2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Risk-Aware Neural Navigation From BEV Input for Interactive Driving

Suzanna Jiwani[1], Xiao Li[1], Sertac Karaman[2] and Daniela Rus[1]

***Abstract*— Safety has been a key goal for autonomous driving since its inception, and we believe recognizing and responding to risk is a key component of safety. In this work, we aim to answer the question, "How can explainable risk representations be generated and used to produce risk-averse trajectories?" To answer this question, previous work uses risk metrics to formulate an optimization problem. In contrast, our work is based on research showing the usefulness of grids as a representation to generate image-based risk maps through a trained neural network. We propose a method of determining risk from a bird's eye view (BEV) of an autonomous vehicle's surroundings. Our method consists of (1) a *risk map generator*, which is trained to recognize risk associated with nearby agents and the map, (2) *differentiable value iteration* using the risk map to learn a policy, and (3) a *trajectory sampler*, which samples from this policy to generate a trajectory. We evaluate our planner in a close-loop manner and find improvements in its overall ability to mimic human driving while maintaining comparable safety statistics. Self-ablation also reveals the potential for fine-tuning the behavior of the planner given a designer's needs.**

I. INTRODUCTION

Autonomous driving has been the next big breakthrough in the transportation industry for at least a decade. Safety, which is defined by collision avoidance, is a requirement but by no means a sufficient condition for enabling the widespread adoption of self-driving cars. Humans drive in a risk-aware manner, meaning we are anticipating of our surroundings (static obstacles and moving traffic agents) and are responsive well before collisions happen. While it is clear in concept what collision avoidance means (e.g. geometric areas that should not be visited), risk is a more vague term and can be difficult to specify. In this work, we aim to address the following question: *"can we develop a planner that is (1) able to generate rich and explainable risk representations; (2) able to produce trajectories that are averse to the generated risk; and (3) trainable end-to-end using expert demonstrations?"*. In the remainder of the paper, we refer to *ego* vehicle as one that is controlled by our planner, and all others as *ado* vehicles.

Risk-aware planning and policy learning has been looked at in the context of traditional risk metrics such as mean risk, value at risk, etc [1], [2]. Such risk metrics are used to formulate a receding horizon optimization problem (as objective functions or constraints). Risk defined this way is restrictive in the obtainable high-level environmental semantics (e.g. vehicle motion states, map definition) and how they are combined in the risk formulation (often designed manually). We aim to address this problem by leveraging a data-driven approach where the risk map is generated from a top-down view image of the ego-centric surroundings, which is then used for trajectory planning (Section II provides details on contrasting with current literature).

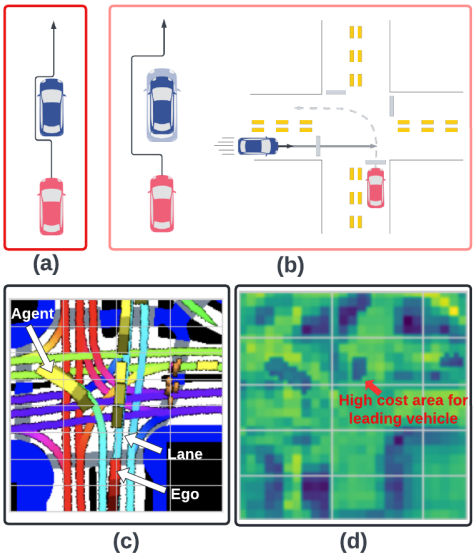

Fig. 1 : Illustration of safety vs. risk. The components of risk are shown in (a) and (b), with the red car representing the ego, and the blue leading ado agent on the road. An example BEV and corresponding risk map are shown in (c) and (d).

In this paper, we make the distinction between safety and risk as depicted in Figure 1. In an overtaking scenario, Figure 1 (a) shows that the ego vehicle (red) can travel very close to the leading ado vehicle (blue) and still avoid collision. This ensures safety but is obviously not ideal. Our notion of risk involves dealing with the uncertainty in the positioning of other vehicles (Figure 1 (b) left) and the anticipation of other vehicle's future motion (Figure 1 (b) right).

Accurately formulating both collision avoidance (static and dynamic obstacles) and rich risk representation as an optimization problem while also striking a balance between inference speed and situation coverage can be challenging. This is especially true if taking the model-based optimization motion planning route (e.g. model predictive control). Instead, we make use of the birds-eye view (BEV) representation, which has recently gained popularity as the intermediate rich representation between perception and planning [3]. It incorporates the map as well as traffic agents' information in a semantically colored image (similar to an occupancy grid map) which can be consumed by downstream planners.

