2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Risk-Predictive Planning for Off-Road Autonomy

Lukas Lao Beyer[1], Gilhyun Ryou[1], Patrick Spieler[2] and Sertac Karaman[1]

***Abstract*— Efficiently navigating off-road environments presents a number of challenges arising from their unstructured nature. In the absence of high-fidelity maps, occlusions from obstacles and terrain lead to limited information available to inform planning decisions. Furthermore, resolution and latency limitations of real-world perception systems lead to potentially of degraded perception performance when traversing such environments at high speeds. We address these problems by proposing an algorithm which plans trajectories while anticipating future observations. In particular, we introduce a model which learns to predict the evolution of future riskmaps conditioned on the future path and speed profile of the vehicle. The model is trained in a self-supervised fashion using recordings of vehicle trajectories. We then present an algorithm which leverages a way to efficiently query the model along candidate paths and speed profiles to produce time-optimal trajectories while maintaining a bound on the future expected risk. We assess the predictive performance of our risk model through a comparison with real vehicle driving logs. Furthermore, our closed-loop simulations of several benchmark scenarios demonstrate how the behavior of our planner leads to qualitatively distinct trajectories, leading to improvements in both success rate and speed by up to 60%.**

I. Introduction

High-speed off-road driving in unknown environments poses unique challenges to autonomous navigation. For example, deserts and forests contain steep terrain and vegetation leading to many occlusions and unobserved hazards. In absence of a high-fidelity map, it is therefore crucial for any planning algorithm to reason about unobserved space.

Model-based *risk-aware planning* approaches tackle these challenges by quantifying uncertainty and imposing bounds on uncertainty along future trajectories, therefore actively averting potentially high-risk regions [1], [2], [3], [4]. For example, trajectories can be optimized while explicitly considering visibility and the inevitable collision sets resulting from potential unobserved obstacles [5]. Approaches in this category are able to provide safety guarantees, but may not improve navigation performance over longer horizons.

To enable longer-term foresight, *implicit planning* architectures have been proposed in which optimization of task performance leads to learning of latent world models with a temporal component [6]. For example, latent world models that perform future prediction can emerge to compensate for missing observations [7]. Dreamer [8] provides another example of learning a compressed latent world model. These approaches have the advantage of learning world models from data, but often require interaction with a simulation environment during training. Some approaches, like ViKiNG [9], can learn environment representations offline from recorded data, but also perform planning and prediction implicitly. Such planners and their latent world models may not be readily interpretable, posing challenges for deployment and integration into larger systems.

Fig. 1: Clockwise from the top left: The Polaris RZR vehicle used for data collection; two example environments showing hilly and forested areas, respectively; visualization of the risk-predictive planner in action, showing the selected trajectory (magenta) alongside rejected candidate paths (green) overlaid on an elevation-mapped occupancy map with unknown regions due to terrain occlusions.

Action-conditioned video prediction approaches attempt to directly predict sensor inputs rather than relying on latent world models. Many such approaches have been limited to low-resolution inputs and environments with a readily available simulator, such as Atari video games [10]. While more recent approaches such as Pathdreamer [11] can predict high-resolution video of real environments, and integrating such a predictive component into a model-based planner may turn out to be prohibitively slow. The *occupancy anticipation* approach presented in [12] predicts action-conditioned future occupancy maps explicitly, but relies on reinforcement learning to learn policies which leverage the predictions. Furthermore, training these models requires availability of high quality 3D simulations.

Attempting to combine desirable properties of these different approaches while mitigating the drawbacks, we present the following contributions. First, a novel formulation of the problem of estimating the future evolution of the riskmap given an arbitrary planned trajectory, in a manner which captures dependence on traversal speed as well as on visibility. Second, an efficient model which allows querying

[1]Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA. {llb, ghryou, sertac}@mit.edu
[2]NASA Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. patrick.spieler@jpl.nasa.gov

individual points of interest of predicted future riskmaps, and a self-supervised framework for offline training of the model on readily available recordings of vehicle trajectories and perceived riskmaps. Finally, a risk-predictive motion planner which uses the trained risk model to co-optimize a speed profile and path selection, computing a time-optimal trajectory while enforcing bounds on future expected risk. In summary, efficient prediction of future risk values which are directly used by the planner enables real-time operation of our planner in complex environments such as those shown in Fig. 1, and the model-based nature of the main planning algorithm facilitates integration and deployment in the context of complex real-world systems.

II. Preliminaries

Following the development in [13], we understand a risk metric to be a function mapping an uncertain cost distribution to a concrete risk value. We denote a function mapping some location in the vehicle's neighborhood to its risk value as a *riskmap*. Furthermore, we refer to the current riskmap, produced by the perception pipeline using the latest available sensor measurements, as the *perceived riskmap*. This is to distinguish it from future *predicted* riskmaps. Unlike the perceived risk, we treat future risk as a random variable due to its dependence on unknown future sensor inputs.

