Risk-Aware Neural Navigation for Interactive Driving

by

Suzanna Jiwani

SB, Computer Science and Engineering,
Massachusetts Institute of Technology (2022)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2023

Author .....
Department of Electrical Engineering and Computer Science
September 12, 2022

Certified by .....
Daniela Rus
Andrew (1956) and Erna Viterbi Professor, CSAIL Director
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Risk-Aware Neural Navigation for Interactive Driving

by

Suzanna Jiwani

Submitted to the Department of Electrical Engineering and Computer Science
on September 12, 2022, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

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 used to produce accurate and safe 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 novel 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 using a modified loss to encourage recognizing risk associated with nearby agents, (2) 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 uniquely evaluate our planner in an interactive manner, adjusting the surroundings at each time step, and find significant improvements in its overall ability to mimic human driving, with an 86.56% decrease in average displacement error and an 87.72% decrease in the average distance from the goal while maintaining comparable safety statistics when compared with baseline methods. Self ablation also reveals the potential for fine-tuning the behavior of the planner given a designer's needs.

Thesis Supervisor: Daniela Rus
Title: Andrew (1956) and Erna Viterbi Professor, CSAIL Director

Acknowledgments

I owe everything I have achieved so far in life to my wonderful family, especially my mom, who has supported me and pushed me to do the things I love in both life and academia. Because of you, I could be confident in myself and my choice to attend MIT. Thank you so much for celebrating my achievements, both big and small, and for making sure I always have a place to call home. Also, thank you Junior for checking in on me and my code all summer long.

I would also love to thank my partner Arkadiusz for all the time he's sacrificed, helping me prepare for presentations, talking through ideas, and even debugging code with me. There is absolutely no way I would have been able to do as much as I have without your help; this thesis is as much yours as it is my own.

Importantly, I want to thank Daniela Rus for her support and for providing me with so many resources. Lastly, I owe a huge thank you to Xiao Li for all of his guidance, advice, and mentorship. Our Friday meetings truly have been indispensable - not only did I enjoy our life updates, but each and every one helped keep me on track and encouraged me. I honestly was not sure that I would be able to finish my thesis, or even write a true research paper, but your advice (and experience in the field) really made these goals concrete and attainable.

Thank you all so much.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Self-driving cars, or more broadly autonomous vehicles (AVs), not only have the potential to help many everyday people get around safely, but they are steadily becoming more of a reality. Research in the area, including heavy investments from companies Google, Uber, and Tesla, is resulting in rapid progress. To contribute to the body of work surrounding AVs, **the goal of this work is to develop a planner that is (1) able to generate rich and explainable risk representations; (2) produce trajectories that are averse to the generated risk; and (3) trainable end-to-end using expert demonstrations.**

1.1.1 Autonomous Vehicles

At the center of the development of AVs is the key problem of developing an autonomous system capable of understanding complex environments and appropriately responding to them. Specifically in the context of short-term trajectory planning for self-driving cars (ie, planning the trajectory for the car over the next 3-5 seconds), there are many factors, including lane dividers, other car's speeds, traffic signals, other car's intended maneuvers, pedestrians, etc that determine a car's trajectory. For an autonomous car to be successful and reactive, it needs to be aware of its sur-

roundings. In practice, this is done by attaching some sort of sensing unit to the vehicle (typically LiDAR or an array of cameras) and applying specialized algorithms capable of extracting valuable information from these inputs.

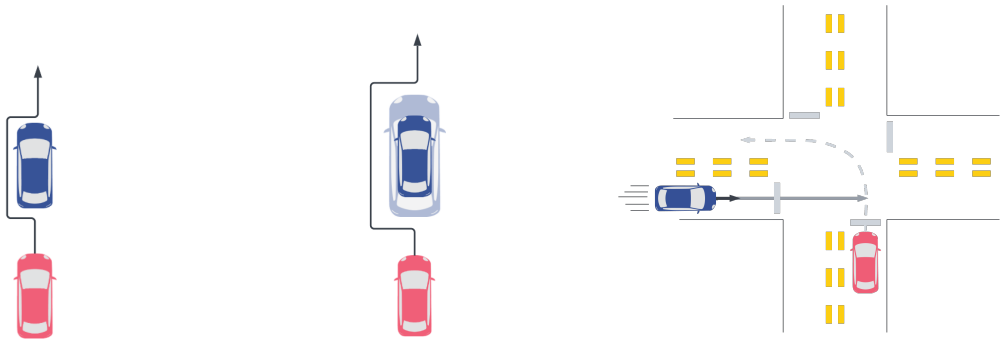
When deciding which information is important for an AV to be successful, the most intuitive approach is to simply draw inspiration from humans. However, hand-tuned algorithms that mimic human behavior are too rule-based and cannot learn from additional data encountered. To bridge this gap between human behavior and implemented policy, we can utilize machine learning models that are capable of finding complex patterns and train them to replicate humans. The ability to be aware of other vehicles on the road is an important factor that makes humans successful drivers. Using their own experience and sensing systems, humans are able to avoid other drivers on the road, even though they cannot be completely certain what behavior others will exhibit in the next few seconds. Humans implicitly assign risk to each of these obstacles on the road - a road barrier certainly will not move, but another driver may choose to behave unexpectedly.