Toyota Research Institute provided funds to support this work.

[1] Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology {sjiwani, xiaoli, rus}@mit.edu

[2] Laboratory for Information and Decision Systems, Massachusetts Institute of Technology sertac@mit.edu

Our approach is to design a differentiable architecture that takes a BEV image as input (Figure 1 (c)), learns to generate a semantically meaningful and compact risk map (Figure 1 (d)) from human driving data, and outputs a risk-aware trajectory plan. To summarize our contributions, we:

- propose a neural navigation architecture that learns to generate interpretable risk maps and risk-averse trajectory plans from human driving data;
- show that the proposed neural planner is able to incorporate users' risk preferences;
- evaluate on a data-driven simulation environment and show improved explainability, safety, and flexibility.

II. RELATED WORK

Risk-aware planning and policy learning. The most straightforward implementations of risk in the field of vehicle trajectory planning either hardcode values to indicate the importance of avoiding specific types of agents [4] or simplify/abstract the notion of risk to a metric for collision likelihood [5]. The authors of [1] provided a comprehensive survey on this topic in the context of reinforcement learning (RL) solutions. In [2], the authors dive into similar topics with a focus on autonomous driving. In general, common risk metrics such as conditional value at risk, mean-variance, worst-case analysis, etc. have been used as either an auxiliary loss in the objective function or as a set of constraints. In many robotic and driving tasks, however, RL is not readily applicable given its exploration requirement. Imitation learning and offline RL help with addressing this problem. In [6], the authors use kernelized movement primitives to estimate uncertainties in the demonstrations, and the uncertainties are then used to find optimal gains in a controller. The authors of [7] use offline distributional RL to learn a policy adverse to conditional value at risk. Most similar to our work, [8] predicts future motion using an occupancy-map based safety metric. *Compared to previous works that use risk metrics to formulate an optimization problem, our approach of generating image-based risk maps using a trained network provides better flexibility, integration of complex agent and map considerations, and inference speed when outputting the final trajectory. We are also able to learn a distribution shift resilient planner without the need for exploration*.

Trajectory Planning from BEV input. BEV is becoming an increasingly popular intermediate representation that connects perception and prediction/planning components in a deep architecture. In [9], [10], the authors have developed efficient methods to generate BEV images of the ego vehicles surrounding from camera and LiDAR data. In [11], [3], the authors use the BEV images to generate trajectories for prediction. In trajectory planning, the concept of BEV is often realized as occupancy grid maps [12], [13] and grid/lattice-based planners are used on them. Less effort has been made in incorporating BEV in neural architectures for planning. We recognize BEV as a powerful representation that can be used to not only represent the geometric distribution of static and dynamic objects (e.g. map and traffic), but can also be used to represent the ego agent's understanding of its environment in the form of a risk map, a mapping of positions to ego's level of risk of being in that position. Manually specifying such a risk map to incorporate a wide coverage of scenarios is difficult. Therefore, *our method aims to integrate a risk map generator with a trajectory generator in the same neural architecture and train them using human driving data*. Our model learns a risk map by optimizing for an objective which takes into account the uncertainty of ado vehicles' positions. The most relevant literature to our work which also serves as an inspiration is [14], where the authors learn a reward map from BEV and use it for multi-model trajectory prediction. The difference in our work is that we use additional risk supervision and goal conditioning to make our architecture a "steerable" risk-aware planner (details will be discussed in Section IV).

III. BACKGROUND

A. Max Entropy Inverse Reinforcement Learning

In this section, we briefly introduce the max entropy IRL framework [15], [16], [17] in order to describe how we arrived at D_τ and D_θ, which are used in the training process for the Risk Map Generator, described in more detail in section IV-D. First, we define the discrete state and action space Markov Decision Process (MDP) as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$ where $\mathcal{S}$ is the set of states on a 2D grid. $\mathcal{A} = \{\text{left, right, up, down, end}\}$ is a set of discrete actions on the grid. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a deterministic transition function. $r : \mathcal{S} \to \mathbb{R}$ is the reward function.