A. Choice of risk metric

Our algorithm will assume the availability of the perceived riskmap. The chosen risk metric should capture measurement uncertainty in the sense that regions with few or no observations shall be assigned some positive risk value. For instance, an estimate of the conditional value-at-risk (CVaR) [13] or a heuristic risk metric incorporating the number or density of observations for each position may be used. Typically, such a risk metric will assign very high risk to unobserved parts of the map, causing it to be dependent on the vehicle's traversed trajectory. Clearly, obstacle and terrain occlusions induce a dependence on the traversed path. However, we note that under latency and update frequency limits of any perception pipeline or sparsity of measurements provided by sensors such as certain LIDARs, there is also a dependence of the risk on the speed profile along the traversed trajectory.

In our case, we employ a heuristic risk metric which assigns risk values in $[0, 1]$ based on known occupancy (obtained from a pointcloud produced by several LIDARs, and including semantic information from camera images), while additionally considering any regions with a low number of LIDAR observations as lethal.

Since visibility is essential in capturing the behavior of the riskmap, we additionally assume that an elevation map is available. This elevation map may contain regions of unknown elevation.

B. Notation and risk model

Trajectories are discretized at a set of $N + 1$ fixed sample points located at arclengths s_n with $0 \leq n \leq N$, where $0 = s_0 < s_1 < \cdots < s_N$. Denote the arclengths, times, positions, and orientations corresponding to these sample points as

$$\begin{aligned} \mathbf{s} &:= \begin{bmatrix} s_0 \cdots s_N \end{bmatrix}^\top, \quad \mathbf{t} := \begin{bmatrix} t_0 \cdots t_N \end{bmatrix}^\top, \\ \mathbf{p} &:= \begin{bmatrix} p_0 \cdots p_N \end{bmatrix}^\top, \quad \mathbf{o} := \begin{bmatrix} o_0 \cdots o_N \end{bmatrix}^\top, \end{aligned} \tag{1}$$

respectively, and write $\mathbf{x}_{n:m} := \begin{bmatrix} x_n \cdots x_m \end{bmatrix}^\top$ to denote slicing of a sequence $\mathbf{x}$. Denote the discretized trajectory by the tuple $\mathcal{T} := (\mathbf{s}, \mathbf{t}, \mathbf{p}, \mathbf{o})$ and let the notation $\mathcal{T}_{n:m}$ denote slicing of each of the trajectory tuple's elements.

Let $R : \mathbb{R}^2 \to \mathbb{R}$ denote the perceived riskmap, so that $R(q)$ is the risk value at a particular position $q \in \mathbb{R}^2$. Similarly, let $M : \mathbb{R}^2 \to \mathbb{R}$ denote a map containing terrain elevation data as obtained from the perception pipeline.

We model future risk as a random variable with dependence on the perceived risk map, the elevation map, and the vehicle's trajectory. Letting R_n denote the future riskmap which will be perceived after traversing an arclength of s_n along the trajectory $\mathcal{T}$, we aim to estimate the *predictive risk*

$$h_n := \mathbb{E}[R_n \mid M, R_0, \mathcal{T}_{0:n}]\,. \tag{2}$$

We will denote our approximation of h_n, introduced in Section III-A, by $\hat{h}_n$.

III. Algorithm

We propose an algorithm to find a time-optimal trajectory while imposing a bound on the predictive risk along it. The full algorithm operates on a set of candidate paths, producing an optimal path selection as well as the corresponding speed profile, but we first consider the speed adaptation problem without path selection. The solution to this problem is a trajectory $\mathcal{T}$ which minimizes traversal time along a predefined path $(\mathbf{s}, \mathbf{p}, \mathbf{o})$ while keeping the predictive risk $\hat{h}_n$ along the trajectory below a specified threshold:

$$\begin{aligned} \arg\min_{\mathbf{t}} \quad & t_N \\ \text{subject to} \quad & \max_{i \in \{0,\ldots,N\}} \hat{h}_i(p_i) \leq h_{\max} \\ & f(\mathcal{T}) \geq 0. \end{aligned} \tag{3}$$

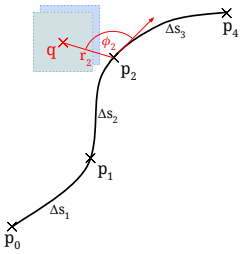
Here, $h_{\max}$ is a user-defined risk threshold, and additional constraints such as limits on speed or acceleration have been encapsulated into the constraint on $f(\mathcal{T})$.