1.1.2 Risk

While humans almost always implicitly think about risk when driving, it can be difficult to determine an appropriate metric of risk from an algorithmic standpoint. Currently, many papers in the self-driving field use the term "risk" to mean any number of things, typically collision risk. Therefore, it is important to clearly define the metric of risk as used in this thesis.

A visualization for the key components of risk used in this work can be found in Figure 1-1, which shows a few interactions between the vehicle making decisions (ego) and the other vehicles on the road (agents), and highlights the difference between safety, or collision avoidance, and the two components of risk.

The first component, seen in Figure 1-1b, is derived from the inherent uncertainty in where other agents are on the road. Since self-driving cars rely on sensors to determine where objects are around them, they cannot be sure what the ground truth of their positions is. The hardware used is only rated to a certain degree of accuracy, as



(a) Safety (b) Uncertainty-based risk (c) Prediction-based risk

Figure 1-1: **Illustrating risk compared to safety.** The red car represents the ego, while the blue is another agent on the road. While in subfigure 1-1a, the ego avoids a collision with the agent, the components of risk are shown in 1-1b and 1-1c. With uncertainty-based risk, the ego recognizes the positional uncertainty of the agent and takes care to avoid all risky areas. With prediction-based risk, the ego uses the past trajectory of the agent to avoid risky areas where the agent may be in the future.

are the software components that transform the measurements coming from hardware into (x, y) positions of agents, and this potential inaccuracy compounds as information is passed around. While most AV systems can be fairly sure about the positions of objects relative to the ego, there is always some small level of uncertainty, which is represented in this component of risk.

The second component of risk, seen in Figure 1-1c, represents the future positions of agents rather than the current. This is largely dictated by the agent's driving over the last 1-3 sec and is driven by the question, "Where can I expect these agents to be in the next 1-3 sec?" Just as a human driver assesses the cars around them to determine whether to expect them to move slowly, quickly, in a straight line, or turn, so should an AV.

However, it is unrealistic to assume a self-driving car will have immediate access to a clear value for the risk of other agents, so we will attempt to predict those relative risk values from birds-eye views (BEVs) of the scene. To do so, we develop a machine learning model that, from BEVs, will gather information about the scene and the riskiness of the other objects on and around the road. Using this information, it will then create a trajectory for the ego vehicle to follow for the next 3-5 seconds.

Ultimately, the goal of engineers in the field is to create self-driving technology

that is safe for the people using it. Determining the risks of agents on the road can help quantify situations that ML models should avoid, leading to overall safer AVs.

1.2 Contributions

We propose a novel approach to determining risk, where a trained convolutional neural network is able to identify a rich grid-based risk map without explicit identification of uncertainty during inference. Instead, the network is exposed to generated uncertainty maps during training so that it can learn to incorporate the measurement uncertainty of agent positions into its risk map generations. The proposed architecture is then able to utilize this risk information to couple seamlessly with a specified goal so that it can generate "plans" (potential trajectories for the ego to follow), discussed in more detail in Chapter 4. With this planner, we observe an 86.56% decrease in average displacement error and an 87.72% decrease in the average distance from the goal when compared to a variety of baseline models, while maintaining comparable safety (percent close encounters). In addition, we provide an in-depth examination of how our method's hyperparameters, like the resolution of risk maps, can be tuned to alter planner behavior in desired ways. We find that we are able to tune how conservative, precise, or goal-oriented the planner is. A more detailed analysis can be found in Chapter 5.

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.

1.3 Outline

Chapter 2 will discuss other work in the field of AVs related to risk. Chapter 3 will outline what data was used in the models presented here and how labels were generated. Chapter 4 will discuss the model used, with sections 4.1, 4.2, 4.3 detailing the architecture of the model, and section 4.4 detailing the training routines used. Results are presented in Chapter 5, and conclusions and future work are included in Chapter 6.

Chapter 2

Related Work

This work unifies two distinct approaches to the task of trajectory planning. First, there are a set of models that are risk-aware and consider the risk of other drivers when making predictions; these are discussed in section 2.1. Next, there are a set of models that are able to predict the ego vehicles' trajectory from bird's eye view (BEV) images directly, which are discussed in section 2.2. To the best of our knowledge, there is no existing published work that plans trajectories from BEVs while also explicitly utilizing the risk of other vehicles as a feature.

2.1 Risk-Aware Models

Risk-based trajectory planners can be decomposed into two major components - (1) the risk utilization, how risk is used to plan trajectories (typically using methods to minimize risk), and (2) the risk definition, the method by which risk is determined.

In terms of utilizing risk, much work has been done on modeling the problem of motion planning as a partially observable Markov decision process (POMDP). POMDPs tend themselves well towards this task since they intrinsically take an AV's uncertainty of the true environment state into account. Therefore, work modeled off POMDPs use POMDP solvers [17] [14] as their method of risk utilization. These models of risk are excellent for managing the risk associated with state uncertainties, though limited in their ability to visualize that risk for humans, which is a key aspect

the proposed approach intends on addressing.

