2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Bi-level Trajectory Optimization on Uneven Terrains with Differentiable Wheel-Terrain Interaction Model.

Amith Manoharan[1], Aditya Sharma[2], Himani Belsare[2], Kaustab Pal[2],
K. Madhava Krishna[2], and Arun Kumar Singh[1]

***Abstract*— Navigation of wheeled vehicles on uneven terrain necessitates going beyond the 2D approaches for trajectory planning. Specifically, it is essential to incorporate the full $6dof$ variation of vehicle pose and its associated stability cost in the planning process. To this end, most recent works aim to learn a neural network model to predict vehicle evolution. However, such approaches are data-intensive and fraught with generalization issues.**

In this paper, we present a purely model-based approach that just requires the digital elevation information of the terrain. Specifically, we express the wheel-terrain interaction and $6dof$ pose prediction as a non-linear least squares (NLS) problem. As a result, trajectory planning can be viewed as a bi-level optimization. The inner optimization layer predicts the pose on the terrain along a given trajectory, while the outer layer deforms the trajectory itself to reduce the stability and kinematic costs of the pose.

We improve the state-of-the-art in the following respects. First, we show that our NLS-based pose prediction closely matches the output of a high-fidelity physics engine. This result, coupled with the fact that we can query gradients of the NLS solver, makes our pose predictor a differentiable wheel-terrain interaction model. We further leverage this differentiability to efficiently solve the proposed bi-level trajectory optimization problem. Finally, we perform extensive experiments and comparisons with a baseline to showcase the effectiveness of our approach in obtaining smooth, stable trajectories.

I. Introduction

Applications like forestry, construction, and search and rescue require wheeled vehicles to navigate over uneven terrains. Thus, for automation in these use cases, we need a trajectory planner that can account for the vehicle tip-over stability over uneven terrains. This, in turn, requires explicitly predicting the vehicle's $6dof$ pose on the terrain and the kinematic and stability costs associated with those poses.

There are broadly two ways to predict the pose of a wheeled vehicle on an uneven terrain. The first approach is to use high-fidelity physics simulators that can model complex wheel-terrain interaction [1], often in the form of complex differential equations [2]. However, these simulators are often too slow to be used during online planning. Alternately, wheel-terrain interaction and pose prediction can be learned from recorded vehicle motions on different terrains [3]. However, purely data-driven approaches are data-hungry and often struggle to generalize to novel scenarios.

[1] are with the Institute of Technology, University of Tartu, Tartu, Estonia.
[2] are with RRC, IIIT Hyderabad, India.

This work was co-funded by the European Social Fund and Estonian Research Council via project TEM-TA101, Grant PSG753, and in part by grants available through Mathworks India.

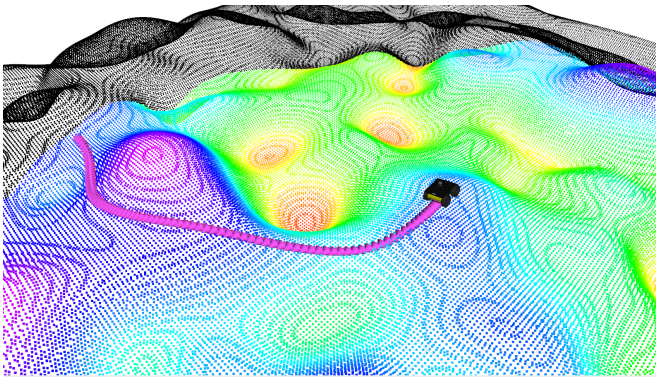


Fig. 1: Husky navigating an uneven terrain. The black points represent the entire terrain, whereas the coloured points represent the terrain's local patch. All points within a given radius from the Husky are used to model the terrain, which is used to compute a stable trajectory to the goal. As seen above, the trajectory (shown in pink) successfully avoids the ditch and goes around it to reach the goal.