A. Riskmap prediction

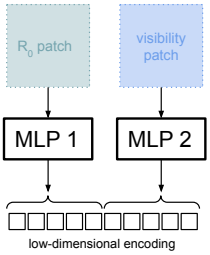
In solving (3), it is crucial to introduce a predictive riskmap model which is efficient to sample. To this end, we impose the following pointwise recursive structure on our approximation of the predictive risk:

$$\hat{h}_n(q) = g(\hat{h}_{n-1}(q), M, R_0, \mathcal{T}_{n-1:n}, q), \text{ with } \hat{h}_0 = R_0. \tag{4}$$

Here, g is a function which pointwise describes the evolution of the expected riskmap after traversal of a single segment of the discretized trajectory $\mathcal{T}_{n-1:n}$, so that prediction of future riskmaps at various timesteps can be obtained through recursive application. Note that global or local context around the point q may be provided to g by the means of the full perceived riskmap R_0 and the full elevation map M, but



(a) Egocentric polar coordinates used as part of the input query to the GP model.



(b) Encoding of square patches of initial riskmap and precomputed visibility information.

Fig. 2: Gaussian process risk prediction model inputs.

full riskmaps (as opposed to the individual samples $\hat{h}_n(q)$) deliberately do not participate in the recursion to facilitate efficient evaluation of the predictive risk.

1) Riskmap evolution model: We use Gaussian process (GP) regression [14] to model samples of the future riskmap $\hat{h}_n(q)$ at a particular position q:

$$\hat{h}_n(q) = \mathbb{E}\left[\hat{g}(\hat{h}_{n-1}(q), \Psi_n(R_0, M, q), \overline{\Delta\mathcal{T}_n}, \overline{q})\right]. \quad (5)$$

Here, $\hat{g}$ denotes the predictive distribution of a GP with a scaled squared exponential kernel. The query position q is expressed in egocentric polar coordinates $\overline{q} = (r_n, \phi_n)$ with respect to the vehicle's future pose (p_n, o_n) as illustrated in Fig. 2a. The trajectory step $\mathcal{T}_{n-1:n}$ is expressed as an arclength step size and an average velocity $\overline{\Delta\mathcal{T}_n} = (\Delta s_n, v_n)$, where $\Delta s_n = s_n - s_{n-1}$ and $v_n = \Delta s_n/(t_n - t_{n-1})$. The initial riskmap R_0 and visibility information computed using the initial elevation map M are encoded into a low-dimensional feature vector with the function Ψ_n.

2) Geometric context encoding: As discussed in Section II-A, visibility has a strong influence on the riskmap, so we assume the availability of a terrain elevation map M. Using the vehicle pose (p_n, o_n) and the elevation map we compute a two-layer explicit visibility map via raycasting, containing (i) a binary visibility mask describing which regions of the map are certainly visible, and (ii) an upper-bound-slope map, containing the slope of the steepest visibility ray which is not blocked by terrain occlusion. This latter layer is helpful in quantifying potential visibility improvements in unobserved areas of the map. See Fig. 3 for an illustration of the upper-bound-slope computation.

As shown in Fig. 2b, a patch centered around q is extracted from this two-layer visibility map, and fed into an encoder model. Since (5) discards all but a single point of the input riskmap, we encode a patch of the initial riskmap R_0 for added context. Each encoder is composed of two three-layer multi layer perceptrons outputting a low-dimensional encoding. The concatenation of the two encodings is fed into the GP. In summary, we define Ψ_n and Ψ as follows:

$$\Psi(P_1, P_2) := \text{concat}(\text{MLP}_1(P_1), \text{MLP}_2(P_2)) \quad (6)$$

$$\Psi_n(R_0, M, q) := \Psi(\pi(R_0, q), \pi(\beta(M, p_n, o_n), q)), \quad (7)$$

where $\pi(\cdot, q)$ denotes the patch extraction centered at q, and $\beta(\cdot, p_n, o_n)$ performs computation of the visibility maps using the vehicle pose (p_n, o_n) as described in Fig. 3.

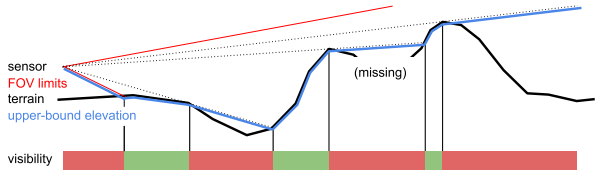


Fig. 3: Visibility raycasting along a particular azimuth angle. Only parts of the terrain which are directly visible are marked as such. Upper-bound-slope is calculated as the slope of the upper-bound-elevation shown in blue.

3) Implementation and model training: We train the model in a self-supervised fashion on a dataset containing recordings of the vehicle's traversed trajectory alongside the perceived riskmap and terrain elevation map. Denote such a dataset, containing K risk and elevation maps and the corresponding trajectory, as $\mathcal{D}$:

$$\mathcal{D} = \{(\mathcal{T}_k, M_k, R_k)\}_{k\in\{1\ldots K\}}. \quad (8)$$

Here, $\mathcal{T}_k$ is the trajectory sample corresponding to M_k and R_k, which denote the perceived elevation and riskmaps, respectively.