Common non-POMDP approaches optimize over a cost function involving risk (known as reinforcement learning, or RL) [25]. The authors of [11] provided a comprehensive survey on the topic of "Safe Reinforcement Learning", or RL wherein safety is an important factor, and describe how most approaches will either add a safety factor to the cost function being optimized or add guidance to the exploration process. Our method is most similar to the first, adding a safety factor to the cost function, since we perform value iteration to minimize risk (our cost). In [26], 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 [22], the authors use kernelized movement primitives to estimate uncertainties in the demonstrations, the uncertainties are used to find optimal gains in a controller. The authors of [24] use offline distributional RL to learn a policy adverse to conditional value at risk. These methods, like ours, attempt to minimize the level of risk through environment exploration. However, by utilizing a fixed horizon value iteration, our method is able to avoid the typical time limitations of exploration in RL.

There are a wide range of risk definitions represented in literature today. The most straightforward might divide agents into a predefined set of attributes - for example, whether their driving behavior is aggressive, timid, or normal [23]. Others may hardcode values to indicate the importance of avoiding specific types of agents [25] or reduce the notion of risk to simply a metric for collision likelihood [20]. Artificial intelligence has also been utilized to determine risk values, including fuzzy logic models that incorporate in-vehicle data [10].

Unlike prior work that uses common risk metrics to formulate an optimization problem, the proposed model will predict risk values (where "risk" is as defined in section 1-1) across a grid using deep learning. We then learn a policy using these

risk values from maximum entropy inverse reinforcement learning (MaxEnt RL) that determines the probability of moving from one grid location to another. Not explicitly defining risk during inference gives us the advantage of letting the planner learn risk rather than taking the time to define risk in real time. *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.*

These risk-aware models have great potential for improving a car's ability to plan safe trajectories. One such model showed the potential by comparing the results of fully omnipotent Monte Carlo Tree Search (MCTS) planners with MCTS planners that assume the same internal state for all drivers on the road [23].

2.2 Trajectory Planning from Bird's Eye View

BEV is becoming an increasingly popular intermediate representation that connects perception and prediction/planning components in a deep architecture. While in practice, AVs do not have immediate access to a semantically segmented BEV, there has been significant progress in developing efficient methods to generate BEV images of the ego vehicles surrounding from camera and LiDAR data, making BEVs more accessible to algorithms like ours that rely on a segmented BEV as input. In fact, there are methods which are capable of fusing 2D camera information with 3D LiDAR data to create BEVs that draw from an even larger wealth of information than from a single source [18, 3, 15]. End-to-end models (from sensor input to a trajectory) are also possible, and [13] has shown excellent results from camera images alone.

Other existing work has shown the potential for generating trajectories for prediction directly from BEV [5, 21], though they do not provide an explicit input for and visualization of risk, as our model does. In trajectory planning specifically, the concept of BEV is often realized as occupancy grid maps [16, 19] and grid/lattice-based planners are used on them. Less effort has been made in incorporating a semantically

segmented (rasterized) BEV in neural architectures for planning. We recognize that BEV is 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.

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*. The most relevant literature to our work which also serves as an inspiration is [6], 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 4.4).

Chapter 3

Data

3.1 NuScenes

We use the NuScenes dataset [2] for training and evaluation. This dataset follows a schema wherein a 20 second sequence of consecutive frames known as a *scene* is split into *samples* taken with a frequency of 2Hz. Each sample encompasses input data (including LiDAR data and camera images from the 6 on-board cameras), information about the ego car's pose, as well as sample annotations for all of the elements in the sample (ie. pedestrians, traffic cones, other cars) that details information about them. These details can be stacked together to provide labeling for the scene representation in vector form, and for labeling scene representation as a BEV, NuScenes also has segmented BEVs available per sample. 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).

In sum, there are 3 pieces of data passed to the planner: a 200 pixel x 200 pixel BEV, motion features, and goal. While the BEV is available directly from the NuScenes dataset, the others must be manually formed using information provided by NuScenes. The motion features are of shape $(3, grid\_size, grid\_size)$, where $grid\_size$ is the dimension of the risk map, and represent $(v, \frac{x}{grid\_size}, \frac{y}{grid\_size})$ across each grid state. The goal is of shape $(grid\_size, grid\_size)$, and is computed by

taking the ground truth goal (ie. the future location of the human driver) and mapping a Gaussian to the surrounding pixels. All of these pieces of data reflect the same portion (referred to as *patch* in the NuScenes dataset) of the scene, meaning if $grid\_size = 200$, there would be a pixel-to-pixel mapping between them. Therefore, if the ground truth goal is on the edge of the BEV, then the entire Gaussian would not be visible. Look to Figure 3-1 for an example.

(a) Straight Trajectory

(b) Turning Trajectory

Figure 3-1: **A Sample Engineered Goal**. Here, we not only see how the engineered goal maps directly to the rasterized BEV, but also how the engineered goal can be flexibly defined.