Using the MDP definition above and the max entropy principle, the probability of a state action sequence $\tau = \{(s_1, a_1), (s_2, a_2), \dots (s_N, a_N)\}$ is proportional to the reward exponential.

$$P(\tau) = \frac{1}{Z} \exp\left(\sum_{i=1}^{N} r\left(s_i\right)\right) \tag{1}$$

where Z is the normalizing constant. The goal of IRL is to learn the reward function $r_\theta(s)$ parametrized by parameters θ, which maximizes the log-likelihood of demonstrations $\mathcal{T} = \{\tau_1, \tau_2, \dots \tau_K\}$

$$\arg\max_\theta \mathcal{L}_{\text{MDP}} = \arg\max_\theta \sum_{\tau \in \mathcal{T}} \log\left(\frac{1}{Z_\theta} \exp\left(r_\theta(\tau)\right)\right). \tag{2}$$

with the gradient of the log-likelihood $\mathcal{L}_{\text{MDP}}$ as

$$\frac{d\mathcal{L}_{\text{MDP}}}{d\theta} = \sum_{\tau \in \mathcal{T}} \left(D_\tau - D_\theta\right) \frac{dr_\theta}{d\theta} \tag{3}$$

In Equation (3), D_τ are the state visitation frequencies (SVFs) for the demonstrations τ. D_θ are the expected SVFs for the optimal policy given the current set of reward parameters θ. D_τ is obtained by mapping the demonstration trajectories on the 2-D grid, whereas D_θ can be obtained by propagating the policy for N steps for all the states on the grid (starting from the initial state distribution) and summing over time.

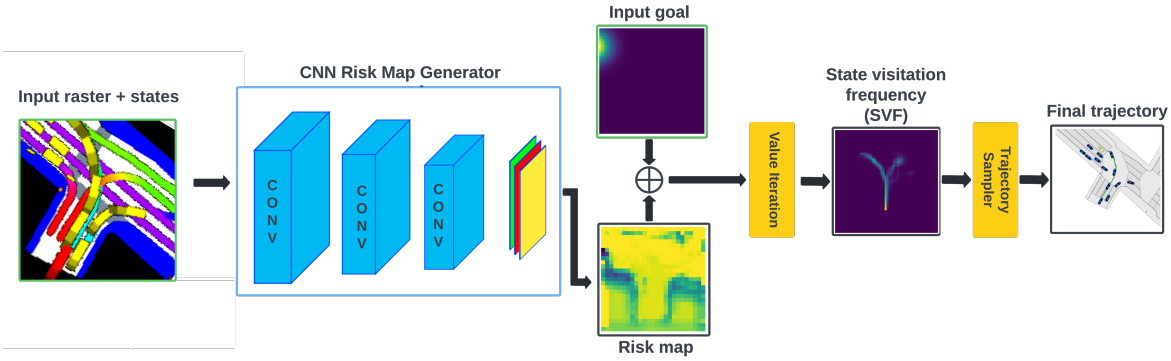

Fig. 2 : Architecture with intermediate outputs. A rasterized BEV and motion features (a 2D vector with 3 channels $(v, \frac{x}{grid\_size}, \frac{y}{grid\_size})$, written as "states") are passed to the CNN Risk Map Generator, which outputs the risk map. This risk map is used alongside a goal costmap input for a soft value iteration module to generate a stochastic policy. The policy is propagated a number of timesteps to obtain the SVF, which in turn is sampled and transformed to obtain the final trajectory.

IV. Risk-Aware Neural Trajectory Generator

The proposed model takes as input a rasterized BEV and motion features. Figure 2 shows how these are passed to a CNN Risk Map Generator to generate a risk map of the scene. The downstream portion of our model takes in both the goal and the generated risk map and perform a soft value iteration to generate a policy for how to traverse the pixels of the risk map. This policy is then sampled and translated to coordinates in the ego frame.

A. Risk Map Generator

The Risk Map Generator is a learned component which maps a risk score to each of the grid states in the risk map (usually a lower dimensional image than the BEV). The raster image is encoded using the stem and first 3 residual blocks of ResNet-34 [18], followed by an additional 2D convolution. This encoding is concatenated with motion features (written as "states" in Figure 2) that is a 2D vector with the following channels: (1) the velocity of the ego v, (2) the x coordinate of the center of the grid state in the local frame divided by the extent of the grid in meters, and (3) the y coordinate of the center of the grid state in the local frame divided by the extent of the grid in meters. This concatenation is then passed through a basic convolutional neural network to produce the risk map.

B. Differentiable Value Iteration

For our value iteration method (shown in Algorithm 1), we take inspiration from [14], with the modification that the goals are now user inputs in the form of a cost map (same size as the risk map). Within the cost map, the goal position occupies a high value whereas all the other positions are low valued. In addition, a Gaussian is applied around the goal position to better guide the agent in the goal vicinity towards the center. Specifically, Algorithm 1 involves solving for the max entropy policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, given the current reward function r_θ, and the goal state s_{goal} (reward function = risk map + goal costmap). π_θ represents the probability of taking action a, given state s. For each step, π_θ is given by

$$\pi_\theta(a \mid s) = \exp\left(Q(s,a) - V(s)\right), \tag{4}$$

where $V(s)$ and $Q(s,a)$ are the soft estimates of the expected sum of future rewards given the current state / state-action respectively.

