

STEP: Stochastic Traversability Evaluation and Planning for Risk-Aware Off-road Navigation

David D. Fan^{*1}, Kyohei Otsu^{*1}, Yuki Kubo², Anushri Dixit³,
Joel Burdick³, and Ali-Akbar Agha-Mohammadi¹

Abstract— Although ground robotic autonomy has gained widespread usage in structured and controlled environments, autonomy in unknown and off-road terrain remains a difficult problem. Extreme, off-road, and unstructured environments such as undeveloped wilderness, caves, and rubble pose unique and challenging problems for autonomous navigation. To tackle these problems we propose an approach for assessing traversability and planning a safe, feasible, and fast trajectory in real-time. Our approach, which we name STEP (Stochastic Traversability Evaluation and Planning), relies on: 1) rapid uncertainty-aware mapping and traversability evaluation, 2) tail risk assessment using the Conditional Value-at-Risk (CVaR), and 3) efficient risk and constraint-aware kinodynamic motion planning using sequential quadratic programming-based (SQP) model predictive control (MPC). We analyze our method in simulation and validate its efficacy on wheeled and legged robotic platforms exploring extreme terrains including an abandoned subway and an underground lava tube. (See video: <https://youtu.be/N97cv4eH5c8>)

I. INTRODUCTION

Consider the problem of a ground robot tasked to autonomously traverse an unknown environment. In real-world scenarios, environments which are of interest to robotic operations are highly risky, containing difficult geometries (e.g. rubble, slopes) and non-forgiving hazards (e.g. large drops, sharp rocks) (See Figure 1) [22]. Determining where the robot may safely travel is a non-trivial problem, compounded by several issues: 1) Localization error affects how sensor measurements are accumulated to generate dense maps of the environment. 2) Sensor noise, sparsity, and occlusion induces biases and uncertainty in analysis of traversability. 3) Environments often pose a mix of various sources of traversability risk, including slopes, rough terrain, low traction, narrow passages, etc. 4) These various risks create highly non-convex constraints on the motion of the robot, which are compounded by the kinodynamic constraints of the robot itself.

To address these issues we adopt an approach in which we directly quantify the traversal cost along with the



Fig. 1. Top left: Boston Dynamics Spot quadruped robot exploring Valentine Cave at Lava Beds National Monument, CA. Top right, bottom left: Clearpath Husky robot exploring Arch Mine in Beckley, WV. Bottom middle, right: Spot exploring abandoned Satsop power plant in Elma, WA.

uncertainties associated with that cost. We refer to this cost as *traversability*, e.g. a region of the environment in which the robot will suffer or become damaged has a high traversability cost. Building upon our previous work on traversability in extreme terrains [38], we formulate the problem as a risk-aware, online nonlinear Model Predictive Control (MPC) problem, in which the uncertainty of traversability is taken into account when planning a trajectory. Our goal is to minimize the traversability cost, but directly minimizing the mean cost leads to an unfavorable result because tail events with low probability of occurrence (but high consequence) are ignored (Figure 2). Instead, in order to quantify the impact of uncertainty and risk on the motion of the robot, we employ a formulation in which we find a trajectory which minimizes the Conditional Value-at-Risk (CVaR) [33]. Because CVaR captures the expected cost of the tail risk past a given probability threshold, we can dynamically adjust the level and severity of uncertainty and risk we are willing to accept (which depends on mission-level specifications, user preference, etc.). While online chance-constrained nonlinear MPC problems often suffer from a lack of feasibility, our approach allows us to relax the severity of CVaR constraints by adding a penalizing loss function.

We quantify risk via a traversability analysis pipeline (for system architecture, see Figure 3). At a high level, this pipeline creates an uncertainty-aware 2.5D traversability map of the environment by aggregating uncertain sensor measurements. Next, the map is used to generate both environment and robot induced costs and constraints. These constraints are convexified and

^{*}These authors contributed equally to this work.

¹These authors are with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA {david.d.fan, kyohei.otsu, aliaghaha}@jpl.nasa.gov

²This author is with the University of Tokyo, Tokyo, Japan kubo.yuki@ac.jaxa.jp

³These authors are with California Institute of Technology, Pasadena, CA {adixit, jwb}@robotics.caltech.edu

©2021, California Institute of Technology. All Rights Reserved

used to build an online receding horizon MPC problem, which is solved in real-time. As we will demonstrate, we push the state-of-the-art in making this process highly efficient, allowing for re-planning at rates which allow for dynamic responses to changes and updates in the environment, as well as high travel speeds.

In this work, we propose STEP (Stochastic Traversability Evaluation and Planning), that pushes the boundaries of the state-of-the-practice to enable safe, risk-aware, and high-speed ground traversal of unknown environments. Specifically, our contributions include:

- 1) Uncertainty-aware 2.5D traversability evaluation which accounts for localization error, sensor noise, and occlusion, and combines multiple sources of traversability risk.
- 2) An approach for combining these traversability risks into a unified risk-aware CVaR planning framework.
- 3) A highly efficient MPC architecture for robustly solving non-convex risk-constrained optimal control problems.
- 4) Real-world demonstration of real-time CVaR planning on wheeled and legged robotic platforms in unknown and risky environments.

II. RELATED WORK

Our work is related to other classical approaches to traversability. Most traversability analyses are dependent on sensor type and measured through geometry-based, appearance-based, or proprioceptive methods [32]. Geometry-based methods often rely on building a 2.5D terrain map which is used to extract features such as maximum, minimum, and variance of the height and slope of the terrain [18]. Planning algorithms for such methods take into account the stability of the robot on the terrain [19]. In [30, 17], the authors estimate the probability distributions of states based on the kinematic model of the vehicle and the terrain height uncertainty. Furthermore, a method for incorporating sensor and state uncertainty to obtain a probabilistic terrain estimate in the form of a grid-based elevation map was considered in [15]. Our work builds upon these ideas by performing traversability analyses using classical geometric methods, while incorporating the uncertainty of these methods for risk-aware planning [2, 38].

Risk can be incorporated into motion planning using a variety of different methods, including chance constraints [29, 39], exponential utility functions [24], distributional robustness [40], and quantile regression [14, 11]. Risk measures, often used in finance and operations research, provide a mapping from a random variable (usually the cost) to a real number. These risk metrics should satisfy certain axioms in order to be well-defined as well as to enable practical use in robotic applications [27]. Conditional value-at-risk (CVaR) is one such risk measure that has this desirable set of properties, and is a part of a class of risk metrics known as *coherent risk measures* [6]. Coherent risk measures have been used in a variety of decision

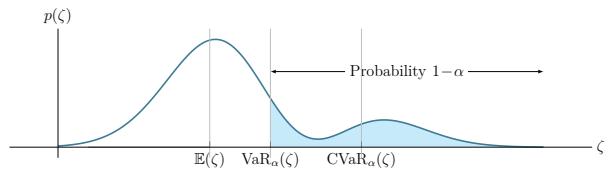


Fig. 2. Comparison of the mean, VaR, and CVaR for a given risk level $\alpha \in (0, 1]$. The axes denote the values of the stochastic variable ζ , which in our work represents traversability cost. The shaded area denotes the $(1-\alpha)\%$ of the area under $p(\zeta)$. $\text{CVaR}_\alpha(\zeta)$ is the expected value of ζ under the shaded area.

making problems, especially Markov decision processes (MDPs) [10]. In recent years, Ahmadi et al. synthesized risk averse optimal policies for partially observable MDPs, constrained MDPs, and for shortest path problems in MDPs [4, 3, 5]. Coherent risk measures have been used in a MPC framework when the system model is uncertain [36] and when the uncertainty is a result of measurement noise or moving obstacles [12]. In [20, 12], the authors incorporated risk constraints in the form of distance to the randomly moving obstacles but did not include model uncertainty. Our work extends CVaR risk to a risk-based planning framework which utilizes different sources of traversability risk (such as collision risk, step risk, slippage risk, etc.) Moreover, this paper introduces the first field-hardened and theoretically grounded approach to traversability assessment and risk-constrained planning using CVaR metrics. Using CVaR to assess traversability risks allows us to dynamically tune the entire system's behavior - from aggressive to highly conservative - by changing a single value, the risk probability level.