Our main motivation in this paper is to make the physics/model-based approach more computationally tractable by finding the appropriate simplified abstraction for the wheel-terrain interaction. Moreover, we aim to make the interaction model differentiable and parallelizable to ensure online planning. We achieve these features through some core algorithmic innovations, which are summarized below, along with their benefits.

Algorithmic Contribution: We jointly view the wheeled vehicle and the underneath terrain through the lens of closed-loop kinematics, typically used for modeling parallel manipulators. This, in turn, leads to a set of coupled non-linear equations that dictate how the vehicle's $6dof$ pose depends on the terrain surface geometry. The pose prediction can then be obtained by minimizing the residual of the non-linear equations, which, in our case, is done through a least-squares optimization problem. Importantly, our set-up allows for differentiating through the least squares solver using implicit function theorem to ascertain how the pose of the vehicle will vary if its position on the terrain is perturbed. Thus, in essence, our non-linear least squares (NLS) approach provides a differentiable wheel-terrain interaction model, albeit at the kinematic level.

The differentiability feature also opens up efficient ways of performing trajectory planning on uneven terrain. Specifically, *for the first time*, we show that the planning problem can be viewed as a bi-level optimization. Its inner layer (NLS) predicts pose along a given trajectory while the outer layer perturbs the trajectory itself to lower the kinematic and stability costs. The implicit gradients from the inner layer

can be used within any momentum-based gradient descent approach to efficiently solve the bi-level problem.

State-of-the-Art Performance: We show that the prediction from our NLS-based approach closely matches the output from a high-fidelity physics engine. As a result, the former provides a computationally cheap alternative for learning-based approaches that require tele-operating vehicles on potentially dangerous terrains. We show that by modeling the stability costs through force-angle measure [4], our trajectory optimizer produces trajectories that successfully avoid risky areas such as ditches and valleys. Finally, we show that our gradient-based approach is competitive with more computationally demanding sampling-based optimization, such as the Cross Entropy Method [5] that does not require access to implicit gradients.

II. Related Works

Learning Based Wheel-Terrain Interaction: An unsupervised method for the classification of a ground rover's slip events based on proprioceptive signals is proposed in [6]. The method is able to automatically discover and track various degrees of slip on the given terrain. A meta-learning-based approach is proposed in [7] to predict future dynamics by leveraging rover-terrain interaction data. In [8], an approach for learning and predicting the motion resistance encountered by a vehicle while traversing is studied using the information obtained from a stereovision device. A learning-based approach for predicting the pitch and roll angle from a depth map was proposed in [9]. Authors in [10] show that purely data-driven approaches struggle to accurately predict the vehicle pose on the terrain. Thus, they embed physics-based priors into the learning pipeline. Our current approach is closely related [10], as our NLS-based pose predictor can also be potentially used as a physics-based prior.

Motion Planning on Uneven Terrains: Model-based planning on uneven terrains requires a pose predictor module. This can be either a learned model as used in [3], or that obtained from first principles of physics and differential equations [11]. A hybrid trajectory optimization method for generating stable and smooth trajectories for articulated tracked vehicles is proposed in [12]. A vehicle-terrain contact model is developed to divide the vehicle's motion into hybrid modes of driving and traversing, which is converted into a nonlinear programming problem to be solved in a moving-horizon manner. A terrain pose mapping to describe the impact of terrain on the vehicle is proposed in [13], based on which a trajectory optimization framework for car-like vehicles on uneven terrain is built. Our proposed method differs from these cited works in how we leverage implicit gradients from a first principle-based wheel-terrain interaction model for trajectory optimization.

End-to-End Learning Based Approaches: A learning-based end-to-end navigation for planetary rovers is given in [14]. A deep reinforcement learning-based navigation method is proposed to autonomously guide the rover towards goals through paths with low wheel slip ratios. An end-to-end network architecture is proposed, in which the visual perception and the wheel-terrain interaction are combined to learn the terrain properties implicitly. In [15], a sim-to-real pipeline for a vehicle to learn how to navigate on rough terrains is presented. A deep reinforcement learning architecture is used to learn a navigation policy from data obtained from simulated environments. An approach that employs a fully-trained deep reinforcement learning network that uses elevation maps of the environment, vehicle pose, and goal as inputs to compute an attention mask of the environment is proposed in [16]. The attention mask is then used to identify reduced stability regions in the terrain.