Algorithm 1 Differentiable Value Iteration

Inputs: risk map $\mathcal{R}$; goal map g
$V_\mathcal{R} \leftarrow -\infty$
$V_g \leftarrow g$
$V \leftarrow [V_\mathcal{R}, V_g]$ ▷ $[\cdot,\cdot]$ is the append operator
for n=1 ...N **do**
  $Q_\mathcal{R}(s,a) = r + \gamma \cdot V_r(s')\ \forall s, a, s' = \mathcal{T}(s,a)$
  if $\mathcal{T}(s,a)$ `out of bounds` **then**
    $Q_\mathcal{R}(s,a) = -\infty$
  end if
  $Q_g(s,a) = -\infty$
  $V_\mathcal{R}(s) = \text{logsumexp}_a Q_\mathcal{R}(s,a) \forall s$
  $Q \leftarrow [Q_\mathcal{R}, Q_g]$
  $\pi(a|s) = \exp(Q(s,a) - V(s))$
end for

C. Trajectory Sampler

Once this policy is generated, we sample probabilistically according to the policy to determine the sequence of grid states. Since this is probabilistic, we sample 1000 times (per future timestep) then cluster the sequences and pick the one with the most elements in the cluster. For each pixel location in the trajectory, we compute the coordinate of the center of the pixel and append that to our transformed trajectory. Since the model is trained to plan 6 seconds into the future, the trajectory is evenly downsampled to 12 waypoints (2 per second as is the frequency of our dataset). This method does not take into account a velocity profile when distributing waypoints throughout the generated plan, but the Risk Map Generator is capable of using the history in the rasterized BEV as well as the ego velocity in the motion features to determine an appropriate distance of travel.

D. Training

The Risk Map Generator is the only learned component of the architecture. Before starting the training process,

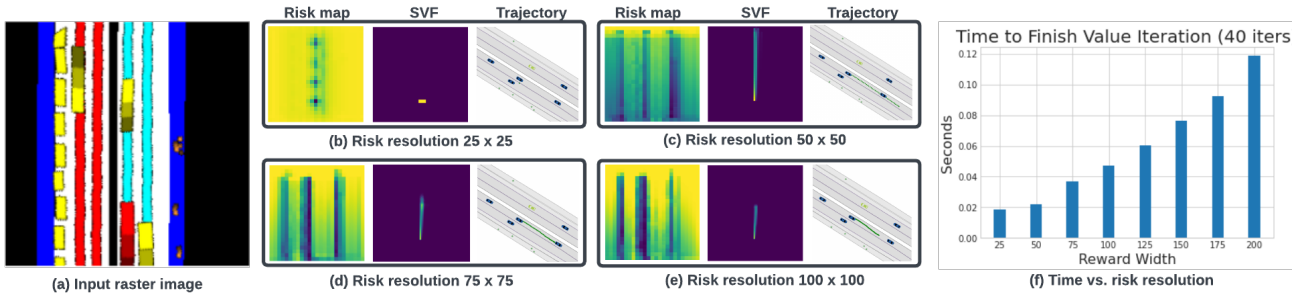

Fig. 3 : Risk map resolution study. For the example scene (a), we examine the intermediate risk map and SVF, as well as the final trajectory, generated by the proposed planner for a variety of risk map resolutions (b)-(e). We observe improved details in the risk map features coupled with more reasonable trajectories as the resolution increases, though (as indicated in (f)), there is an increase in time required to perform value iteration.

labels for D_τ and D_θ must be generated using the process described in section III-A. Then, the Risk Map Generator is trained independently of the trajectory sampling process using gradient descent on a modified version of $\mathcal{L}_{\text{MDP}}$

$$\frac{d\mathcal{L}_\theta}{d\theta} = \sum_{\tau \in \mathcal{T}} \left((D_\tau - D_\theta) + \mathcal{N}_{\text{overlap}} \right) \frac{dr_\theta}{d\theta} \tag{5}$$

with respect to the CNN's parameters, where $D_\tau - D_\theta$ is the difference between the ground truth SVF and the SVF generated by following the policy from value iteration. The inclusion of this term encourages the Risk Map Generator to generate maps which, once passed onto value iteration, will result in SVFs similar to human driving.