Model Predictive Control has a long history in controls as a means to robustly control more complex systems, including time-varying, nonlinear, or MIMO systems [9]. While simple linear PID controllers are sufficient for simpler systems, MPC is well-suited to more complex tasks while being computationally feasible. In this work, MPC is needed to handle a) complex interactions (risk constraints) between the robot and the environment, including non-linear constraints on robot orientation and slope, b) non-linear dynamics which include non-holonomic constraints, and c) non-convex, time-varying CVaR-based constraints and cost functions. In particular, we take an MPC approach known as Sequential Quadratic Programming (SQP), which iteratively solves locally quadratic sub-problems to converge to a globally (more) optimal solution [7]. Particularly in the robotics domain, this approach is well-suited due to its reduced computational costs and flexibility for handling a wide variety of costs and constraints [35, 25]. A common criticism of SQP-based MPC (and nonlinear MPC methods in general) is that they can suffer from being susceptible to local minima. We address this problem by incorporating a trajectory library (which can be predefined and/or randomly generated, e.g. as in [21]) to use in a preliminary trajectory selection process. We use this as a means to find more globally optimal initial guesses for the SQP problem to refine locally. Another common difficulty with risk-constrained nonlinear MPC problems

is ensuring recursive feasibility [26]. We bypass this problem by dynamically relaxing the severity of the risk constraints while penalizing CVaR in the cost function.

III. RISK-AWARE TRAVERSABILITY AND PLANNING

A. Problem Statement

We first give a formal definition of the problem of risk-aware traversability and motion planning. Let x_k , u_k , z_k denote the state, action, and observation at the k -th time step. A path $x_{0:N} = \{x_0, x_1, \dots, x_N\}$ is composed of a sequence of poses. A policy is a mapping from state to control $u = \pi(x)$. A map is represented as $m = (m^{(1)}, m^{(2)}, \dots)$ where m^i is the i -th element of the map (e.g., a cell in a grid map). The robot's dynamics model captures the physical properties of the vehicle's motion, such as inertia, mass, dimension, shape, and kinematic and control constraints:

$$x_{k+1} = f(x_k, u_k) \quad (1)$$

$$g(u_k) \succsim 0 \quad (2)$$

where $g(u_k)$ is a vector-valued function which encodes control constraints/limits.

Following [32], we define *traversability* as the capability for a ground vehicle to reside over a terrain region under an admissible state. We represent traversability as a cost, i.e. a continuous value computed using a terrain model, the robotic vehicle model, and kinematic constraints, which represents the degree to which we wish the robot to avoid a given state:

$$r = \mathcal{R}(m, x, u) \quad (3)$$

where $r \in \mathbb{R}$, and $\mathcal{R}(\cdot)$ is a traversability assessment model. This model captures various unfavorable events such as collision, getting stuck, tipping over, high slippage, to name a few. Each mobility platform has its own assessment model to reflect its mobility capability.

Associated with the true traversability value is a distribution over possible values based on the current understanding about the environment and robot actions. In most real-world applications where perception capabilities are limited, the true value can be highly uncertain. To handle this uncertainty, consider a map belief, i.e., a probability distribution $p(m|x_{0:k}, z_{0:k})$, over a possible set \mathcal{M} . Then, the traversability estimate is also represented as a random variable $R: (\mathcal{M} \times \mathcal{X} \times \mathcal{U}) \rightarrow \mathbb{R}$. We call this probabilistic mapping from map belief, state, and controls to possible traversability cost values a *risk assessment model*.

A risk metric $\rho(R): R \rightarrow \mathbb{R}$ is a mapping from a random variable to a real number which quantifies some notion of risk. In order to assess the risk of traversing along a path $x_{0:N}$ with a policy π , we wish to define the cumulative risk metric associated with the path, $J(x_0, \pi)$. To do this, we need to evaluate a sequence of random variables $R_{0:N}$. To quantify the stochastic outcome as a real number, we use the dynamic, time-consistent risk metric given by compounding the one-step risk metrics [34]:

$$J(x_0, \pi; m) = R_0 + \rho_0(R_1 + \rho_1(R_2 + \dots + \rho_{N-1}(R_N))) \quad (4)$$

where $\rho_k(\cdot)$ is a one-step coherent risk metric at time k . This one-step risk gives us the cost incurred at time-step $k+1$ from the perspective of time-step k . Any distortion risk metric compounded as given in (4) is time-consistent (see [27] for more information on distortion risk metrics and time-consistency). We use the Conditional Value-at-Risk (CVaR) as the one-step risk metric:

$$\rho(R) = \text{CVaR}_\alpha(R) = \inf_{z \in \mathbb{R}} \mathbb{E} \left[z + \frac{(R-z)_+}{1-\alpha} \right] \quad (5)$$

where $(\cdot)_+ = \max(\cdot, 0)$, and $\alpha \in (0, 1]$ denotes the *risk probability level*.

We formulate the objective of the problem as follows: Given the initial robot configuration x_S and the goal configuration x_G , find an optimal control policy π^* that moves the robot from x_S to x_G while 1) minimizing time to traverse, 2) minimizing the cumulative risk metric along the path, and 3) satisfying all kinematic and dynamic constraints.

B. Hierarchical Risk-Aware Planning

We propose a hierarchical approach to address the aforementioned risk-aware motion planning problem by splitting the motion planning problem into geometric and kinodynamic domains. We consider the geometric domain over long horizons, while we solve the kinodynamic problem over a shorter horizon. This is convenient for several reasons: 1) Solving the full constrained CVaR minimization problem over long timescales/horizons becomes intractable in real-time. 2) Geometric constraints play a much larger role over long horizons, while kinodynamic constraints play a much larger role over short horizons (to ensure dynamic feasibility at each timestep). 3) A good estimate (upper bound) of risk can be obtained by considering position information only. This is done by constructing a position-based traversability model \mathcal{R}_{pos} by marginalizing out non-position related variables from the risk assessment model, i.e. if the state $x = [p_x, p_y, x_{\text{other}}]^\top$ consists of position and non-position variables (e.g. orientation, velocity), then

$$\mathcal{R}_{\text{pos}}(m, p_x, p_y) \geq \mathcal{R}(m, x, u) \quad \forall x_{\text{other}}, u \quad (6)$$

Geometric Planning: The objective of geometric planning is to search for an *optimistic* risk-minimizing path, i.e. a path that minimizes an upper bound approximation of the true CVaR value. For efficiency, we limit the search space only to the geometric domain. We are searching for a sequence of poses $x_{0:N}$ which ends at x_G and minimizes the position-only risk metric in (4), which we define as $J_{\text{pos}}(x_{0:N})$. The optimization problem can be written as:

$$x_{0:N}^* = \operatorname{argmin}_{x_{0:N}} \left[J_{\text{pos}}(x_{0:N}) + \lambda \sum_{k=0}^{N-1} \|x_k - x_{k+1}\|^2 \right] \quad (7)$$

$$\text{s.t. } \phi(m, x_k) \succsim 0 \quad (8)$$

where the constraints $\phi(\cdot)$ encode position-dependent traversability constraints (e.g. constraining the vehicle to prohibit lethal levels of risk) and $\lambda \in \mathbb{R}$ weighs the tradeoff between risk and path length.