3.2 Supplementary Label Creation

In order to aid the training regimen described in section 4.4, there are additional data points that need to be generated so that they can serve as labels. As discussed in more detail in 4.4, the state visitation frequency representing ground truth movement (or SVF_{gt}) is important when determining loss. Like the supplementary data discussed in section 3.1, SVF_{gt} matches the patch location of the rasterized BEV. In order to generate this label, we map the human trajectory's global coordinates to pixels on the rasterized BEV and mark them as visited.

This data also does not explicitly label "risk" as part of the sample annotation. Therefore, we have some potential options for labeling, discussed in section 3.2.1.

3.2.1 Sensor Uncertainty

One drawback of using nuScenes data as described in 3.1, is that uncertainty in particular objects' positions (derived from hardware limitations or upstream model uncertainty) is not provided. Therefore, a primary goal when creating labels for uncertainty was to remain flexible and allow for these uncertainties to be inputted, as they are during self-evaluation in section 5.4.

The first step in creating labels for uncertainty around an object was deciding what shape they should be. NuScenes provides positions for each object, given as a "translation" in the annotation describing the object. This translation gives a single position, even though each object truly occupies more space than a single point. Knowing that, in birds-eye view, a car is more rectangular than circular and a truck is more elongated than a car, it becomes clear that the uncertainty cannot be a fixed shape for all objects; there is a need for variable length and width. This was the driving factor behind the structure used to represent uncertainty.

$$f(x, y) = A \exp\left(-\left(\frac{(x - x_0)^2}{2\sigma_X^2} + \frac{(y - y_0)^2}{2\sigma_Y^2}\right)\right) \tag{3.1}$$

To address these needs, a 2D (or bivariate) Gaussian is a natural choice. Not only does it inherently lend itself to rapidly diminishing values around a center, but it is easy to manipulate to adjust the shape and size of the curve. The results of applying a bivariate Gaussian to agents in a specific scene are shown in Figure 3-2. Equation 3.2.1 gives the value of the 2D Gaussian for any (x, y) pair, where for a particular clockwise angle θ, the values a, b, c are defined as

$$\begin{aligned} a &= \frac{\cos^2\theta}{2\sigma_X^2} + \frac{\sin^2\theta}{2\sigma_Y^2}, \\ b &= -\frac{\sin 2\theta}{4\sigma_X^2} + \frac{\sin 2\theta}{4\sigma_Y^2}, \\ c &= \frac{\sin^2\theta}{2\sigma_X^2} + \frac{\cos^2\theta}{2\sigma_Y^2}, \end{aligned}$$

(a) The pixels where an agent is present in the scene, on a 200x200 grid

(b) Gaussians centered at each agent with $\sigma_x = 0.5$, $\sigma_y = 1.5$

Figure 3-2: **Sample Uncertainty Map**. Subfigure 3-2a shows the exact pixels that are colored yellow by the rasterized BEV, representing the current position of the agents. Subfigure 3-2b clearly shows the accuracy of the method for transforming from global coordinates of each agent to 2D Gaussians.

such that the matrix

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

is positive definite. The angle θ is set to be equal to the orientation of the agent, and σ_x, σ_y are left as a design choice. While there is the potential to adjust the size and shape of the 2D Gaussians surrounding each agent to the type of agent (ie. smaller circles for pedestrians and longer ellipses for trucks), fixed-shaped Gaussians worked relatively well. In fact, section 5.4 shows how a variety of fixed-shape Gaussian masks were used during evaluation to understand how it affects the trajectories generated.

Chapter 4

Architecture

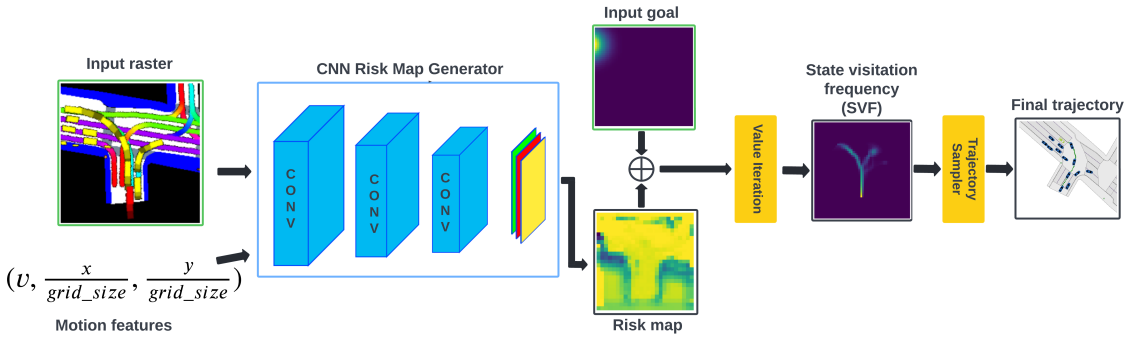


Figure 4-1: **Architecture with intermediate outputs.** Motion features and a rasterized BEV are passed to the CNN Risk Map Generator, which outputs a risk map. This risk map is used alongside an inputted goal during value iteration to generate a policy (visualized here as an SVF), which is sampled and transformed to make the final trajectory.