For $\frac{d\mathcal{L}_\theta}{d\theta}$, we modify equation (3) by adding the $\mathcal{N}_{\text{overlap}}$ term. $\mathcal{N}_{\text{overlap}} = D_\theta \cdot \upsilon \cdot \mathcal{N}$ is a term which represents the uncertainty of the positions of the agents, where $\mathcal{N}$ is the "uncertainty map" (an image informed by the locations of the ado agents in the scene, where a bivariate Gaussian is centered at each agent, visualized in Figure 6). The operator '$\cdot$' is an element-wise product, and υ is a weight to control the effect of this term. Since non-occupied states in $\mathcal{N}$ have a value of zero, intuitively, this term will only add loss for states on the SVF which are close to, or are occupied by, ado agents, thereby discouraging generating SVFs which overlap with ado agent's uncertainties. Using $\mathcal{N}_{\text{overlap}}$ in the loss as opposed to incorporating uncertainty into inference/inputs allows the CNN to recognize *implicit* uncertainties associated with surrounding agents. In future work, the bivariate Gaussian uncertainty can be replaced with the uncertainty model / characteristics of the sensors.

V. Experiments

Dataset. We use the NuScenes dataset [19] for training and evaluation. The dataset contains 1000 scenes of 20s each collected in Boston and Singapore. It also includes rich semantic information including 23 object classes (pedestrian, vehicle, etc) and HD maps with 11 annotated layers (lanes, walkways, etc).

Implementation Details. We train and test with an observation history of 2 seconds and a prediction horizon of 6 seconds with a frequency of 2 Hz. We use a train batch size of 32 and train for a total of 100 epochs on a cluster using 1 NVidia Tesla V100 GPU.

Method of Evaluation. We evaluate our method and comparison cases in terms of *optimality*, *safety*, and *similarity to human driving*. Within the dataset, we will set the human ego vehicle's start and end positions as the initial and goal positions. Optimality is measured as the time to travel from the initial position to the vicinity of the goal position. Safety is measured as the percentage of encounters with neighboring ado vehicles, across all time steps, where the ego is dangerously close, and similarity to human driving is the L2-norm between the planner and human trajectories (also called average displacement error or ADE for short). During evaluation, we control the ego vehicle with our learned planner, the ado vehicles move according to the trajectories recorded in the dataset with synchronized time. This data-driven simulation approach provides a closed-loop testing environment with realistic ado vehicle movements. All results are averaged over the validation set.

Comparison Cases. Five planner variants are used for comparison. *Ours* refers to the proposed method; *Human* refers the human driver in the dataset; *CNN* refers to a planner that maps BEV directly to controls (Implemented similarly to [20]); and *CNN-LSTM* refers the previous planner with an added LSTM component to keep track of history. *RvS-G* refers to the goal-conditioned offline RL via supervised learning proposed in [21]. For all planners, the same CNN backbone is used to extract features from the rasterized BEV image (similar to [22]). These planners were taken from their respective authors' source code with a modification only to roll out trajectory plans for a fairer comparison.

TABLE I: Performance Comparisons

Model	Safety (%)		ADE (m)		Goal (m)	
	mean	90%	mean	90%	mean	90%
Human	19%	51%	n/a	n/a	n/a	n/a
CNN	**10%**	**30%**	23.22	43.54	16.2	55.7
CNN-LSTM	13.2%	40%	18.7	39.6	18.1	60.4
RvS-G	25%	67%	16.15	31.16	32.91	72.55
Ours	12.16%	31%	**2.17**	**3.81**	**1.99**	**3.68**

Results And Discussion. We start by comparing our planner against the four other planner variants across the three key metrics of safety - average displacement error (ADE), and goal achievement - as shown in table I. We observe that our planner outperforms the comparisons in its