To generate a training example, a frame index $k \in \{1 \ldots k - N\}$ is sampled uniformly at random. We then sample a point q lying within a distance r_q of the trajectory segment $T_{k:k+N}$, and build the vector of reference risk values

$$\mathbf{H} = \left[R_{k+1}(q) \cdots R_{k+N}(q)\right]^\top. \quad (9)$$

Thus, $\mathbf{H}$ contains the evolution of a particular point in the riskmap as the vehicle traverses the trajectory segment. The full training example is then given by $D = (X, \mathbf{H})$, where

$$X = (\mathcal{T}_{k:k+N}, M_k, R_k). \quad (10)$$

It is important to train the model with more than one prediction step (i.e. $N > 1$) to ensure stability when recursively applying many prediction steps at test time. Since the training dataset usually lacks examples of the vehicle traversing high-risk areas, we further note that the radius r_q not be too small. In particular, it must be large enough to allow the sampling of high-risk areas of the map.

To make training on large datasets tractable and to enable minibatch training, we implement $\hat{g}$ as a stochastic variational GP [15], [16]. During training, the combined GP and encoders model (5) is applied recursively to a minibatch of inputs. We wish to minimize the Kullback-Leibler divergence between the posterior distribution predicted by our model and the corresponding distribution of ground truth data from the training examples. As a proxy for minimizing the KL divergence in the variational GP setting, we maximize the evidence-lower bound (ELBO) [17]. Concretely, the ELBO of the true risk values from $\mathbf{H}$ is computed for each application of $\hat{g}$. The sum of the ELBO values is maximized using the Adam optimizer [18]. Parameters of the variational GP and two encoder models are updated jointly within the optimization. Fig. 4 illustrates how the model's recursion is rolled out at training and test time, and provides an overview of the inputs, outputs, the optimization objective used during training, as well as the various input preprocessing steps.

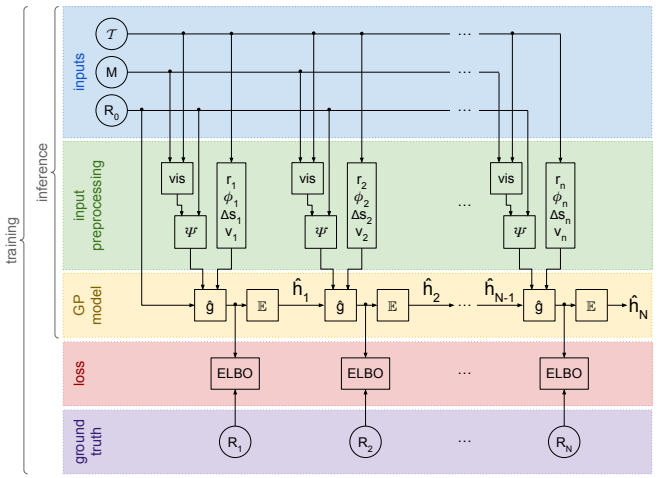


Fig. 4: Schematic depiction of risk prediction model, illustrating how the input trajectory $\mathcal{T}$, elevation map M and riskmap R_0 are preprocessed and fed to the GP model $\hat{g}$ in order to produce estimates of the future expected risk, $\hat{h}_1, \ldots, \hat{h}_N$ at a particular point q (not shown). Also shown is the computation of the evidence lower bound (ELBO) of the ground truth risk values $R_1, \ldots, R_N$, which occurs for each prediction step and whose sum is maximized in order train both the encoders Ψ and the GP model $\hat{g}$.

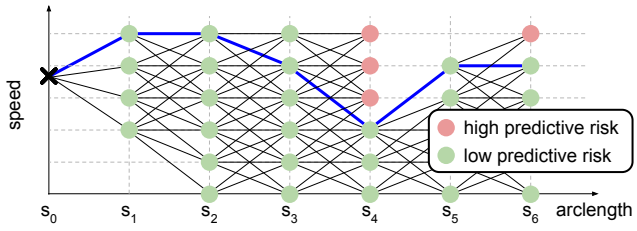


Fig. 5: Speed adaptation with predictive risk constraint (3) solved via tree search. The solution is shown in blue. We implement A* search guided by a heuristic (the remaining time to reach the end of the path assuming maximum acceleration and speed), so expansion of all states shown here does not occur. Note also that this figure shows only a single dot at each (s, v) point, but in fact there are multiple distinct predictive risk values corresponding to every possible path leading to that point.