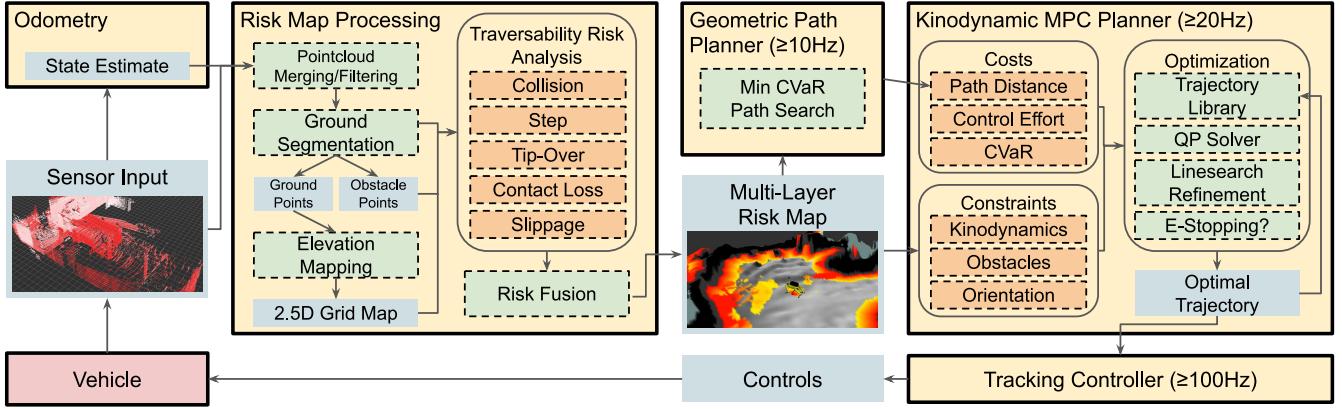


Fig. 3. Overview of system architecture for STEP. From left to right: Odometry aggregates sensor inputs and relative poses. Next, Risk Map Processing merges these pointclouds and creates a multi-layer risk map. The map is used by the Geometric Path Planner and the Kinodynamic MPC Planner. An optimal trajectory is found and sent to the Tracking Controller, which produces control inputs to the robot.

Kinodynamic Planning: We then solve a kinodynamic planning problem to track the optimal geometric path, minimize the risk metric, and respect kinematic and dynamics constraints. The goal is to find a control policy π^* within a local planning horizon $T \leq N$ which tracks the path $X_{0:N}^*$. The optimal policy can be obtained by solving the following optimization problem:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmin}} \left[J(x_0, \pi) + \lambda \sum_{k=0}^T \|x_k - x_k^*\|^2 \right] \quad (9)$$

$$s.t. \forall k \in [0, \dots, T]: \quad x_{k+1} = f(x_k, u_k) \quad (10)$$

$$g(u_k) \succ 0 \quad (11)$$

$$h(m, x_k) \succ 0 \quad (12)$$

where the constraints $g(u)$ and $h(m, x_k)$ are vector-valued functions which encode controller limits and state constraints, respectively.

IV. STEP FOR UNSTRUCTURED TERRAIN

Having outlined our approach for solving the constrained CVaR minimization problem, in this section we discuss how we compute traversability risk and efficiently solve the risk-aware trajectory optimization problem. At a high level, our approach takes the following steps (see Figure 3): 1) Assuming some source of localization with uncertainty, aggregate sensor measurements to create an uncertainty-aware map. 2) Perform ground segmentation to isolate the parts of the map the robot can potentially traverse. 3) Compute risk and risk uncertainty using geometric properties of the pointcloud (optionally, include other sources of risk, e.g. semantic or other sensors). 4) Aggregate these risks to compute a 2.5D CVaR risk map. 5) Solve for an optimistic CVaR minimizing path over long ranges with a geometric path planner. 7) Solve for a kinodynamically feasible trajectory which minimizes CVaR while staying close to the geometric path and satisfying all constraints.

A. Modeling Assumptions

Among many representation options for rough terrain, we use a 2.5D grid map in this paper for its efficiency in

processing and data storage [16]. The map is represented as a collection of terrain properties (e.g., height, risk) over a uniform grid.

For different vehicles we use different robot dynamics models. For example, for a system which produces longitudinal/lateral velocity and steering (e.g. legged platforms), the state and controls can be specified as:

$$x = [p_x, p_y, p_\theta, v_x, v_y, v_\theta]^\top \quad (13)$$

$$u = [a_x, a_y, a_\theta]^\top \quad (14)$$

While the dynamics $x_{k+1} = f(x_k, u_k)$ can be written as $x_{k+1} = x_k + \Delta t \Delta x_k$, where $\Delta x_k = [v_x \cos(p_\theta) - v_y \sin(p_\theta), v_x \sin(p_\theta) + v_y \cos(p_\theta), \kappa v_x + (1 - \kappa)v_\theta, a_x, a_y, a_\theta]^\top$. We let $\kappa \in [0, 1]$ be a constant which adjusts the amount of turning-in-place the vehicle is permitted. In differential drive or ackermann steered vehicles we can remove the lateral velocity component of these dynamics if desired. However, our general approach is applicable to any vehicle dynamics model. (For differential drive model, see Appendix A)

B. Traversability Assessment Models

The traversability cost is assessed as the combination of multiple risk factors. These factors are designed to capture potential hazards for the target robot in the specific environment (Figure 4). Such factors include:

- *Collision*: quantified by the distance to the closest obstacle point.
- *Step size*: the height gap between adjacent cells in the grid map. Negative obstacles can also be detected by checking the lack of measurement points in a cell.
- *Tip-over*: a function of slope angles and the robot's orientation.
- *Contact Loss*: insufficient contact with the ground, evaluated by plane-fit residuals.
- *Slippage*: quantified by geometry and the surface material of the ground.
- *Sensor Uncertainty*: sensor and localization error increase the variance of traversability estimates.

To efficiently compute the CVaR traversability cost for $l > 1$ risk factors, we assume each risk factor R_l is an in-

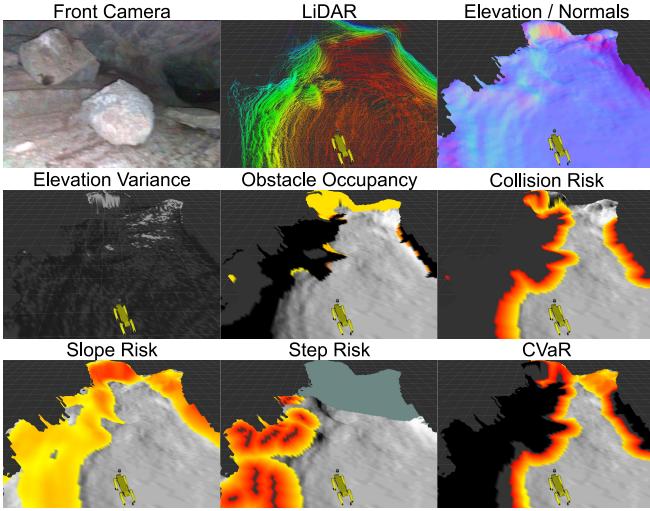


Fig. 4. Multi-layer traversability risk analysis, which first aggregates recent pointclouds (top). Then, each type of analysis (slope, step, collision, etc.) generates a risk map along with uncertainties (middle rows). These risks are aggregated to compute the final CVaR map (bottom).

dependent random variable which is normally distributed, with mean μ_l and variance σ_l . We take a weighted average of the risk factors, $R = \sum_l w_l R_l$, which will also be normally distributed as $R \sim \mathcal{N}(\mu, \sigma^2)$. Let φ and Φ denote the probability density function and cumulative distribution function of a standard normal distribution respectively. The corresponding CVaR is computed as:

$$\rho(R) = \mu + \sigma \frac{\varphi(\Phi^{-1}(\alpha))}{1-\alpha} \quad (15)$$

We construct R such that the expectation of R is positive, to keep the CVaR value positive.

Construction of the mean and variance of each risk factor depends on the type of risk. For example, collision risk is determined by checking for points above the elevation map, and the variance is derived from the elevation map variance, which is mainly a function of localization error. In contrast, negative obstacle risk is determined by looking for gaps in sensor measurements. These gaps tend to be a function of sensor sparsity, so the risk variance increases with distance from the sensor frame.

C. Risk-aware Geometric Planning

In order to optimize (7) and (8), the geometric planner computes an optimal path that minimizes the position-dependent dynamic risk metric in (4) along the path. Substituting (15) into (4), we obtain:

$$J_{\text{pos}}(x_{0:N}) = \mu_0 + \sum_{k=1}^N \left[\mu_k + \sigma_k \frac{\varphi(\Phi^{-1}(\alpha))}{1-\alpha} \right] \quad (16)$$

(For a proof, see Appendix B.)