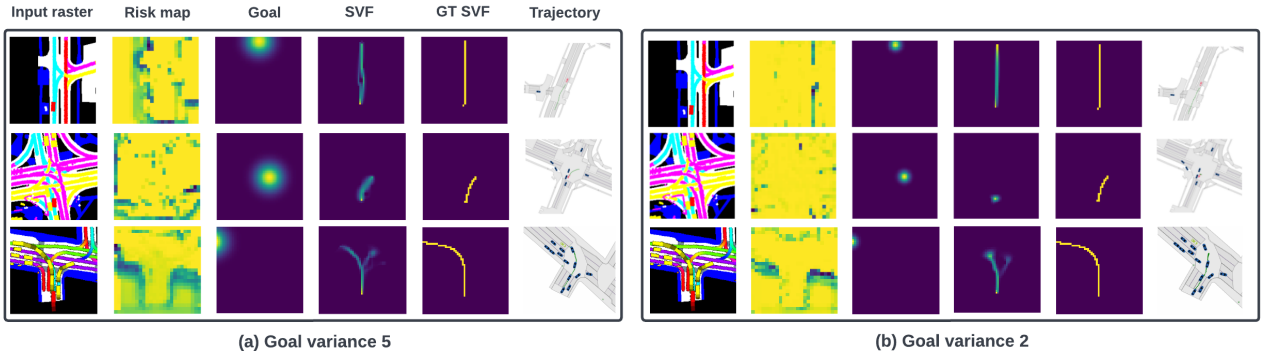

Fig. 4 : Goal specification study. For three sample scenes - a straight trajectory, a slow turn, and a large turn - we examine the intermediary outputs as well as the final trajectory generated by two planner variants - (a) one trained with a goal variance of 5 and (b) another with a goal variance of 2. We observe the smaller variance results in straight trajectories with better goal achievement but also makes turning more difficult.

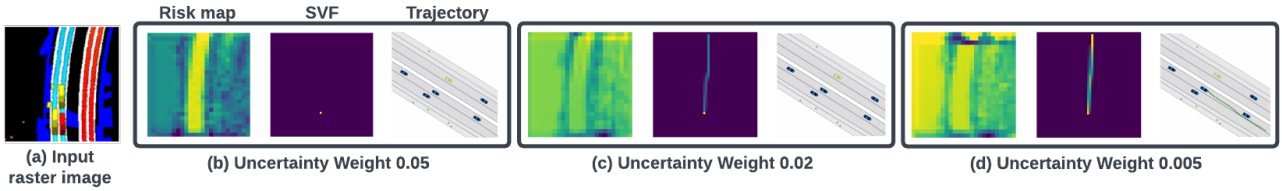

Fig. 5 : Uncertainty weight ablations. For the example scene presented in (a), we examine the intermediate risk map and SVF, as well as the final trajectory, generated by the proposed planner for a variety of uncertainty weights. We observe planners trained with higher uncertainty weighting generate more conservative trajectories in response to nearby agents. (a) Input BEV with a mostly straight trajectory where there are agents directly in front and to the left of the ego. (b) Highest uncertainty weighting, generates an SVF with little/no movement. (c) Medium uncertainty weighting, with an SVF that indicates a possibility for forward movement. (d) Low uncertainty weighting, generates an SVF with a clear forward trajectory.

ability to mimic human driving and reach the ground truth goal while still retaining comparable safety scores as the safest planner (CNN). There are a few possible reasons for this. One is that these comparisons are done with a 50x50 resolution grid, which may cause the trajectory mapping to an SVF less precise. There are also a number of other tuneable parameters that can affect our planner's performance, which will be discussed in depth in the rest of this section.

Higher risk map resolution yields better trajectories at the cost of longer inference time. We investigate the effect of risk map resolution on the generated trajectories. Intuitively, we expect a higher resolution to allow for more reasonable movements. Figure 3 presents a case study for how the model performs on a single scene with 4 different risk map resolutions. With the smallest resolution (25x25), the risk map appears to prioritize leading vehicle avoidance (with anticipation of future motion), without the ability to distinguish between the two lanes on the correct side of the highway. This results in an SVF where the ego vehicle simply stays in place. At the next resolution (50x50), lane dividers start to emerge through the risk map, with going off-road clearly discouraged. As a result, the ego vehicle moves forward, though without a clear stopping point. The next two larger resolutions (75x75 and 100x100) show risk maps that are visually similar to the 50x50 resolution, with more defined features (larger negative values outside of road boundaries and ado agent possible futures). They are therefore able to take into account the ado agents ahead of the ego and modify the distance of the forward trajectory. However, this increased ability to specify lanes, dividers, agents, and other obstacles comes at a cost. Figure 3 (f) indicates an increase in the time required to perform value iteration, making higher resolutions infeasible for real-time inference. We conclude that increasing the resolution of the generated risk maps allows for more detailed traffic features, which improves human driving similarity, though value iteration takes longer.

Larger goal radius allows for more flexibility. Next, we examine the effects of adjusting the relative size of the conditioned goal input across a variety of representative scenes. Figure 4 presents three scenes - a straight trajectory in the first row, a slow turn in the second row, and a larger turn in the last row - each with a Gaussian goal with a variance of 5 and another with a variance of 2. Across all three scenes, it is evident that the model with a larger variance generates risk maps with more details. We observe that using a bivariate Gaussian distribution with high variances as the goal results in flexible path generations. We hypothesize that the larger area of the goal within the state grid causes the model to focus on incorporating scene features along a path rather than getting to the goal. This is evident in the straight trajectory (Figure 4 (a) top row), where the generated SVF shows the model considering two different ways of going up the straight road. We also note that the higher variance results in a trajectory that falls short of the ground truth, likely because enough reward can be achieved from being within a larger vicinity of the goal. However, this effect can actually be reversed in a turning scenario, shown in the second row. The decreased flexibility