As discussed in Chapter 3, the proposed model takes as input a rasterized BEV and motion features. Figure 4-1 shows how these are passed to a CNN Risk Map Generator to generate a risk map of the scene. Our model is unique in its ability to utilize a given goal to act as a planner, taking both the goal and the risk map and performing a modified value iteration to generate a policy for how to traverse the pixels of the risk map. This policy is them sampled and translated to coordinates in the ego frame.

Our biggest contribution is our technique to train the Risk Map Generator to take into account uncertainties associated with agents on the road, discussed further in

section 4.4.

4.1 Risk Model

The Risk Map Generator is a learned component which maps a rasterized BEV of a scene and motion features $(v, \frac{x}{grid\_size}, \frac{y}{grid\_size})$ of each grid state to a risk score for each of the grid states (risk map). The raster image is encoded using the stem and first 3 residual blocks of ResNet-34 [12], followed by an additional 2D convolution. This encoding is concatenated with the motion features and passed through a basic convolutional neural network to produce the risk map.

4.2 Value Iteration

Our BEV scene representation has the added benefit that BEV scenes are inherently grid-worlds, meaning that we can model them easily as an MDPs. Every grid cell state $s \in \mathcal{S}$ is observable, has possible actions $\mathcal{A} = \{\text{LEFT}, \text{RIGHT}, \text{UP}, \text{DOWN}, \text{END}\}$ with reward $R(s)$ being the risk map and with deterministic transition probabilities $T(s, a)$ where actions deterministically dictate which cell is moved into next.

This formulation means that classical Value Iteration [1] is a natural choice for finding an optimal policy of ego moving through the BEV grid-world. However, convergence of dynamic programming based value iteration takes a long time, so we choose finite-horizon value iteration, which works well enough empirically (see chapter 5). Moreover, we change the classical formulation by taking inspiration from [6] - such that there is not a single goal state. Instead, the grid-world has two layers - one layer of the grid represents path states and the other layer represents goal states. Ego starts in a path state, and stays there until it chooses the action END, which represents moving from the path grid to the goal grid, indicating episode termination. The path and goal states each have their own reward. In order to incentivize value iteration terminating close to the intended goal, we engineer the rewards for the goal states to be high around the known destination grid cell, smoothed with a Gaussian filter to

allow for more flexibility.

Algorithm 1 displays pseudocode for this modified value iteration. We start by initializing the effective value at each goal state as the value given by the goal mapping g and at each path state as $-\infty$. Combined, these form the variable $V(s)$, representing the expected value of being in a state. For each step until the fixed horizon N, we do the following: (1) update the $Q(s, a)$ value (representing the expected value of taking an action a at state s) for each state-action pair in the path states to be equal to the reward from the current state plus the value accumulated by travelling to the next state multiplied by a discount factor γ, (2) set the Q value for each state-action pair in the goal states to be $-\infty$ (this mathematically makes the policy never move from goal states since $\exp(-\infty) \to 0$), (3) update the value for each state to be an aggregation of taking all actions at that state, (4) update the policy $\Pi(a|s)$ by applying a softmax function over the expected values of taking an action at any given state.

Algorithm 1 Value Iteration

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

4.3 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 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). 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.

4.4 Training

The Risk Map Generator is the only learned component of the architecture. It is trained independently of the trajectory sampling process using gradient descent on the loss $\mathcal{L} = \text{SVF}_{\text{diff}} + \mathcal{N}_{\text{overlap}}$ with respect to the CNN's parameters, where $\text{SVF}_{\text{diff}} = \text{SVF}_{\text{gen}} - \text{SVF}_{\text{gt}}$ is the difference between the ground truth SVF and the SVF generated by following the policy from value iteration.

$\mathcal{N}_{\text{overlap}} = \text{SVF}_{\text{gen}} \cdot \upsilon \cdot \mathcal{N}$ is a novel term which represents the uncertainty of the positions of the agents, where $\mathcal{N}$ is the uncertainty map described in section 3.2.1, $\cdot$ is an elementwise product, and υ is a weight to control the effect of this term. Intuitively, this term adds loss if ego visited states on the grid which are close to, or are occupied by, agents. 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. This model for uncertainty is consistent with measurement uncertainty from sensors, since they are both scene-independent.

Chapter 5

Results

5.1 Method of Evaluation

A key drawback to the model as presented so far is that it plans a trajectory using agent information that will quickly become stale. More specifically, the uncertainty masks used to assess risk utilize agent information to avoid where agents are currently, not where they will be in the future.

In order to simulate using this model in a changing environment, additional infrastructure was created to generate what will be referred to as "rollouts". These rollouts represent trajectories that were generated by the model when the environment is updating at each time step.

In this infrastructure, at each time step, the model outputs a set of trajectories that are each a single step along with their probabilities. The most likely of them is selected, and the scene is simulated to move forward by 1 time step. The ego moves according to the output of the model while all other agents move as they did when the data was collected. Then, the inputs to the model are updated to reflect these changes, and the process begins anew.