B. Speed profile search

The speed adaption (3) is solved via an A*-based [19] search operating on a discretized set of velocities (see Fig. 5). The search is initialized with the current vehicle speed as its root state, and proceeds by expanding child speeds corresponding to the next arclength discretization point along the input path. We consider constraints $f(\mathcal{T}) \geq 0$ which impose maximum speed and acceleration limits. They are enforced by skipping the expansion of infeasible child states.

Enforcement of the predictive risk constraint involves a query to the model g, which computes $\hat{h}(p_i)$ based on the candidate speed profile. Observe that, except for the speed v_n, all inputs to the risk prediction model are speed-independent, including visibility raytracing, patch extraction, and MLP encoding. Thus, these inputs are precomputed once before starting the search. Furthermore, when expanding child states and evaluating $\hat{h}_n$, the recursive nature of the model g is exploited by caching the parent state's query $\hat{h}_{n-1}$. Finally, we leverage favorable performance scaling of GPU-accelerated inference to large batches of inputs by performing thousands of individual queries to the model $\hat{g}$ at once, in a single batch. Therefore, unlike standard A*, our algorithm expands large portions of the search frontier simultaneously.

C. Candidate path generation and selection

We extend the proposed speed adaptation algorithm to the complete path candidate selection and speed optimization problem. This is achieved by first sampling several candidate paths, and then applying a modified version of the speed adaptation algorithm to the full set of candidates.

1) Candidate path generation: In order to navigate towards a long-horizon goal which may be located well beyond the planning horizon of the risk predictive planner, we require a long-horizon planner which is able to produce heuristic cost-to-go values. We implement this long-horizon planner as a coarse Dijkstra search producing a map of estimated time to reach the goal for each pose. This planner is not risk aware and operates instead on the available occupancy map and desired waypoint locations.

To obtain kinematically feasible candidate paths, we additionally require a short-horizon ($< 50\,\mathrm{m}$) planner producing feasible paths between arbitrary poses. It uses a costmap which considers known occupancy and terrain, and is therefore "optimistic" in the sense it treats unknown space as traversable. Standard techniques for kinodynamic planning can be applied to implement such a planner. In our case, a tree search over a state lattice is performed.

To generate candidate paths, the long-horizon planner is first used to choose an intermediate goal region located within the short-horizon planner's horizon. The short-horizon planner then produces a path $\mathcal{P}_{\text{ref}}$ from the start pose to the intermediate goal. Finally, we generate additional candidate paths by considering a semicircle around the current pose, with the center of the semicircle chosen to coincide with the final position of $\mathcal{P}_{\text{ref}}$. The bottom left panel in Fig. 1 shows example candidate paths generated in this manner.

2) Joint candidate selection and speed optimization: Let $\mathcal{C} := \{(\mathcal{P}^{(1)}, c_1), \ldots, (\mathcal{P}^{(K)}, c_K)\}$ denote the set of candidate paths $\mathcal{P}^{(k)}$ and associated costs c_k. These costs correspond to the heuristic cost-to-go produced by the long-horizon planner for the final pose of each candidate path. The solution to the joint path selection and speed adaptation problem is given by

$$\begin{aligned} \underset{k,\mathbf{t}^{(k)}}{\arg\min} \quad & c_k + t_N^{(k)} \\ \text{subject to} \quad & \max_{i \in \{0,\ldots,N\}} \hat{h}_i(p_i^{(k)}) \leq h_{\max} \quad \forall k \\ & f(\mathcal{T}^{(k)}) \geq 0 \qquad\qquad\quad \forall k. \end{aligned} \tag{11}$$

The batch-A* algorithm from Section III-B can easily be adapted to also solve the path selection problem. By initializing the search with K roots corresponding to each candidate path, speed profile are expanded along each candidate path in a single search. Compared to running an individual speed profile search for each candidate, using the same search queue for all candidates results in computational savings, since the heuristic prevents the computation of full speed profiles for every candidate.

IV. Experiments

The proposed algorithm is evaluated in a simulation environment that takes into account the dynamics of the Polaris PZR vehicle on different types of terrain, including scenarios with changes in elevation leading to potential terrain occlusions and significant roll and pitch of the vehicle. Raycasting according to realistic sensor specifications is used to accurately model LIDAR measurements. To better understand the predictive performance of our risk model, we first evaluate dense predictive riskmaps against ground truth derived from real-world driving logs. Then, we evaluate the complete speed adaptation and path selection pipeline in six environments with diverse terrain and obstacle configurations. A video presentation of the experimental results can be found at https://youtu.be/Jh1iXktZjAg.