Connection to Author's Prior Work:: The proposed NLS-based pose predictor builds upon and improves our prior work [17]. Herein, small angle approximation and local planar assumption on the underlying terrain were used to obtain closed-form solutions for pose prediction. However, such an approach is not appropriate for highly uneven terrains.

III. Preliminaries

Symbols and Notation: Scalars are denoted by normal lowercase letters, whereas bold font variants are used for vectors. Matrices are represented by uppercase bold font letters. The superscript T is used to define the transpose of a matrix or a vector.

A. Primer on Chain Rule of Differentiation

This section covers some basic notations from multi-variable calculus that will be crucial for developing the implicit differentiation rules later in the paper. Most of the developments follow from the work [18].

Consider a function $\psi(\lambda, \phi, \mu)$ where ϕ and μ are themselves functions of λ. In such a scenario, the chain rule of differentiation takes the following form.

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\psi = \frac{\partial \psi}{\partial \lambda}\frac{\mathrm{d}\lambda}{\mathrm{d}\lambda} + \frac{\partial \psi}{\partial \phi}\frac{\mathrm{d}\phi}{\mathrm{d}\lambda} + \frac{\partial \psi}{\partial \mu}\frac{\mathrm{d}\mu}{\mathrm{d}\lambda}. \tag{1}$$

For a scalar function over multiple variables, $\psi(\lambda_1, \lambda_2, \dots \lambda_n) : \mathbb{R}^n \to \mathbb{R}$, the derivative vector can be written as

$$\mathrm{D}\psi = \left[\frac{\partial \psi}{\partial \lambda_1}, \frac{\partial \psi}{\partial \lambda_2}, \dots, \frac{\partial \psi}{\partial \lambda_n}\right] \in \mathbb{R}^{1\times n}. \tag{2}$$

Analogously, for vector-valued functions $\boldsymbol{\psi} : \mathbb{R} \to \mathbb{R}^m$, defined over a scalar argument, we can write

$$\mathrm{D}\psi = \left[\tfrac{\mathrm{d}\psi_1}{\mathrm{d}\lambda}, \dots, \tfrac{\mathrm{d}\psi_m}{\mathrm{d}\lambda}\right] \in \mathbb{R}^{m\times 1}. \tag{3}$$

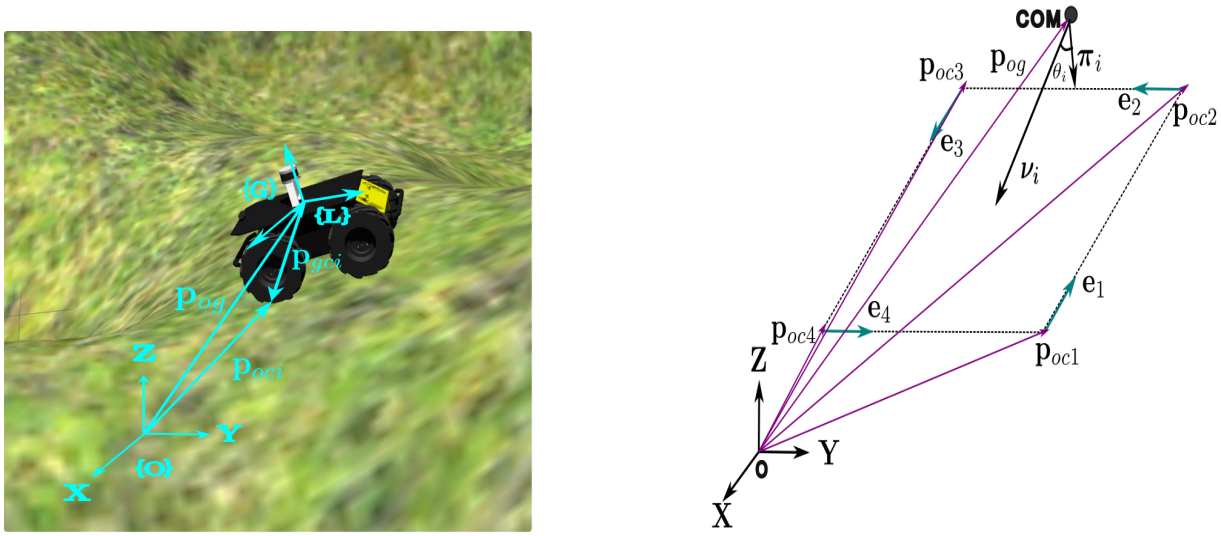
The general definition of $\mathrm{D}\boldsymbol{\psi}$ of $\boldsymbol{\psi} : \mathbb{R}^n \to \mathbb{R}^m$ can be written as an $m \times n$ matrix with entries