We use the A* algorithm to solve (7) over a 2D grid. A* requires a path cost $g(n)$ and a heuristic cost $h(n)$, given by:

$$g(n) = J_{\text{pos}}(x_{0:n}) + \lambda \sum_{k=0}^{n-1} \|x_k - x_{k+1}\|^2 \quad (17)$$

$$h(n) = \lambda \|x_n - x_G\|^2 \quad (18)$$

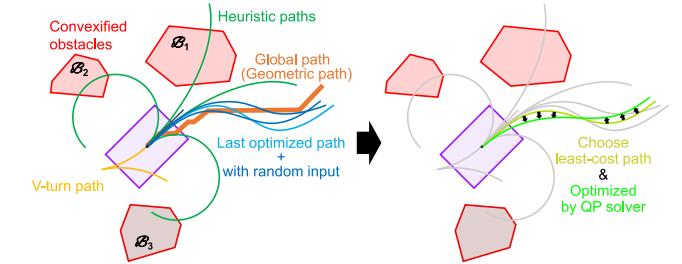


Fig. 5. Diagram of kinodynamic MPC planner, which begins with evaluating various paths within a trajectory library. The lowest cost path is chosen as a candidate and optimized by the QP solver.

Algorithm 1 Kinodynamic MPC Planner (sequences $\{\text{var}_k\}_{k=0:T}$ are expressed as $\{\text{var}\}$ for brevity)

Input: current state x_0 , current control sequence (previous solution) $\{u^*\}^{(j)}$
Output: re-planned trajectory $\{x^*\}^{(j+1)}$, re-planned control sequence $\{u^*\}^{(j+1)}$

Initialization

- 1: $\{x^r\} = \text{updateReferenceTrajectory}()$
- 2: $\{u^*\}^{(j)} = \text{stepControlSequenceForward}(\{u^*\}^{(j)})$
- Loop process*
- 3: **for** $i=0$ to qp_iterations **do**
- 4: $l = \text{generateTrajectoryLibrary}(x_0)$
- 5: $\{[x^c], \{u^c\}\} = \text{chooseCandidateFromLibrary}(l)$
- 6: $\{[\delta x^*], \{\delta u^*\}\} = \text{solveQP}(\{x^c\}, \{u^c\}, \{x^r\})$
- 7: $[\gamma, \text{solved}] = \text{lineSearch}(\{x^c\}, \{\delta x^*\}, \{u^c\}, \{\delta u^*\})$
- 8: $u_k^c = u_k^c + \gamma \delta u_k^*, \forall k=0:T$
- 9: $\{x^c\} = \text{rollOutTrajectory}(x_0, \{u^c\})$
- 10: **end for**
- 11: **if** solved **then**
- 12: $\{x^*\}^{(j+1)}, \{u^*\}^{(j+1)} = \{x^c\}, \{u^c\}$
- 13: **else**
- 14: $\{x^*\}^{(j+1)}, \{u^*\}^{(j+1)} = \text{getStoppingTrajectory}()$
- 15: **end if**
- 16: **return** $\{x^*\}^{(j+1)}, \{u^*\}^{(j+1)}$

For the heuristic cost we use the shortest Euclidean distance to the goal. The value of lambda is a relative weighting between the distance penalty and risk penalty and can be thought of as having units of (traversability cost / m). We use a relatively small value, which means we are mainly concerned with minimizing traversability costs.

D. Risk-aware Kinodynamic Planning

The geometric planner produces a path, i.e. a sequence of poses. We wish to find a kinodynamically feasible trajectory which stays near this path, while satisfying all constraints and minimizing the CVaR cost. To solve (9)-(12), we use a risk-aware kinodynamic MPC planner, whose steps we outline (Figure 5, Algorithm 1, Appendix C).

Trajectory library: Our kinodynamic planner begins with selecting the best candidate trajectory from a trajectory library, which stores multiple initial control and state sequences. The selected trajectory is used as initial solution for solving a full optimization problem. The trajectory library can include: 1) the trajectory accepted in the previous planning iteration, 2) a stopping (braking) trajectory, 3) a geometric plan following trajectory, 4) heuristically defined trajectories

(including v-turns, u-turns, and varying curvatures), and 5) randomly perturbed control input sequences.

QP Optimization: Next, we construct a nonlinear optimization problem with appropriate costs and constraints (9–12). We linearize the problem about the initial solution and solve iteratively in a sequential quadratic programming (SQP) fashion [28]. Let $\{\hat{x}_k, \hat{u}_k\}_{k=0:T}$ denote an initial solution. Let $\{\delta x_k, \delta u_k\}_{k=0:T}$ denote deviation from the initial solution. We introduce the solution vector variable X :

$$X = [\delta x_0^T \dots \delta x_T^T \delta u_0^T \dots \delta u_T^T]^T \quad (19)$$

We can then write (41–44) in the form:

$$\text{minimize } \frac{1}{2} X^T P X + q^T X \quad (20)$$

$$\text{subject to } l \leq A X \leq u \quad (21)$$

where P is a positive semi-definite weight matrix, q is a vector to define the first order term in the objective function, A defines inequality constraints and l and u provide their lower and upper limit. (See Appendix E.) In the next subsection we describe these costs and constraints in detail. This is a quadratic program, which can be solved using commonly available QP solvers. In our implementation we use the OSQP solver, which is a robust and highly efficient general-purpose solver for convex QPs [37].

Linesearch: The solution to the SQP problem returns an optimized variation of the control sequence $\{\delta u_k^*\}_{k=0:T}$. We then use a linesearch procedure to determine the amount of deviation $\gamma > 0$ to add to the current candidate control policy π : $u_k = u_k + \gamma \delta u_k^*$. (See Appendix F.)

Stopping Sequence: If no good solution is found from the linesearch, we pick the lowest cost trajectory from the trajectory library with no collisions. If all trajectories are in collision, we generate an emergency stopping sequence to slow the robot as much as possible (a collision may occur, but hopefully with minimal energy).

Tracking Controller: Having found a feasible and CVaR-minimizing trajectory, we send it to a tracking controller to generate closed-loop tracking behavior at a high rate ($>100\text{Hz}$), which is specific to the robot type (e.g. a simple cascaded PID, or legged locomotive controller).

E. Optimization Costs and Constraints

Costs: Note that (9) contains the CVaR risk. To linearize this and add it to the QP matrices, we compute the Jacobian and Hessian of ρ with respect to the state x . We efficiently approximate this via numerical differentiation.

Kinodynamic constraints: Similar to the cost, we linearize (10) with respect to x and u . Depending on the dynamics model, this may be done analytically.

Control limits: We construct the function $g(u)$ in (11) to limit the range of the control inputs. For example in the 6-state dynamics case, we limit maximum accelerations: $|a_x| < a_x^{\max}$, $|a_y| < a_y^{\max}$, and $|a_\theta| < a_\theta^{\max}$.

State limits: Within $h(m, x)$ in (12), we encode velocity constraints: $|v_x| < v_x^{\max}$, $|v_y| < v_y^{\max}$, and $|v_\theta| < v_\theta^{\max}$. We also constrain the velocity of the vehicle to be less than

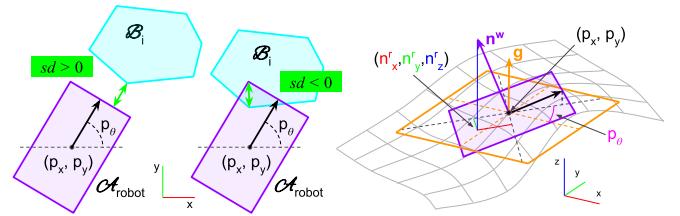


Fig. 6. Left: Computing convex to convex signed distance function between the robot footprint and an obstacle. Signed distance is positive with no intersection and negative with intersection. Right: Robot pitch and roll are computed from the surface normal rotated by the yaw of the robot. Purple rectangle is the robot footprint with surface normal n^w . g denotes gravity vector, $n_{x,y,z}^r$ are the robot-centric surface normal components used for computing pitch and roll.

some scalar multiple of the risk in that region, along with maximum allowable velocities:

$$|v_\theta| < \gamma_\theta \rho(R_k) \quad (22)$$

$$\sqrt{v_x^2 + v_y^2} < \gamma_v \rho(R_k) \quad (23)$$

This reduces the energy of interactions the robot has with its environment in riskier situations, preventing more serious damage.