The training data for the risk-prediction model consists of more than 50,000 frames of simulated vehicle driving data, covering approximately $20\,\mathrm{km}$ at multiple driving speeds over varied terrain. Input risk and elevation maps have a resolution of $2\,\mathrm{m}^{-1}$ and encompass a square area of $200\,\mathrm{m} \times 200\,\mathrm{m}$ around the vehicle's position. The patches extracted from the initial riskmap and the precomputed elevation information have a size of $16.5\,\mathrm{m} \times 16.5\,\mathrm{m}$, and the MLPs both contain two hidden layers with 256 and 512 layers, each producing a 6-dimensional encoding. Trajectories are discretized with an arclength step size of $\Delta s \approx 2\,\mathrm{m}$. The GP and encoder models are implemented in *PyTorch* [20] using the *GPyTorch* library [21]. The path selection procedure is performed with 16 candidate trajectories, where each candidate is planned with a $30\,\mathrm{m}$ horizon. The risk threshold is set to $h_{\max} = 0.15$. The complete path selection and speed adaptation pipeline executes at $2\,\mathrm{Hz}$ on an NVIDIA RTX A5000, utilizing approximately $2\,\mathrm{GiB}$ of GPU memory.

A. Evaluation of predictive riskmaps

While our planning algorithm relies on prediction of single points, it is illustrative to visualize densely sampled parts of the riskmap. For this experiment, we extract a ground-truth trajectory alongside the perceived risk and elevation maps from a recording from a real-world, human-driven trajectory of the Polariz RZR vehicle. Note that the riskmap prediction model is trained only on simulated driving logs, so this experiment additionally provides an example of good sim-to-real transfer performance. Fig. 6 presents examples of such ground-truth riskmaps alongside corresponding dense predictions. We highlight that the model is able to produce predictions which capture the future evolution of the riskmap when compared to the ground-truth even over long prediction horizons of more than $50\,\mathrm{m}$. Such long horizon lengths require application of over 20 recursive prediction steps $\hat{g}$, which is significantly more than the number of steps N used during training. Yet, we find that the recursive prediction process remains stable and accurate, as further evidenced by the accuracy metrics a_h and a_l remaining near or above 80% (see Fig. 6). Here, the accuracy metric a_h is defined as the ratio of pixels corresponding to high-risk areas of the riskmap which were correctly predicted as being high-risk, considering the square $80\,\mathrm{m} \times 80\,\mathrm{m}$ patch of pixels centered around the current vehicle position. The metric a_l is defined analogously, but considers low-risk areas of the riskmap. The threshold between high and low risk is set to $h_{\max}$. Additionally, the metrics a_h and a_l are computed along randomly sampled rollouts along the trajectories from our simulation dataset. At a prediction horizon of $40\,\mathrm{m}$, we achieve mean accuracies of $a_h = 91\%$ / $a_l = 87\%$ on the test set, and $a_h = 92\%$ / $a_l = 87\%$ on the training set.

