, t_h is the current point in time, and α and $\mathbf{Q}$ are the same weighting factors and matrices as in Eq. (4).

Similarly, to handle imperfect measurements due to sensor bias, we define a *tube-based MPC* which minimizes the collision probability and enforces the state space constraints within a tube around the nominal trajectory. We will test implementations with the tube radii 0.3m, 0.5m, and 1m which are denoted as *M1, M2,* and *M3*, respectively.

For each environment, we assume that the minimum cost is achieved by an oracle, *O*, that computes the optimal solution of Prob. (12) with $h = 0$ and $H = N = 100$. To compute the optimal solution, we do not consider computation time constraints nor uncertainties. Thus, we assume that the oracle knows the true initial state and the exact system model.

Numerical evaluation: First, we analyze the motion planning methods in 50 randomly generated environments. The motion planners are compared using the same criteria defined in the ablation study with the difference that a failure is reported when $J_C > 10$ or when the distance from the final state to the goal is larger than 4.5m (the distance between the start and the goal position lies between 10 and 70m).
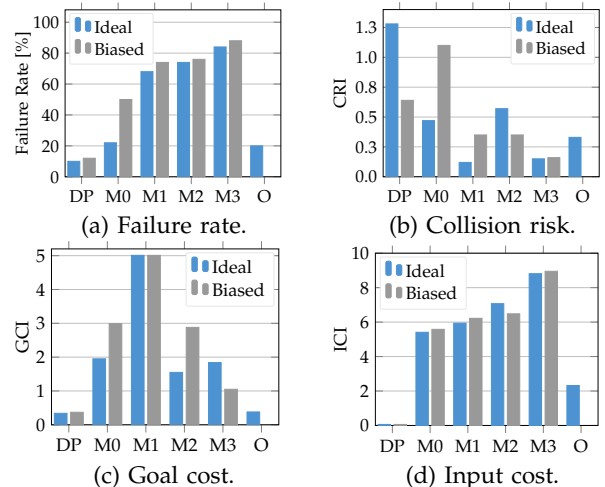

Fig. 4: Comparison in artificial environments.

In Fig. 4a, we can see that the proposed density planner (*DP*) is the most reliable method given a

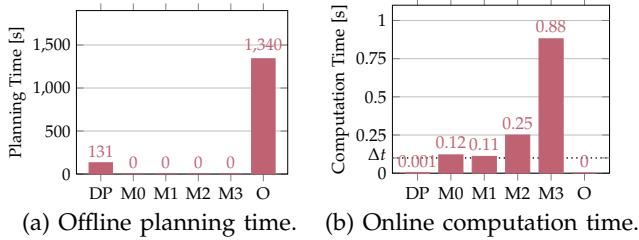(a) Offline planning time.   (b) Online computation time.

Fig. 5: Computational time evaluation.

success rate of 90% when using perfect measurements. Furthermore, all trajectories end within an average distance of $0.71m$ from the goal and low input cost. However, the collision risk in Fig. 4b is higher when compared to the other motion planning approaches due to the higher risk tolerance of the proposed algorithm: Even in very crowded environments, DP is able to reach the goal and stay below the acceptance tolerance for the collision risk ($J_C \leq 10$) such that these environments are considered in the computations of the CRI. The other planners, on the other side, either reach the goal with a very low collision risk or are very far off such that a failure is reported and the trajectory is discarded.

When the measurements are biased, the failure rate of DP is marginally higher than for the perfect measurement cases. This shows that the proposed algorithm is robust against deterministic disturbances. The collision risk is, on average, lower compared to the case of perfect measurements since the density NN was mostly trained with data where $\theta_{\text{bias}} \neq 0$ and hence better approximates the density for these cases. In contrast, the MPC methods present lower collision risks at the expense of higher goal costs and higher failure rates due to the conservative nature of the algorithms. Additionally, the input costs are considerably higher than for DP which is a byproduct of the short prediction horizon.

Fig. 5a shows the planning time which the algorithms need before starting the online control. DP takes on average $131s$ to compute the reference trajectory, which is a tenth of the planning time of the oracle. Furthermore, the oracle requires the true initial state when starting the planning while the density planner only needs the initial density distribution. The MPCs solve the corresponding optimization problems online and consequently do not need any planning time in advance. On the other hand, the online computational complexity of the MPC algorithms is high, as shown in Fig. 5b. Only M0 and M1 are able to solve the optimization problem in approximately real-time when using a time step of $\Delta t = 0.1s$. DP only has to compute the output of the contraction controller online to follow the precomputed reference trajectory. Hence, the resulting online computational complexity is significantly lower.