Position risk constraints: Within $h(m, x_k)$ we would like to add constraints on position and orientation to prevent the robot from hitting obstacles. The general form of this constraint is:

$$\rho(R_k) < \rho^{\max} \quad (24)$$

To create this constraint, we locate areas on the map where the risk ρ is greater than the maximum allowable risk. These areas are marked as obstacles, and are highly non-convex. To obtain a convex and tractable approximation of this highly non-convex constraint, we decompose obstacles into non-overlapping 2D convex polygons, and create a signed distance function which determines the minimum distance between the robot's footprint (also a convex polygon) and each obstacle [35]. Let $\mathcal{A}, \mathcal{B} \subset \mathbb{R}^2$ be two sets, and define the distance between them as:

$$\text{dist}(\mathcal{A}, \mathcal{B}) = \inf \{ \|T\| \mid (T + \mathcal{A}) \cap \mathcal{B} \neq \emptyset \} \quad (25)$$

where T is a translation. When the two sets are overlapping, define the penetration distance as:

$$\text{penetration}(\mathcal{A}, \mathcal{B}) = \inf \{ \|T\| \mid (T + \mathcal{A}) \cap \mathcal{B} = \emptyset \} \quad (26)$$

Then we can define the signed distance between the two sets as:

$$\text{sd}(\mathcal{A}, \mathcal{B}) = \text{dist}(\mathcal{A}, \mathcal{B}) - \text{penetration}(\mathcal{A}, \mathcal{B}) \quad (27)$$

We then include within $h(m, x_k)$ a constraint to enforce the following inequality:

$$\text{sd}(\mathcal{A}_{\text{robot}}, \mathcal{B}_i) > 0 \quad \forall i \in \{0, \dots, N_{\text{obstacles}}\} \quad (28)$$

Note that the robot footprint $\mathcal{A}_{\text{robot}}$ depends on the current robot position and orientation: $\mathcal{A}_{\text{robot}}(p_x, p_y, p_\theta)$, while each obstacle $\mathcal{B}_i(m)$ is dependent on the information in the map (See Figure 6).

Orientation constraints: We wish to constrain the robot's orientation on sloped terrain in such a way as to prevent the robot from rolling over or performing dangerous maneuvers. To do this, we add constraints to $h(m, x_k)$ which limit the roll and pitch of the robot as it settles on the surface of the ground. Denote the position

as $p = [p_x, p_y]^\top$ and the position/yaw as $s = [p_x, p_y, p_\theta]^\top$. Let the robot's pitch be ψ and roll be ϕ in its body frame. Let $\omega = [\psi, \phi]^\top$. The constraint will have the form $|\omega| \prec \omega^{\max}$. At p , we compute the surface normal vector, call it $n^w = [n_x^w, n_y^w, n_z^w]^\top$, in the world frame. Let $n^r = [n_x^r, n_y^r, n_z^r]^\top$, be the surface normal in the body frame, where we rotate by the robot's yaw: $n^r = R_\theta n^w$ (see Figure 6), where R_θ is a basic rotation matrix by the angle θ about the world z axis. Then, we define the robot pitch and roll as $\omega = g(n^r)$ where:

$$\omega = g(n^r) = \begin{bmatrix} \text{atan2}(n_x^r, n_z^r) \\ -\text{atan2}(n_y^r, n_z^r) \end{bmatrix} \quad (29)$$

Note that ω is a function of s . Creating a linearly-constrained problem requires a linear approximation of the constraint:

$$|\nabla_s \omega(s) \delta s + \omega(s)| < \omega^{\max} \quad (30)$$

This is accomplished by finding the gradients with respect to position and yaw separately (See Appendix D).

Box Constraint: Note that if δx and δu are too large, linearization errors will dominate. To mitigate this we also include box constraints within (11) and (12) to maintain a bounded deviation from the initial solution: $|\delta x| < \epsilon_x$ and $|\delta u| < \epsilon_u$.

Adding Slack Variables: To further improve the feasibility of the optimization problem we introduce auxilliary slack variables for constraints on state limits, position risk, and orientation. For a given constraint $h(x) > 0$ we introduce the slack variable ϵ , and modify the constraint to be $h(x) > \epsilon$ and $\epsilon < 0$. We then penalize large slack variables with a quadratic cost: $\lambda_\epsilon \epsilon^2$. These are incorporated into the QP problem (20) and (21).

F. Dynamic Risk Adjustment

The CVaR metrics allows us to dynamically adjust the level and severity of risk we are willing to accept. Selecting low α reverts towards using the mean cost as a metric, leading to optimistic decision making while ignoring low-probability but high cost events. Conversely, selecting a high α leans towards conservatism, reducing the likelihood of fatal events while reducing the set of possible paths. We adjust α according to two criteria: 1) Mission-level states, where depending on the robot's role, or the balance of environment and robot capabilities, the risk posture for individual robots may differ. 2) Recovery Behaviors, where if the robot is trapped in an unfavorable condition, by gradually decreasing α , an escape plan can be found with minimal risk. These heuristics are especially useful in the case of risk-aware planning, because the feasibility of online nonlinear MPC is difficult to guarantee. When no feasible solution is found for a given risk level α , a riskier but feasible solution can be quickly found and executed.

V. EXPERIMENTS

In this section, we report the performance of STEP. We first present a comparative study between different adjustable risk thresholds in simulation on a wheeled differential drive platform. Then, we demonstrate

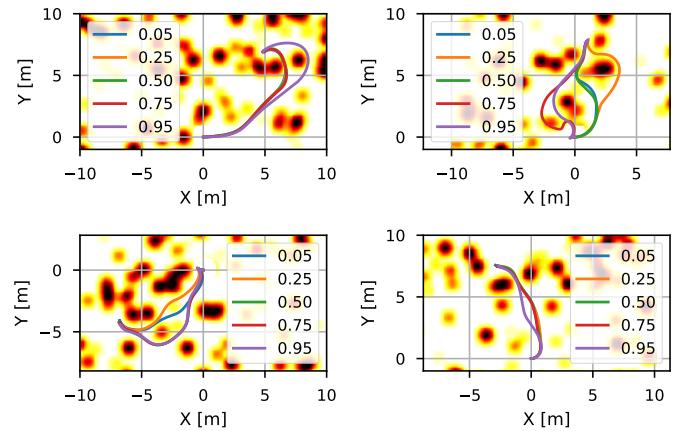


Fig. 7. Path distributions from four simulated runs. The risk level α spans from 0.1 (close to mean-value) to 0.95 (conservative). Smaller α typically results in a shorter path, while larger α chooses statistically safe paths.

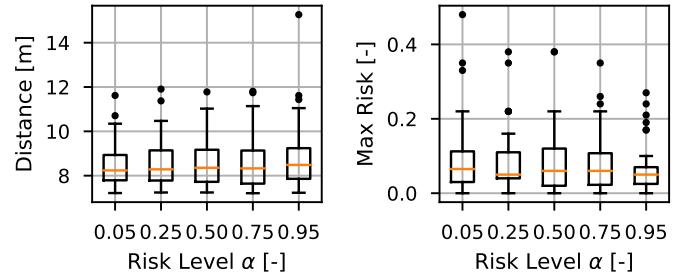


Fig. 8. Distance vs risk trade-off from 50 Monte-Carlo simulations. Left: Distributions of path distance. Right: Distributions of max risk along the traversed paths. Box plot uses standard quartile format and dots are outliers.

real-world performance using a wheeled robot deployed in an abandoned subway filled with clutter, and a legged platform deployed in a lava tube environment.

A. Simulation Study

To assess statistical performance, we perform 50 Monte-Carlo simulations with randomly generated maps and goals. Random traversability costs are assigned to each grid cell. The following assumptions are made: 1) no localization error, 2) no tracking error, and 3) a simplified perception model with artificial noise. We give a random goal 8 m away and evaluate the path cost and distance. We use a differential-drive dynamics model (no lateral velocity).

We compare STEP using different α levels. Figure 7 shows the distribution of paths for different planning configurations. The optimistic (close to mean-value) planner $\alpha=0.05$ typically generates shorter paths, while the conservative setting $\alpha=0.95$ makes long detours to select statistically safer paths. The other α settings show distributions between these two extremes, with larger α generating similar paths to the conservative planner and smaller α generating more time-optimal paths. Statistics are shown in Figure 8.