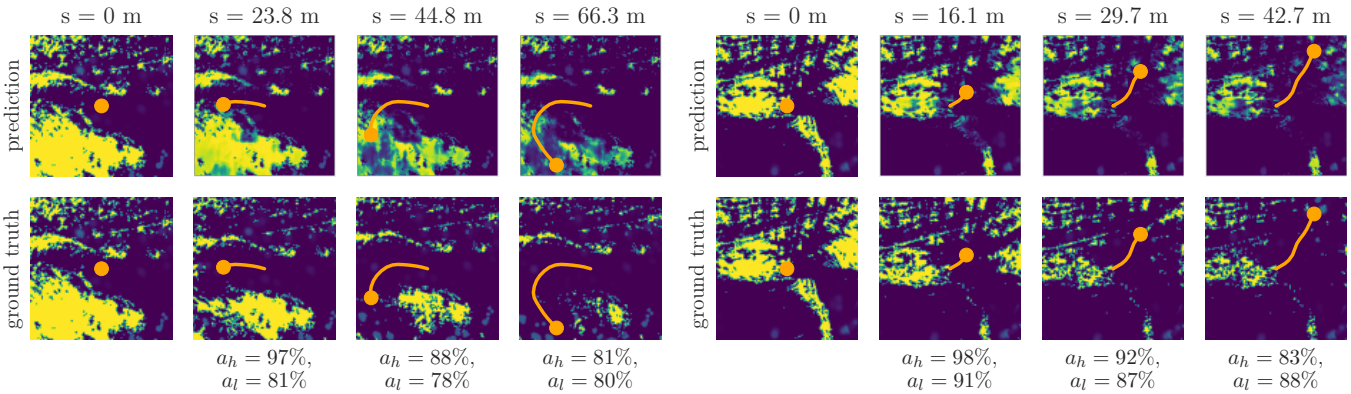


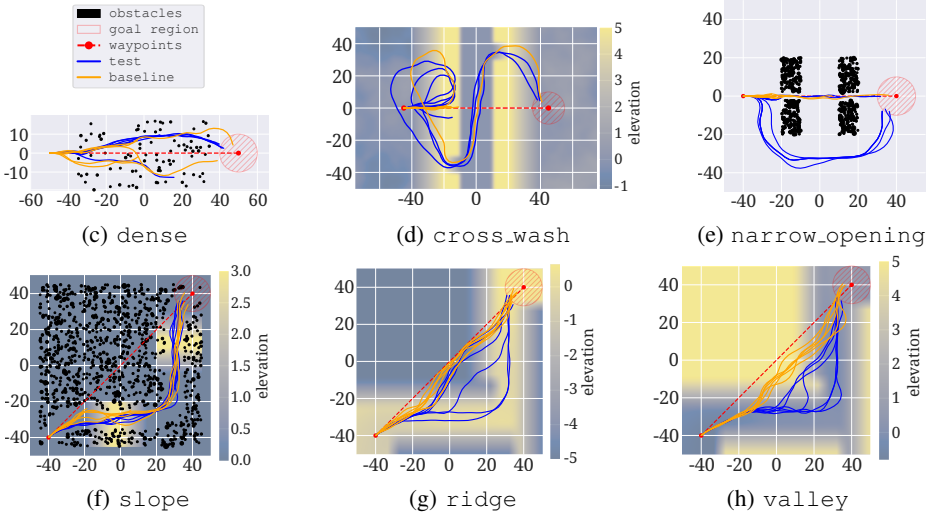
Fig. 6: Dense future riskmap predictions, showing the expected future risk (top) along a given trajectory (orange) for every pixel in the image, alongside ground truth (bottom). Purple is zero risk, while yellow is maximum risk. The model only sees the perceived risk and elevation maps at $s = 0\,\mathrm{m}$ (first column), and is rolled out recursively to produce long-horizon predictions. The dataset used for these examples contains human-driven trajectories in a hilly desert environment, as recorded the Polaris RZR vehicle. The riskmap prediction model is trained on a dataset consisting purely of simulation clips.

	baseline	test
dense	0.67 ± 0.25	0.90 ± 0.14
cross_wash	0.54 ± 0.21	0.65 ± 0.25
narrow_opening	0.49 ± 0.21	0.77 ± 0.30
slope	0.79 ± 0.29	1.00 ± 0.00
ridge	1.00 ± 0.00	1.00 ± 0.00
valley	1.00 ± 0.00	1.00 ± 0.00

(a) Completion rate ($\pm$ stddev).

	baseline	test
dense	2.07 ± 0.41	2.26 ± 0.35
cross_wash	2.45 ± 0.57	2.46 ± 0.74
narrow_opening	1.56 ± 0.16	2.55 ± 0.79
slope	2.64 ± 0.64	2.88 ± 0.25
ridge	3.13 ± 0.26	3.33 ± 0.25
valley	3.16 ± 0.13	3.19 ± 0.37

(b) Avg. speed in $^{m}/_{s}$ ($\pm$ stddev).



(c) `dense` (d) `cross_wash` (e) `narrow_opening`

(f) `slope` (g) `ridge` (h) `valley`

Fig. 7: Results from closed-loop simulation experiments of the riskmap prediction based planner (*test*) alongside those of the same planning stack without the riskmap prediction based speed adaptation and path selection (*baseline*). Aggregated results are shown in tables 7a and 7b. The completion rate in 7a is computed as $1 - \frac{d_{\text{final}}}{d_{\text{init}}}$, where d_{final} and d_{init} are the Euclidean distances of the vehicle to the goal region at the end and at the beginning of the simulation, respectively.

B. Closed-loop simulation results

We compare the closed-loop performance of our planner against a baseline planner which is identical to the proposed planning stack as introduced in Section III-C, but does not impose any constraints on the predictive risk. Without the predictive-risk-based speed adaptation, the path selection step causes this baseline planner to always follow the short-horizon planner's reference $\mathcal{P}_{\text{ref}}$, as the a priori cost of all alternate candidate paths (i.e. without considering predictive risk constraints) is higher. However, the baseline planner is subject to the same speed and acceleration constraints f as introduced in (3). A summary of our evaluation can be found in Fig. 7, which showcases scenarios with significant or potentially unexpected changes in occlusion and elevation. We highlight that in many cases a significant improvement in success rate, as well as an increase in speed, can be observed. This is due to qualitatively different behavior as compared to the baseline planner. Across the evaluated scenarios, the proposed planning algorithm demonstrates up to 57% higher completion rate and up to 63% higher average speed compared to the baseline method.

We focus now on a few types of behavior visible in the simulation results from Fig. 7c–7h. For example, in the scenario from Fig. 7d, a wash is flanked by higher elevation terrain on both sides. This terrain obstructs view of steep, untraversable slopes on the banks of the wash. As a result, the baseline planner approaches the steep sides until it is too late to recover without backing up in 5 out of 6 runs. However, the risk-predictive planner achieves higher success rate by initiating an early turn instead of waiting to observe the untraversable slope. In Fig. 7e, our planner achieves significantly higher speeds and completion rate as well as lower total traversal time than the baseline by bypassing the obstacle fields instead of traversing a narrow gap which requires slowdown and could cause the vehicle to become stuck. Fig. 7g shows a scenario where our planner does not achieve improved success rates and only slightly higher speeds, but exhibits qualitatively safer behavior in several runs by remaining on a ridge rather than descending into lower elevation terrain with more reduced visibility.