Evaluation on a real-world dataset: Next, we show that the proposed algorithm can be applied to real-world environments without modifications. However,

we assume access to the predicted occupancy maps for the environment. Here, we use the inD dataset [10] which contains a collection of naturalistic vehicle, bicyclist, and pedestrian trajectories recorded at German intersections by drones. We add stochasticity around each traffic participant by using a Gaussian distribution with a mean equal to the vehicle position and a standard deviation equal to the obstacle length plus 1m. Additionally, we skewed the distribution along the direction of movement.

To evaluate the performance, we compare our algorithm against M0, M2, and O. We choose M2 since its tube radius is a compromise between robustness and low computational complexity. We randomly sample start and goal positions for ten random time periods at three intersections and compute the costs for each planner. The failure rate is visualized in Fig. 6. DP achieves the best results; it finds a path to the goal in 24 of 30 environments when using perfect measurements.
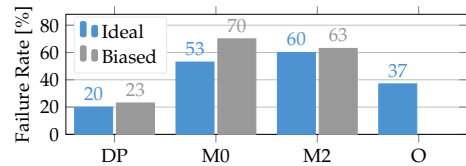

Fig. 6: Failure rate in real-world environments.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a density-based motion planning approach to ensure system safety under uncertainties. The approach is based on a differentiable cost function that considers the density evolution starting from arbitrary initial state distributions and can handle nonlinear system dynamics. The motion planner was applied to simulated environments generated from artificial data and from real-world data where it outperformed the baseline methods with significantly less online computational complexity.

However, the algorithm needs a planning time of approximately 130s to compute the reference trajectory on basis of the predicted environment occupations and the initial state distribution. As state-of-the-art environment predictors usually cannot look far ahead, the planning time must be decreased by using a more efficient implementation in C or C++ or by modifying our algorithm to perform in a receding horizon fashion. Furthermore, we plan to improve the density NN by using a more sophisticated architecture. For instance, the usage of a multi-dimensional convolutional NN could enable the prediction of the whole density distribution at once such that a differentiable expression for the collision probability could be derived. Thus, instead of minimizing the collision risk for density-weighted sample trajectories, we could minimize the collision probability directly. Additionally, we want to extend the approach to systems with stochastic disturbances by replacing the LE with the Fokker-Planck equation [35, 36].

REFERENCES

[1] M. Althoff. "Reachability Analysis and Its Application to the Safety Assessment of Autonomous Cars." PhD thesis. 2010.

[2] J. Lofberg. "Approximations of closed-loop minimax MPC." In: *IEEE International Conference on Decision and Control*. 2003, pp. 1438–1442.

[3] S. V. Rakovic, E. C. Kerrigan, K. I. Kouramas, and D. Q. Mayne. "Invariant approximations of the minimal robust positively invariant set." In: *IEEE Transactions on automatic control* 50.3 (2005), pp. 406–410.

[4] F. Gruber and M. Althoff. "Computing safe sets of linear sampled-data systems." In: *IEEE Control Systems Letters* 5.2 (2020), pp. 385–390.

[5] A. Mesbah. "Stochastic model predictive control: An overview and perspectives for future research." In: *IEEE Control Systems Magazine* 36.6 (2016), pp. 30–44.

[6] A. Wang, A. Jasour, and B. C. Williams. "Non-gaussian chance-constrained trajectory planning for autonomous vehicles under agent uncertainty." In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6041–6048.

[7] Y. Meng, D. Sun, Z. Qiu, M. T. B. Waez, and C. Fan. "Learning Density Distribution of Reachable States for Autonomous Systems." In: *Conference on Robot Learning*. 2022, pp. 124–136.

[8] M. Ehrendorfer. "The Liouville Equation and Prediction of Forecast Skill." In: *Predictability and Nonlinear Modelling in Natural Sciences and Economics*. 1994, pp. 29–44.

[9] Y. Meng, Z. Qiu, M. T. B. Waez, and C. Fan. "Case Studies for Computing Density of Reachable States for Safe Autonomous Motion Planning." In: *NASA Formal Methods Symposium*. 2022, pp. 251–271.

[10] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein. "The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections." In: *IEEE Intelligent Vehicles Symposium*. 2020, pp. 1929–1934.

[11] J.-C. Latombe. *Robot motion planning*. Vol. 124. Springer Science & Business Media, 2012.

[12] K. Lee and D. Kum. "Collision avoidance/mitigation system: Motion planning of autonomous vehicle via predictive occupancy map." In: *IEEE Access* 7 (2019), pp. 52846–52857.