B. Hardware Results

We deployed STEP on two different robots (wheeled and legged) in two different challenging environments (an abandoned subway and a lava tube). First we tested STEP

on a Clearpath Husky robot in an abandoned subway filled with industrial clutter. The robot was equipped with custom sensing and computing units, and driven by JPL’s NeBula autonomy software [1]. 3 Velodyne VLP-16s were used for collecting LiDAR data. Localization was provided onboard by a LIDAR-based SLAM solution [31, 13]. The entire autonomy stack runs on an Intel Core i7 CPU. The typical CPU usage for the traversability stack is about a single core. The robot successfully explored two levels of the approximately 200m x 100m environment. In Figure 9, we plot the risk map in this environment at varying levels of α . We clearly see the effects on the risk map, where higher values of α result in closing narrow openings, assigning high cost to unknown regions, and increasing the size of obstacles. The effect of these risk analyses results in intuitive outcomes - for example, a low pile of metal, while probably traversable, should be avoided if possible. When the region has inadequate sensor coverage, the risk will be high. When the robot is closer and the sensor coverage is good, then the CVaR cost will decrease, yielding a more accurate risk assessment. This results in more efficient and safer planning when compared to deterministic methods. For example, in our prior work [38], the deterministic approach led to frequent oscillations in planning as obstacles appeared and disappeared with sensor and localization noise.

Next, we deployed STEP on a Boston Dynamics Spot quadruped robot at the Valentine Cave in Lava Beds National Monument, Tulelake, CA. The main sensor for localization and traversability analysis is a Velodyne VLP-16, fused with Spot’s internal Intel realsense data to cover blind spots. The payload was similar to that of Husky, proving on-board, real-time computing.

Figure 10 shows the interior of the cave and algorithm’s representations. The rough ground surface, rounded walls, ancient lava waterfalls, steep non-uniform slopes, and boulders all pose significant traversability stresses. Furthermore, there are many occluded places which affect the confidence in traversability estimates.

We tested our risk-aware traversability software during our fully autonomous runs. The planner was able to navigate the robot safely to the every goal provided by the upper-layer coverage planner [8, 23] despite the challenges posed by the environment. Figure 10 shows snapshots of elevation maps, CVaR risk maps, and planned paths. The risk map captures walls, rocks, high slopes, and ground roughness as mobility risks. STEP enables Spot to safely traverse the entire extent of the lava tube, fully exploring all regions. STEP navigates 420 meters over 24 minutes, covering 1205 square meters of rough terrain.

VI. CONCLUSION

We have presented STEP (Stochastic Traversability Evaluation and Planning), our approach for autonomous robotic navigation in unsafe, unstructured, and unknown environments. We believe this approach finds a sweet-spot between computation, resiliency, performance, and

flexibility when compared to other motion planning approaches in such extreme environments. Our method is generalizable and extensible to a wide range of robot types, sizes, and speeds, as well as a wide range of environments. Our future work includes robustification of subcomponents and extension to higher speeds.

ACKNOWLEDGEMENT

The work is performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004), and Defense Advanced Research Projects Agency (DARPA).

REFERENCES

- [1] A. Agha-mohammadi, et al. “NeBula: Quest for robotic autonomy in challenging environments; TEAM CoSTAR at the DARPA Subterranean Challenge”. In: *Field Robotics* (2021).
- [2] A. Agha-Mohammadi et al. “Confidence-rich grid mapping”. In: *The International Journal of Robotics Research* 38.12-13 (2017), pp. 1352–1374.
- [3] Mohamadreza Ahmadi et al. “Constrained Risk-Averse Markov Decision Processes”. In: *AAAI Conference on Artificial Intelligence*. 2021.
- [4] Mohamadreza Ahmadi et al. “Risk-Averse Planning Under Uncertainty”. In: *American Control Conference*. 2020, pp. 3305–3312.
- [5] Mohamadreza Ahmadi et al. “Risk-Averse Stochastic Shortest Path Planning”. In: *arXiv:2103.14727* (2021).
- [6] Philippe Artzner et al. “Coherent measures of risk”. In: *Mathematical finance* 9.3 (1999), pp. 203–228.
- [7] Paul T Boggs and Jon W Tolle. “Sequential quadratic programming”. In: *Acta numerica* (1995), pp. 529–562.
- [8] Amanda Bouman et al. “Autonomous Spot: Long-range autonomous exploration of extreme environments with legged locomotion”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2020).
- [9] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.
- [10] Yinlam Chow et al. “Risk-sensitive and robust decision-making: a CVaR optimization approach”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 1522–1530.
- [11] Will Dabney et al. “Distributional reinforcement learning with quantile regression”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
- [12] Anushri Dixit, Mohamadreza Ahmadi, and Joel W. Burdick. “Risk-Sensitive Motion Planning using Entropic Value-at-Risk”. In: *European Control Conference*. 2021.
- [13] K Ebadi et al. “LAMP: Large-scale autonomous mapping and positioning for exploration of



Fig. 9. Traversability analysis results for Husky in an abandoned subway experiment. Top left to right: Risk maps at three varying risk levels: $\alpha = 0.1, 0.5, 0.9$, respectively. Colors correspond to CVaR value (white: safe ($r \leq 0.05$), yellow to red: moderate ($0.05 < r \leq 0.5$), black: risky ($r > 0.5$)). Also shown are the most recent LiDAR measurements (green points). Bottom left and middle: Front and right on-board cameras observing the same location. Bottom right: Completed top-down map of the environment after autonomous exploration. Bright dots are pillars, which are visible in the camera images.

- perceptually-degraded subterranean environments”. In: *IEEE International Conference on Robotics and Automation*. 2020, pp. 80–86.
- [14] David D Fan, Ali-akbar Agha-mohammadi, and Evangelos A Theodorou. “Deep learning tubes for Tube MPC”. In: *Robotics: Science and Systems (RSS)* (2020).
- [15] Péter Fankhauser, Michael Bloesch, and Marco Hutter. “Probabilistic Terrain Mapping for Mobile Robots With Uncertain Localization”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3019–3026.
- [16] Péter Fankhauser and Marco Hutter. “A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation”. In: *Robot Operating System (ROS), The Complete Reference* (2016), pp. 99–120.
- [17] S. Ghosh, K. Otsu, and M. Ono. “Probabilistic Kinematic State Estimation for Motion Planning of Planetary Rovers”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2018, pp. 5148–5154.
- [18] Steven B Goldberg, Mark W Maimone, and Larry Matthies. “Stereo vision and rover navigation software for planetary exploration”. In: *IEEE Aerospace Conference*. IEEE. 2002.
- [19] Alain Haït, Thierry Simeon, and Michel Taïx. “Algorithms for rough terrain trajectory planning”. In: *Advanced Robotics* 16.8 (2002), pp. 673–699.
- [20] Astghik Hakobyan, Gyeong Chan Kim, and Insoon Yang. “Risk-aware motion planning and control using CVaR-constrained optimization”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3924–3931.
- [21] Mrinal Kalakrishnan et al. “STOMP: Stochastic trajectory optimization for motion planning”. In: *IEEE International Conference on Robotics and Automation*. 2011, pp. 4569–4574.
- [22] Himangshu Kalita et al. “Path planning and navigation inside off-world lava tubes and caves”. In: *IEEE/ION Position, Location and Navigation Symposium*. 2018, pp. 1311–1318.
- [23] Sung-Kyun Kim et al. “PLGRIM: Hierarchical value learning for large-scale exploration in unknown environments”. In: *International Conference on Automated Planning and Scheduling (ICAPS)*. Vol. 31. 2021.
- [24] Sven Koenig and Reid G Simmons. “Risk-sensitive planning with probabilistic decision graphs”. In: *Principles of Knowledge Representation and Reasoning*. Elsevier. 1994, pp. 363–373.
- [25] Thomas Lew, Riccardo Bonalli, and Marco Pavone. “Chance-constrained sequential convex programming for robust trajectory optimization”. In: *2020 European Control Conference (ECC)*. IEEE. 2020, pp. 1871–1878.
- [26] Johan Löfberg. “Oops! I cannot do it again: Testing for recursive feasibility in MPC”. In: *Automatica* 48.3 (2012), pp. 550–555.
- [27] Anirudha Majumdar and Marco Pavone. “How should a robot assess risk? Towards an axiomatic theory of risk in robotics”. In: *Robotics Research*. Springer, 2020, pp. 75–84.
- [28] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
- [29] Masahiro Ono et al. “Chance-constrained dynamic programming with application to risk-aware robotic space exploration”. In: *Autonomous Robots* 39.4 (2015), pp. 555–571.
- [30] Kyohei Otsu et al. “Fast approximate clearance evaluation for rovers with articulated suspension systems”. In: *Journal of Field Robotics* 37 (5 2020), pp. 768–785.