We evaluate our method and comparison cases in terms of *optimality*, *safety*, and *similarity to human driving* during these rollouts. 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 close encounters to nearby ado (ie. non-ego) vehicles in a scene, comfort is taken as the maximum acceleration during a scene, and similarity to human driving is the L2-norm between the planner and human trajectories (also called average displacement error or ADE). 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. All results are averaged over the validation set.

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 [9]); 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 [7]. For all planners, the same CNN backbone is used to extract features from the rasterized BEV image (similar to [4]).

Table 5.1: 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**

These comparisons shown in table 5.1. We observe that the planner outperforms the comparisons in its ability to mimic human driving and reach the ground truth goal while still retaining comparable safety scores as the safest planner (CNN). There are also a number of tune-able parameters that can affect our planner's performance, which will be discussed in depth in the rest of this chapter.

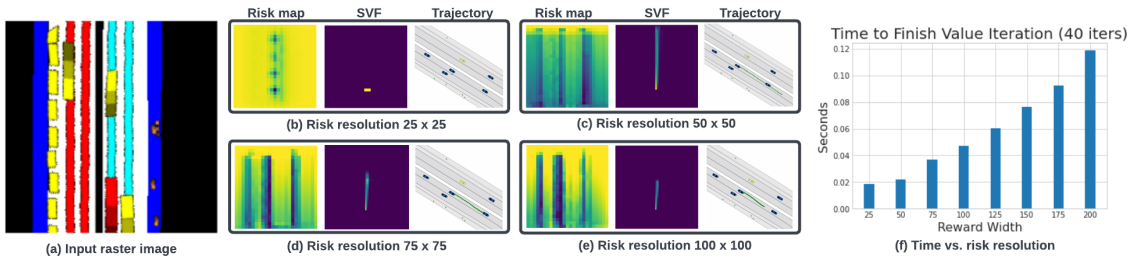


Figure 5-1: **Risk Resolution.** For the sample scene (5-1a), we examine the intermediate risk map and SVF, as well as the final trajectory, generated by the proposed planner for a variety of risk map resolutions (5-1 b-e). We observe more detailed risk maps coupled with more controlled trajectories as the resolution increases, though (as indicated in 5-1 f), there is a distinct increase in time required to perform value iteration.

5.2 Increasing Risk Resolution

First, we investigate the effect of the risk map (also referred to as "reward map" as it pertains to its role in value iteration) dimension on generated trajectories. Intuitively, we expect a greater risk resolution to allow for more precise movements since a small resolution might result in blurry agents and limited capability for distinguishing lanes. Figure 5-1 presents a case study for how the model performs on a single scene with a variety of reward resolutions.

With the smallest resolution (25x25), the risk map appears to prioritize avoiding the lane divider, 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), the two possible lanes are defined through the risk map, with going off-road clearly discouraged. The ego vehicle moves forward, though without a clear stopping point. The next two larger resolutions (75x75 and 100x100) have risk maps that are visually similar to the 50x50 resolution, with more defined features. They are therefore able to take into account the agent 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. The final chart in Figure 5-1 indicates a sharp increase in the time required to perform value iteration on the reward maps, making larger reward

resolutions infeasible for real-time computation.

5.3 Goal Specification

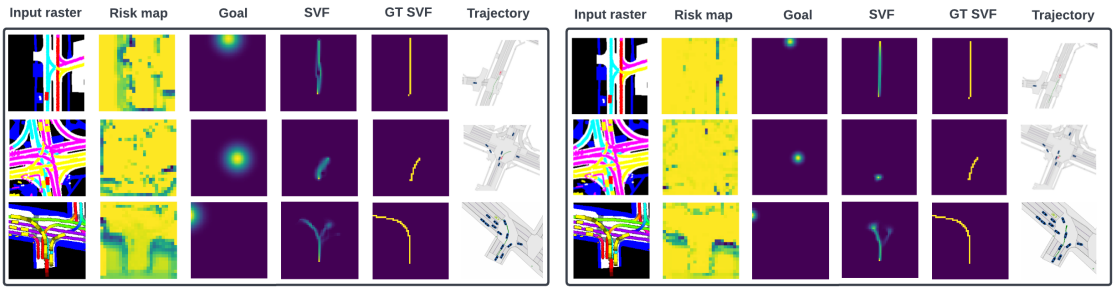


Figure 5-2: **Goal Specification**. For three sample scenes - a straight trajectory, a slow turn, and a large turn - we examine the intermediary outputs as well as the final trajectory generated by two planner variants - one trained with a goal variance of 5 (5-2 a) and another with a goal variance of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

Next, we examine the effects of adjusting the relative size of the inputted goal across a variety of representative scenes. Figure 5-2 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 - for an inputted 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 reward maps with more detail. We observe that using a bivariate Gaussian distribution with high variances as the goal results in flexible path reward 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, 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 standard deviation of the goal. However, this effect can actually be reversed in a turning scenario, shown in the

second row. The decreased flexibility of a smaller variance resulted in a trajectory where the ego stayed in place, whereas the larger variance encouraged movement, even if it fell a bit short of the ground truth. Even with the larger turn represented in the third row, we observe a shorter turn than with a higher variance. Therefore, overall, while a smaller 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.