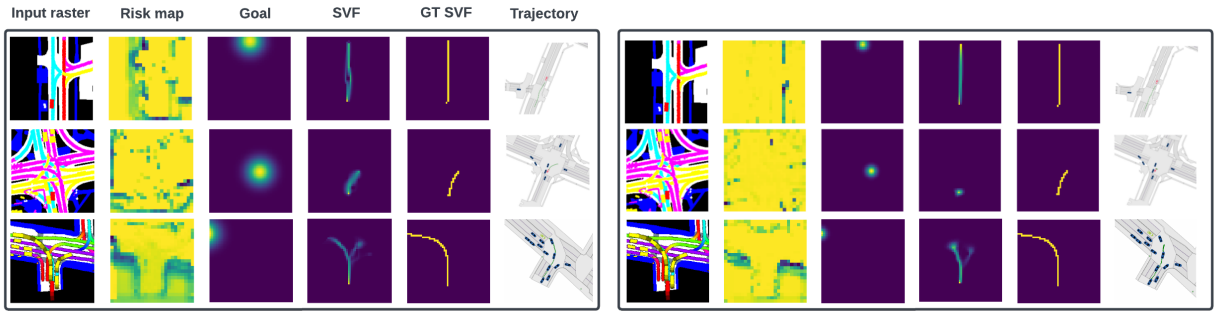

Fig. 6 : Agent uncertainty shape study For this example scene, we examine the intermediate risk map and SVF, as well as the final trajectory, generated by the proposed planner for a variety of agent uncertainty shapes. We observe planners trained with slightly different shapes will respond by generating trajectories that match those shapes. (a) A right turn at an intersection. (b) No uncertainty component in the loss, results in a smooth turn. (c) Small dots as the uncertainty, results in a sharper turn with a notch as the model avoids the agent nearby. (d) Car-sized ovals as the uncertainty, results in an SVF with a more pronounced notch. (e) Large circles as the uncertainty, results in the most pronounced avoidance.

of a smaller variance resulted in a trajectory where the ego stayed in place, whereas the larger variance encouraged movement. With the larger turn represented in the third row, we observe a shorter turn with the higher variance. Overall, while a smaller goal variance may encourage more closely matching the goal in straight trajectory scenarios, the decreased flexibility associated with the smaller goal area becomes detrimental in turning situations.

Larger uncertainty weighting results in a more conservative planner. We examine the ability of the proposed model to adjust to desired risk-awareness by observing its behavior in Figure 5 , where the results of three different training regimens on a single scene are shown. Each training regimen differs only by the uncertainty weight υ of $\mathcal{N}_{overlap}$ in Equation 5, meant to represent the desired risk-awareness. The scene in Figure 5 was chosen to highlight the model's response to nearby agents that directly interfere with movement. We observe that with the largest uncertainty weighting (Figure 5 (b)), both the SVF and the resulting trajectory indicate the ego does not move at all. With the second largest uncertainty weighting (Figure 5 (c)), the SVF shows with low probability the ego moves forward, although the sampled trajectory under such probability results in little movement. Finally, with the smallest uncertainty weighting (Figure 5 (d)), both the SVF and the trajectory indicate the ego vehicle will move forward. Overall, we see that a larger uncertainty weighting results in more conservative driving.

The planner responds to a change in ado vehicle's uncertainty shape. Lastly, we investigate the results of choosing a particular representation for uncertainty masks around ado agents over other representations. As discussed in section IV-D, the proposed model utilizes uncertainty masks such that a bivariate Gaussian with $\sigma_x = 0.5$, $\sigma_y = 1.5$ surrounds each ado agent. Qualitative results of varying the σ's can be seen in Figure 6 . With no uncertainty overlap component included in the loss (Figure 6 (b)), the planner outputs a relatively smooth turning trajectory, with a consistent turn radius. However, once an uncertainty component is introduced to the loss, no matter how the bivariate Gaussians around each agent are shaped, the turn becomes uneven as the planner attempts to avoid the agent in the next lane. With the smallest uncertainty representations ($\sigma_x = 0.5$, $\sigma_y = 0.5$), the planner outputs an SVF that avoids the nearby agent but allows for some flexibility for when it moves into the desired lane. The planner with the largest uncertainty representations ($\sigma_x = 2$, $\sigma_y = 2$) enforces moving into the desired lane as late as possible. Interestingly, the planner with the moderate oval-shaped uncertainty representation ($\sigma_x = 0.5$, $\sigma_y = 1.5$) not only generates a trajectory that avoids the agent with a different path, but the SVF shows a slight possibility of a smooth turn. Overall, this indicates that with our planner, the user has the flexibility to specify the ego agent's risk-seeking behavior.