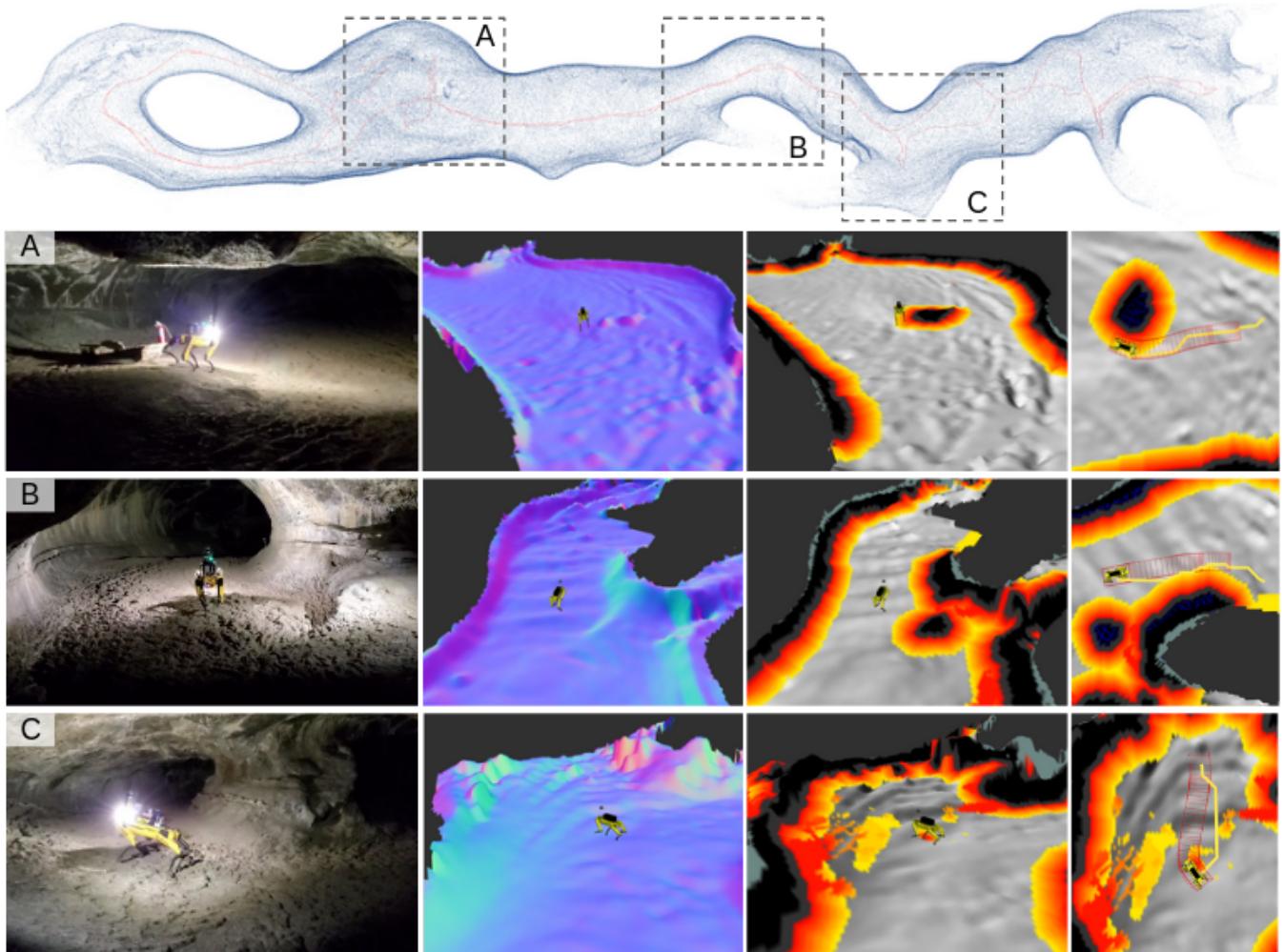


Fig. 10. Traversability analysis results for the Valentine Cave experiment. From left to right: Third-person view, elevation map (colored by normal direction), risk map (colored by risk level. white: safe ($r \leq 0.05$), yellow to red: moderate ($0.05 < r \leq 0.5$), and black: risky ($r > 0.5$)), and planned geometric/kinodynamic paths (yellow lines/red boxes).

- [31] Matteo Palieri et al. “LOCUS: A multi-sensor lidar-centric solution for high-precision odometry and 3D mapping in real-time”. In: *IEEE Robotics and Automation Letters* 6.2 (2020), pp. 421–428.
- [32] Panagiotis Papadakis. “Terrain traversability analysis methods for unmanned ground vehicles: A survey”. In: *Engineering Applications of Artificial Intelligence* 26.4 (2013), pp. 1373–1385.
- [33] R Tyrrell Rockafellar, Stanislav Uryasev, et al. “Optimization of conditional value-at-risk”. In: *Journal of Risk* 2.3 (2000), pp. 21–41.
- [34] Andrzej Ruszczynski. “Risk-averse dynamic programming for Markov decision processes”. In: *Mathematical Programming* 75.2 (2014), pp. 235–261.
- [35] John Schulman et al. “Motion planning with sequential convex optimization and convex collision checking”. In: *The International Journal of Robotics Research* 33.9 (2014), pp. 1251–1270.
- [36] Sumeet Singh et al. “A Framework for Time-Consistent, Risk-Sensitive Model Predictive Control: Theory and Algorithms”. In: *IEEE Transactions on Automatic Control* 64.7 (2018), pp. 2905–2912.
- [37] B. Stellato et al. “OSQP: an operator splitting solver for quadratic programs”. In: *Mathematical Programming Computation* (2020), pp. 1–36.
- [38] Rohan Thakker et al. “Autonomous off-road navigation over extreme terrains with perceptually-challenging conditions”. In: *International Symposium on Experimental Robotics* (2020).
- [39] Allen Wang, Ashkan Jasour, and Brian Williams. “Non-gaussian chance-constrained trajectory planning for autonomous vehicles under agent uncertainty”. In: *Robotics and Automation Letters* 5.4 (2020), pp. 6041–6048.
- [40] Huan Xu and Shie Mannor. “Distributionally robust Markov decision processes”. In: *Advances in Neural Information Processing Systems*. 2010, pp. 2505–2513.

APPENDIX

A. Dynamics model for differential drive

For a simple system which produces forward velocity and steering, (e.g. differential drive systems), we may wish to specify the state and controls as:

$$x = [p_x, p_y, p_\theta, v_x]^\top \quad (31)$$

$$u = [a_x, v_\theta]^\top \quad (32)$$

For example, the dynamics $x_{k+1} = f(x_k, u_k)$ for a simple differential-drive system can be written as:

$$x_{k+1} = x_k + \Delta t \begin{bmatrix} v_x \cos(p_\theta) \\ v_x \sin(p_\theta) \\ \gamma v_x + (1-\gamma)v_\theta \\ a_x \end{bmatrix} \quad (33)$$

where $\gamma \in [0, 1]$ is a constant which adjusts the amount of turning-in-place the vehicle is permitted.