[13] G. Francis, L. Ott, and F. Ramos. "Stochastic functional gradient for motion planning in continuous occupancy maps." In: *IEEE International Conference on Robotics and Automation*. 2017, pp. 3778–3785.

[14] K Macek, I. Petrovic, and E. Ivanjko. "An approach to motion planning of indoor mobile robots." In: *IEEE International Conference on Industrial Technology*. 2003, pp. 969–973.

[15] S. Hacohen, S. Shoval, and N. Shvalb. "Probability navigation function for stochastic static environments." In: *International Journal of Control, Automation and Systems* 17.8 (2019), pp. 2097–2113.

[16] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How. "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns." In: *Autonomous Robots* 35.1 (2013), pp. 51–76.

[17] Z. Huang, W. Schwarting, A. Pierson, H. Guo, M. Ang, and D. Rus. "Safe Path Planning with Multi-Model Risk Level Sets." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2020, pp. 6268–6275.

[18] Y. Chen, T. T. Georgiou, and M. Pavon. "Optimal steering of a linear stochastic system to a final probability distribution, Part I." In: *IEEE Transactions on Automatic Control* 61.5 (2015), pp. 1158–1169.

[19] K. Okamoto and P. Tsiotras. "Optimal stochastic vehicle path planning using covariance steering." In: *IEEE Robotics and Automation Letters* 4.3 (2019), pp. 2276–2281.

[20] J. Ridderhof, K. Okamoto, and P. Tsiotras. "Nonlinear uncertainty control with iterative covariance steering." In: *IEEE Conference on Decision and Control*. 2019, pp. 3484–3490.

[21] Y. Chen. "Covariance Steering for Nonlinear Control-affine Systems." In: *arXiv preprint 2108.09530* (2021).

[22] Z. Yi, Z. Cao, E. Theodorou, and Y. Chen. "Nonlinear covariance control via differential dynamic programming." In: *American Control Conference*. 2020, pp. 3571–3576.

[23] V. Krishnan and S. Martínez. "Distributed optimal transport for the deployment of swarms." In: *IEEE Conference on Decision and Control*. 2018, pp. 4583–4588.

[24] K. F. Caluya and A. Halder. "Finite horizon density steering for multi-input state feedback linearizable systems." In: *American Control Conference*. 2020, pp. 3577–3582.

[25] K. F. Caluya and A. Halder. "Reflected Schrödinger bridge: Density control with path constraints." In: *American Control Conference*. 2021, pp. 1137–1142.

[26] Y. Chen and A. D. Ames. "Duality between density function and value function with applications in constrained optimal control and Markov Decision Process." In: *arXiv preprint 1902.09583* (2019).

[27] Y. Meng, D. Sun, Z. Qiu, M. T. B. Waez, and C. Fan. "Learning density distribution of reachable states for autonomous systems." In: *Conference on Robot Learning*. 2022, pp. 124–136.

[28] I. R. Manchester and J.-J. E. Slotine. "Control Contraction Metrics: Convex and Intrinsic Criteria for

Nonlinear Feedback Design." In: *IEEE Transactions on Automatic Control* 62.6 (2017), pp. 3046–3053.

[29] D. Sun, S. Jha, and C. Fan. "Learning Certified Control using Contraction Metric." In: *Conference on Robot Learning*. 2020.

[30] S. Hoermann, M. Bach, and K. Dietmayer. "Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling." In: *IEEE International Conference on Robotics and Automation*. 2018, pp. 2056–2063.

[31] J. Wu, J. Ruenz, and M. Althoff. "Probabilistic map-based pedestrian motion prediction taking traffic participants into consideration." In: *IEEE Intelligent Vehicles Symposium*. 2018, pp. 1285–1292.

[32] M. Koschi, C. Pek, M. Beikirch, and M. Althoff. "Set-based prediction of pedestrians in urban environments considering formalized traffic rules." In: *IEEE International Conference on Intelligent Transportation Systems*. 2018, pp. 2704–2711.

[33] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization." In: *International Conference on Learning Representations*. 2015.

[34] L. Dubins. "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents." In: *American Journal of Mathematics* 79.3 (1957), pp. 179–202.

[35] G. Terejanu, P. Singla, T. Singh, and P. D. Scott. "Uncertainty Propagation for Nonlinear Dynamic Systems Using Gaussian Mixture Models." In: *Journal of Guidance, Control and Dynamics* 31.6 (2008), pp. 1623–1633.

[36] Y. Xu, H. Zhang, Y. Li, K. Zhou, Q. Liu, and J. Kurths. "Solving Fokker-Planck equation using deep learning." In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 30.1 (2020).