5.4 Adjusting Uncertainty Specifications

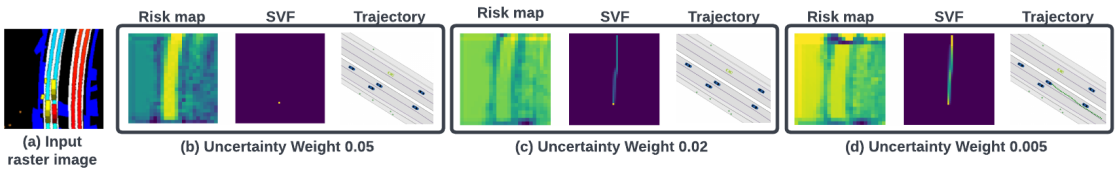


Figure 5-3: **Uncertainty Weight.** For the sample scene (5-3a, a mostly straight trajectory where there are agents directly in front and to the left of the ego), we examine the intermediate risk map and SVF, as well as the final trajectory, generated by the proposed planner for a variety of uncertainty weights (5-3 b-d). We observe planners trained with higher uncertainty weighting generate more conservative trajectories in response to nearby agents.

We also examine the ability of the proposed model to adjust to desired risk-awareness by observing its behavior in Figure 5-3, where the results of three different training regimens on a single scene are shown. Each training regimen differs only by the uncertainty weight of the loss function (discussed in section 4.4), meant to represent desired risk-awareness, and this particular scene 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 5b), both the SVF and the resulting trajectory indicate the ego does not move at all. With the second largest uncertainty weighting (Figure 5c), the SVF shows with some probability the ego moves forward, although the resulting trajectory indicates not moving was more probable. However, with the smallest uncertainty weighting (Figure 5d), both the

SVF and the trajectory indicate the ego vehicle will move forward. Overall, we see that a larger uncertainty weighting will clearly result in a model whose predictions will avoid other road agents more intensely.

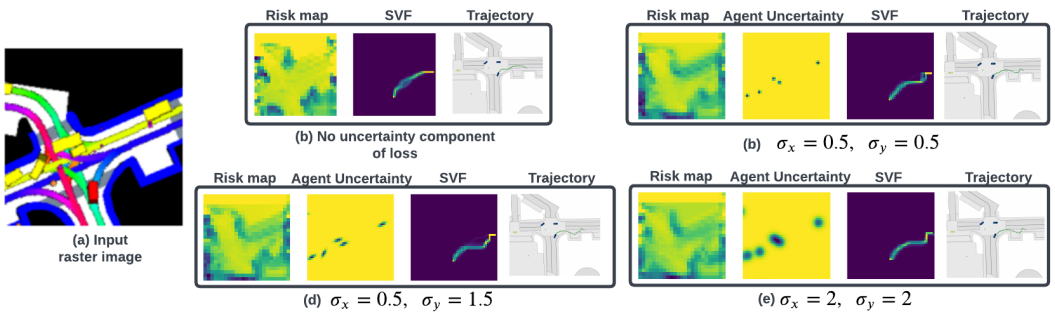


Figure 5-4: **Agent Uncertainty Shape.** For the sample scene (5-4a, right turn at an intersection), we examine the intermediate risk map and SVF, as well as the final trajectory, generated by the proposed planner for a variety of agent uncertainty shapes (5-3 b-e). We observe planners trained with slightly different shapes will respond by generating trajectories that match those shapes.

Lastly, we investigate the results of choosing a particular representation for uncertainty masks around agents over other representations. As discussed in section 4.4, the proposed model utilized uncertainty masks such that a bivariate Gaussian with $\sigma_x = 0.5$, $\sigma_y = 1.5$ surrounded each agent. Qualitative results of varying the σ's can be seen in figure 6. With no uncertainty overlap component included in the loss, 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 clearly avoids the nearby agent but allows for some flexibility for when it moves into the desired lane whereas 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 shape, but the SVF shows a slight possibility of a smooth turn. Overall, this indicates some level

of flexibility for desired agent avoidance behavior when designing a planner under the proposed architecture, even when the planner itself is not aware of these tuning parameters when run during inference.

5.5 Summary of Results

Not only does the proposed planner demonstrate superior ADE and goal achievement when compared to the other planner presented, but it does so without compromising safety. Self ablation on various components of the planner reveals the potential for fine-tuning behavior as desired. Increasing the resolution of the generated risk maps allows for more detail, which improves human driving similarity, though value iteration takes longer. Adjusting the variance of the goal result in better goal achievement when the variance is smaller, though difficulty turning comes with it as a trade-off. The planner can also be made more conservative with relation to nearby agents by increasing the uncertainty weighting in the loss function, and the manner in which the trajectories generated avoid agents can also be tuned by altering the shape surrounding each agent.

Chapter 6

Conclusions

Overall, we see great potential for the architecture presented with its ability to react to minor changes in the given goal and uncertainty masks. Designers can tune the planner's behavior, enforcing smaller goal variances when goal achievement is a priority or allowing for larger goal variances when other aspects 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.