Limitations. For much of the analysis presented in this paper, we chose a risk dimension of 50x50 pixels. In a number of the planned trajectories, we found that the trajectories were unsmooth. We attribute this to our choice of the trajectory sampler which simply traverses through pixels according to the policy. While this could be improved with a learned generator, we also observed larger risk map resolutions yielded much smoother trajectories. This is likely due to the decreased distance in meters between pixels, meaning simply moving between a column of pixels is a less abrupt motion. However, larger resolutions take more time to generate trajectories so more work should be conducted to explore using this approach within the constraints of a real car.

VI. Conclusions

Overall, we see great potential for the proposed architecture with its ability to react to minor changes in the given goal and uncertainty masks. Designers can control the planner's behavior without additional training - enforcing smaller goal variances when goal achievement is a priority or allowing for larger goal variances when flexibility and interactivity are more important. We also observe the potential to tune how conservative the planner is by adjusting the uncertainty weighting in the loss, as well as the ability to adapt to a variety of sensor infrastructures with a range of localization abilities.

REFERENCES

[1] J. Garcia and F. Fernández, "A comprehensive survey on safe reinforcement learning," JMLR, vol. 16, pp. 1437–1480, 2015.

[2] Z. yu Zhu and H. Zhao, "A survey of deep rl and il for autonomous driving policy learning," ArXiv, vol. abs/2101.01993, 2021.

[3] A. Hu, Z. Murez, N. Mohan, S. Dudas, J. Hawke, V. Badrinarayanan, R. Cipolla, and A. Kendall, "Fiery: Future instance prediction in bird's-eye view from surround monocular cameras," ArXiv, vol. abs/2104.10490, 2021.

[4] Q. Wang and M. Gerdts, "Risk-based path planning for autonomous vehicles," 2021.

[5] K. Mokhtari and A. R. Wagner, "Don't get yourself into trouble! risk-aware decision-making for autonomous vehicles," 2021. [Online]. Available: https://arxiv.org/abs/2106.04625

[6] J. Silvério, Y. Huang, F. J. Abu-Dakka, L. Rozo, and D. Caldwell, "Uncertainty-aware imitation learning using kernelized movement primitives," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 90–97, 2019.

[7] N. A. Urp'i, S. Curi, and A. Krause, "Risk-averse offline reinforcement learning," ArXiv, vol. abs/2102.05371, 2021.

[8] X. Ren, T. Yang, E. L. Li, A. Alahi, and Q. Chen, "Safety-aware motion prediction with unseen vehicles for autonomous driving," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 15 711–15 720, 2021.

[9] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus, and S. Han, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," ArXiv, vol. abs/2205.13542, 2022.

[10] X. Chen, T. Zhang, Y. Wang, Y. Wang, and H. Zhao, "Futr3d: A unified sensor fusion framework for 3d detection," ArXiv, vol. abs/2203.10642, 2022.

[11] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," ICRA, 2019.

[12] K. Lee and D. Kum, "Collision avoidance/mitigation system: Motion planning of autonomous vehicle via predictive occupancy map," IEEE Access, vol. 7, pp. 52 846–52 857, 2019.

[13] A. Loukkal, Y. Grandvalet, T. Drummond, and Y. Li, "Driving among flatmobiles: Bird-eye-view occupancy grids from a monocular camera for holistic trajectory planning," 2021 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 51–60, 2021.

[14] N. Deo and M. M. Trivedi, "Trajectory forecasts in unknown environments conditioned on grid-based plans," CoRR, vol. abs/2001.00735, 2020. [Online]. Available: http://arxiv.org/abs/2001.00735

[15] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in AAAI, 2008.

[16] A. Snoswell, S. P. N. Singh, and N. Ye, "Revisiting maximum entropy inverse reinforcement learning: New perspectives and algorithms," 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 241–249, 2020.

[17] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," arXiv: Learning, 2015.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CoRR, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[19] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," ArXiv, vol. abs/1903.11027, 2019.

[20] W. Farag and Z. Saleh, "Behavior cloning for autonomous driving using convolutional neural networks," 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), pp. 1–7, 2018.

[21] S. Emmons, B. Eysenbach, I. Kostrikov, and S. Levine, "Rvs: What is essential for offline rl via supervised learning?" arXiv preprint arXiv:2112.10751, 2021.

[22] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," ICRA, 2019.