B. Computing the dynamic risk metric using CVaR for Normal distributions

$$\begin{aligned} J(x_0, \pi) &= R_0 + \rho_0(R_1 + \rho_1(R_2 + \dots + \rho_{T-1}(R_T))) \\ &= R_0 + \rho \left(R_1 + \rho \left(R_2 + \dots + \rho(R_{T-1} + \right. \right. \\ &\quad \left. \left. \mu_T + \sigma_T \frac{\varphi(\Phi^{-1}(\alpha))}{1-\alpha} \right) \right) \right) \\ &= R_0 + \rho \left(R_1 + \rho \left(R_2 + \dots + \rho(R_{T-2} + \right. \right. \\ &\quad \left. \left. \mu_{T-1} + \mu_T + (\sigma_{T-1} + \sigma_T) \frac{\varphi(\Phi^{-1}(\alpha))}{1-\alpha} \right) \right) \right) \\ &\quad \vdots \\ &= R_0 + \sum_{i=1}^T \left(\mu_i + \sigma_i \frac{\varphi(\Phi^{-1}(\alpha))}{1-\alpha} \right) \\ &= \sum_{i=0}^T \rho(R_i) \end{aligned}$$

C. Kinodynamic Planning Diagram

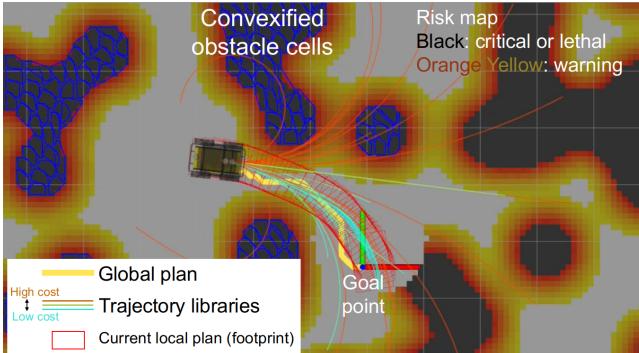


Fig. 11. Diagram of kinodynamic MPC planner, which begins with evaluating various paths within a trajectory library. The lowest cost path is chosen as a candidate and optimized by the QP solver.

D. Gradients for Orientation Constraint

We describe in further detail the derivation of the orientation constraints. Denote the position as $p = [p_x, p_y]^\top$ and the position/yaw as $s = [p_x, p_y, p_\theta]$. We wish to find the robot's pitch ψ and roll ϕ in its body frame. Let $\omega = [\psi, \phi]^\top$. The constraint will have the form $|\omega(s)| \leq \omega_{max}$. At p , we compute the surface normal vector, call it $n^w = [n_x^w, n_y^w, n_z^w]^\top$, in the world frame. To convert the normal vector in the body frame, $n^r = [n_x^r, n_y^r, n_z^r]^\top$, we rotate by the robot's yaw: $n^r = R_\theta n^w$ (see Figure 6), where R_θ is a basic rotation matrix by the angle θ about the world z axis:

$$R_\theta = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (34)$$

Let the robot pitch and roll vector ω be defined as $\omega = g(n^r)$, where:

$$\omega = g(n^r) = \begin{bmatrix} \text{atan}2(n_x^r, n_z^r) \\ -\text{atan}2(n_y^r, n_z^r) \end{bmatrix} \quad (35)$$

Creating a linearly-constrained problem requires a linear approximation of the constraint:

$$|\nabla_s \omega(s) \delta s + \omega(s)| \leq \omega_{max} \quad (36)$$

Conveniently, computing $\nabla_s \omega(s)$ reduces to finding gradients w.r.t position and yaw separately. Let $\nabla_s \omega(s) = [\nabla_p \omega(s), \nabla_\theta \omega(s)]^\top$, then:

$$\nabla_p \omega(s) = (\nabla_{n^r} g)(R_\theta)(\nabla_p n^w) \quad (37)$$

$$\nabla_\theta \omega(s) = (\nabla_{n^r} g)\left(\frac{d}{d\theta} R_\theta\right)(n^w) \quad (38)$$

where:

$$\nabla_{n^r} g = \begin{bmatrix} \frac{n_z^r}{(n_x^r)^2 + (n_z^r)^2} & 0 & \frac{-n_x^r}{(n_x^r)^2 + (n_z^r)^2} \\ 0 & \frac{-n_z^r}{(n_y^r)^2 + (n_z^r)^2} & \frac{n_y^r}{(n_y^r)^2 + (n_z^r)^2} \end{bmatrix} \quad (39)$$

and

$$\nabla_p n^w = \begin{bmatrix} \frac{\partial n_x^w}{\partial p_x} & \frac{\partial n_x^w}{\partial p_y} \\ \frac{\partial n_y^w}{\partial p_x} & \frac{\partial n_y^w}{\partial p_y} \\ \frac{\partial n_z^w}{\partial p_x} & \frac{\partial n_z^w}{\partial p_y} \end{bmatrix} \quad (40)$$

The terms with the form $\frac{\partial n_x^w}{\partial p_x}$ amount to computing a second-order gradient of the elevation on the 2.5D map. This can be done efficiently with numerical methods [16].

E. Converting non-linear MPC problem to a QP problem

Our MPC problem stated in Equations (9-12) is non-linear. In order to efficiently find a solution we linearize the problem about an initial solution, and solve iteratively, in a sequential quadratic programming (SQP) fashion [28]. Let $\{\hat{x}_k, \hat{u}_k\}_{k=0, \dots, T}$ denote an initial solution. Let $\{\delta x_k, \delta u_k\}_{k=0, \dots, T}$ denote deviation from the initial solution. We approximate (9-12) by a problem with quadratic costs and linear constraints with respect to $\{\delta x, \delta u\}$:

$$\begin{aligned} \{\delta x^*, \delta u^*\} &= \underset{\delta x, \delta u}{\operatorname{argmin}} \sum_{k=0}^T \|\hat{x}_k + \delta x_k - x_k^*\|_{Q_k} \\ &\quad + \lambda J(\hat{x}_k + \delta x_k, \hat{u}_k + \delta u_k) \quad (41) \\ \text{s.t. } & \forall k \in [0, \dots, T]: \end{aligned}$$

$$\hat{x}_{k+1} + \delta x_{k+1} = f(\hat{x}_k, \hat{u}_k) + \nabla_x f \cdot \delta x_k + \nabla_u f \cdot \delta u_k \quad (42)$$

$$g(\hat{u}_k) + \nabla_u g \cdot \delta u_k \succ 0 \quad (43)$$

$$h(m, \hat{x}_k) + \nabla_x h \cdot \delta x_k \succ 0 \quad (44)$$

where $J(\hat{x}_k + \delta x_k, \hat{u}_k + \delta u_k)$ can be approximated with a second-order Taylor approximation (for now, assume no dependence on controls):

$$J(\hat{x} + \delta x) \approx J(\hat{x}) + \nabla_x J \cdot \delta x + \delta x^\top H(J) \delta x \quad (45)$$

and $H(\cdot)$ denotes the Hessian. The problem is now a quadratic program (QP) with quadratic costs and linear constraints. To solve Equations (41-44), we introduce the solution vector variable X :

$$X = [\delta x_0^\top \dots \delta x_T^\top \delta u_0^\top \dots \delta u_T^\top]^\top \quad (46)$$

We can then write Equations (41-44) in the form:

$$\text{minimize } \frac{1}{2} X^\top P X + q^\top X \quad (47)$$

$$\text{subject to } l \leq A X \leq u \quad (48)$$

where P is a positive semi-definite weight matrix, q is a vector to define the first order term in the objective function, A defines inequality constraints and l and u provide their lower and upper limit.

F. Linesearch Algorithm for SQP solution refinement

The solution to the SQP problem returns an optimized control sequence $\{u_k^*\}_{k=0:T}$. We then use a linesearch routine to find an appropriate correction coefficient γ , using Algorithm 2. The resulting correction coefficient is carried over into the next path-planning loop.

Algorithm 2 Linesearch Algorithm

Input: candidate control sequence $\{u_k^c\}_{k=0:T}$, QP solution $\{\delta u_k^*\}_{k=0:T}$

Output: correction coefficient γ

Initialization

- 1: initialize γ by default value or last-used value
- 2: $[c, o] = \text{getCostAndObstacles}(\{u_k^c\}_{k=0:T})$

Linesearch Loop

- 3: **for** $i=0$ to max_iteration **do**
 - 4: **for** $k=0$ to T **do**
 - 5: $u_k^{c(i)} = u_k^c + \gamma \delta u_k^*$
 - 6: **end for**
 - 7: $[c^{(i)}, o^{(i)}] = \text{getCostAndObstacles}(\{u_k^{c(i)}\}_{k=0:T})$
 - 8: **if** ($c^{(i)} \leq c$ and $o^{(i)} \leq o$) **then**
 - 9: $\gamma = \min(2\gamma, \gamma_{max})$
 - 10: **break**
 - 11: **else**
 - 12: $\gamma = \max(\gamma/2, \gamma_{min})$
 - 13: **end if**
 - 14: **end for**
 - 15: **return** γ
-