6.1 Lessons Learned

Transitioning from standard semester-long classes with projects that span a month in time (from inception to completion), completing this thesis has inherently been a learning process. Rather than spend a day or two deciding on a project, I took a semester to become comfortable with the field by working on smaller tasks while thinking about potential gaps in the literature. This can feel discouraging if the expectation is to finish quickly, so I learned to keep perspective of longer term goals, like making meaningful contributions to the field of self driving.

During my first semester researching with the lab, I became more comfortable navigating larger project's codebases through practice. The relatively small project of creating visualizations for examining how another person's code and model worked

helped me develop as more nuanced understanding of PyTorch and PyTorch Lightning, and I am grateful to have had the opportunity to be introduced to the field without needing to work on a thesis immediately.

6.2 Future Work

In terms of the existing infrastructure, perhaps more can be done to explore its capabilities. For example, it could be tested on different datasets, including the Waymo Open Dataset [8], to determine whether a particular BEV setup can contribute to overall safer and human-like trajectories. In addition, it could be used in conjunction with the works discussed in 2.2 that generate a BEV from sensor data to create a full-stack pipeline. If effective, it could even be tested on a real car.

This work explored the ways in which grid-based risk can be fine-tuned and affect planner behavior. This is extremely helpful as many papers recently published demonstrate the functionality of using BEV, either as a primary input or as an intermediary representation. However, risk is always present in the field of autonomous driving, even if the representation is not grid-based. Therefore, future work could take inspiration from this exploration of explicit risk management and do the same for other common scene representations.

Bibliography

[1] Richard Bellman. *Dynamic Programming*. Princeton Univ. Pr., 1957.

[2] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020.

[3] Xuanyao Chen, Tianyuan Zhang, Yue Wang, Yilun Wang, and Hang Zhao. Futr3d: A unified sensor fusion framework for 3d detection. *ArXiv*, abs/2203.10642, 2022.

[4] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, J. Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. *ICRA*, pages 2090–2096, 2019.

[5] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. *ICRA*, 2019.

[6] Nachiket Deo and Mohan M. Trivedi. Trajectory forecasts in unknown environments conditioned on grid-based plans. *CoRR*, abs/2001.00735, 2020.

[7] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

[8] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aur'elien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9710–9719, October 2021.

[9] 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)*, pages 1–7, 2018.

[10] Iván Silva Feraud and José Eugenio Naranjo. Are you a good driver?: A data-driven approach to estimate driving style. In *ICCMS 2019*, 2019.

[11] J. Garcia and F. Fernández. A comprehensive survey on safe reinforcement learning. *JMLR*, 16:1437–1480, 2015.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[13] Anthony Hu, Zak Murez, Nikhil Mohan, Sof'ia Dudas, Jeffrey Hawke, Vijay Badrinarayanan, Roberto Cipolla, and Alex Kendall. Fiery: Future instance prediction in bird's-eye view from surround monocular cameras. *ArXiv*, abs/2104.10490, 2021.

[14] Xin Huang, Sungkweon Hong, Andreas G. Hofmann, and Brian C. Williams. Online risk-bounded motion planning for autonomous vehicles in dynamic environments. *CoRR*, abs/1904.02341, 2019.

[15] Yang Jiao, Zequn Jie, Shaoxiang Chen, Jing Chen, Xiaolin Wei, Lin Ma, and Yueping Jiang. Msmdfusion: Fusing lidar and camera at multiple scales with multi-depth seeds for 3d object detection. 2022.

[16] Kibeom Lee and Dongsuk Kum. Collision avoidance/mitigation system: Motion planning of autonomous vehicle via predictive occupancy map. *IEEE Access*, 7:52846–52857, 2019.

[17] Dachuan Li, Yunjiang Wu, Bing Bai, and Qi Hao. Behavior and interaction-aware motion planning for autonomous driving vehicles based on hierarchical intention and motion prediction. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2020.

[18] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *ArXiv*, abs/2205.13542, 2022.

[19] Abdelhak Loukkal, Yves Grandvalet, Tom Drummond, and You 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)*, pages 51–60, 2021.

[20] Kasra Mokhtari and Alan R. Wagner. Don't get yourself into trouble! risk-aware decision-making for autonomous vehicles, 2021.

[21] Tung Phan-Minh, Elena Corina Grigore, Freddy A. Boulton, Oscar Beijbom, and Eric M. Wolff. Covernet: Multimodal behavior prediction using trajectory sets. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14062–14071, 2020.

[22] João Silvério, Yanlong Huang, Fares 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)*, pages 90–97, 2019.

[23] Zachary Sunberg, Christopher Ho, and Mykel J. Kochenderfer. The value of inferring the internal state of traffic participants for autonomous freeway driving. *CoRR*, abs/1702.00858, 2017.

[24] N'uria Armengol Urp'i, S. Curi, and A. Krause. Risk-averse offline reinforcement learning. *ArXiv*, abs/2102.05371, 2021.

[25] Qiannan Wang and Matthias Gerdts. Risk-based path planning for autonomous vehicles. 2021.

[26] Ze yu Zhu and Huijing Zhao. A survey of deep rl and il for autonomous driving policy learning. *ArXiv*, abs/2101.01993, 2021.