V. Conclusions

Our risk-predictive planner combines a novel future riskmap prediction approach which is efficient to query at individual points with a simple tree-search based search algorithm that bounds the predictive risk along the output trajectory. A self-supervised training procedure enables the predictive risk model to learn the evolution of future risk at particular points of interest taking into account complex behavior of the perception pipeline and its behavior at different speeds and under changing visibility. This approach results in a planner that can leverage anticipation learned from data for better decision making in complex unstructured environments. Directions for future work could include the implementation of more sophisticated planners that do not rely on predefined candidate paths to incorporate a predictive risk constraint, or exploring the use of predictive variances provided by the GP models [22] to compute confidence bounds on the predictive risk. The inclusion of additional temporal context, such as perceived risk at both the current and several past frames, may also be an avenue for further improvements to the predictive performance of our model.

Acknowledgment

The research was partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

REFERENCES

[1] J. P. Wilson, Z. Shen, S. Gupta, and T. A. Wettergren, "T*-Lite: A fast time-risk optimal motion planning algorithm for multi-speed autonomous vehicles," in *Global Oceans 2020: Singapore–US Gulf Coast*. IEEE, 2020, pp. 1–6.

[2] K. Otsu, A.-A. Agha-Mohammadi, and M. Paton, "Where to look? Predictive perception with applications to planetary exploration," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 635–642, 2017.

[3] Y. K. Nakka and S.-J. Chung, "Trajectory Optimization of Chance-Constrained Nonlinear Stochastic Systems for Motion Planning Under Uncertainty," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 203–222, 2022.

[4] L. Sharma, M. Everett, D. Lee, X. Cai, P. Osteen, and J. P. How, "RAMP: A Risk-Aware Mapping and Planning Pipeline for Fast Off-Road Ground Robot Navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5730–5736.

[5] L. Janson, T. Hu, and M. Pavone, "Safe Motion Planning in Unknown Environments: Optimality Benchmarks and Tractable Policies," in *Robotics: Science and Systems*, Pittsburgh, USA, June 2018.

[6] D. Ha and J. Schmidhuber, "Recurrent World Models Facilitate Policy Evolution," in *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 2451–2463, https://worldmodels.github.io.

[7] D. Freeman, D. Ha, and L. Metz, "Learning to Predict Without Looking Ahead: World Models Without Forward Prediction," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[8] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to Control: Learning Behaviors by Latent Imagination," in *International Conference on Learning Representations*, 2020.

[9] D. Shah and S. Levine, "ViKiNG: Vision-Based Kilometer-Scale Navigation with Geographic Hints," in *Robotics: Science and Systems XVIII*. Robotics: Science and Systems Foundation, jun 2022.

[10] J. Oh, X. Guo, H. Lee, R. Lewis, and S. Singh, "Action-Conditional Video Prediction Using Deep Networks in Atari Games," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 2863–2871.

[11] J. Y. Koh, H. Lee, Y. Yang, J. Baldridge, and P. Anderson, "Pathdreamer: A World Model for Indoor Navigation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[12] Z. A.-H. Santhosh Kumar Ramakrishnan and K. Grauman, "Occupancy Anticipation for Efficient Exploration and Navigation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[13] A. Majumdar and M. Pavone, "How Should a Robot Assess Risk? Towards an Axiomatic Theory of Risk in Robotics," in *Robotics Research*, N. M. Amato, G. Hager, S. Thomas, and M. Torres-Torriti, Eds. Cham: Springer International Publishing, 2020, pp. 75–84.

[14] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. The MIT Press, 2006.

[15] M. Titsias, "Variational Learning of Inducing Variables in Sparse Gaussian Processes," in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 5. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr 2009, pp. 567–574.

[16] Hensman, James and Fusi, Nicolò and Lawrence, Neil D., "Gaussian processes for big data," in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'13. Arlington, Virginia, USA: AUAI Press, 2013, p. 282–290.

[17] J. Hensman, A. Matthews, and Z. Ghahramani, "Scalable variational Gaussian process classification," in *Artificial Intelligence and Statistics*. PMLR, 2015, pp. 351–360.

[18] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations*, 12 2014.

[19] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[21] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, "GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration," in *Advances in Neural Information Processing Systems*, 2018.

[22] M. Jankowiak, G. Pleiss, and J. Gardner, "Parametric gaussian process regressors," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4702–4712.