$$(\mathrm{D}\boldsymbol{\psi}(\lambda))_{ij} = \frac{\partial \boldsymbol{\psi}_i}{\partial \lambda_j}(\lambda). \tag{4}$$

Now, the chain rule for $h(\lambda) = \zeta(\psi(\lambda))$ can be written as

$$\mathrm{D}h(\lambda) = \mathrm{D}\zeta(\psi(\lambda))\mathrm{D}\psi(\lambda). \tag{5}$$

For the partial derivatives, the variable over which the derivative is computed is written as a subscript. For example, $\mathrm{D}_\lambda\psi(\lambda, \phi)$ denotes partial derivative with respect to λ. Similarly, $\mathrm{D}^2_{\lambda\phi}\psi$ means $\mathrm{D}_\lambda(\mathrm{D}_\phi\psi)^T$.



(a) (b)

Fig. 2: (a) A four-wheeled vehicle with the geometry vectors describing the holonomic constraints. (b) Definition of the vectors associated with tip-over stability.

Finally, when subscripts are not given for multi-variate functions, the operator D means the total derivative with respect to the independent variables. So, with λ as the independent variable, the vector form of Equation (1) is represented by

$$\mathrm{D}\psi = \mathrm{D}_\lambda\psi + \mathrm{D}_\phi\psi\mathrm{D}\phi + \mathrm{D}_\mu\psi\mathrm{D}\mu. \tag{6}$$

B. Trajectory Parametrization

We assume that the vehicle follows a bi-cycle model. Moreover, we can leverage its differential flatness property [19] to plan directly in the positional space (x_k, y_k). The control inputs, for example, linear velocity and curvature, can then be obtained as a function of positional variables and their higher-order derivatives. We parametrize the position-level trajectory of the vehicle in terms of polynomials in the following form:

$$\big[x_1, \ldots, x_n\big] = \mathbf{W}\mathbf{c}_x, \big[y_1, \ldots, y_n\big] = \mathbf{W}\mathbf{c}_y, \tag{7}$$

where, $\mathbf{W}$ is a matrix formed with time-dependent polynomial basis functions and $(\mathbf{c}_x, \mathbf{c}_y)$ are the coefficients of the polynomial. We can also express the derivatives in terms of $\dot{\mathbf{W}}, \ddot{\mathbf{W}}$.

C. Tip-over stability criteria

A vehicle's stability at a given point on the terrain can be calculated by the force angle measure or tip-over stability criteria proposed in [4], [20]. Fig. 2b shows the polygon formed with the ground contact points of a four-wheel vehicle. The edges of the polygon given by the vectors $\mathbf{e}_i$ represent the tip-over axes, i.e., the axes about which the vehicle could possibly tip-over. These axes can be computed in the following way:

$$\mathbf{e}_i = \mathbf{p}_{oc,i+1} - \mathbf{p}_{oc,i}, \forall i = 1, 2, 3, \tag{8}$$

$$\mathbf{e}_4 = \mathbf{p}_{oc,1} - \mathbf{p}_{oc,4}.$$

We define the unit vectors $\hat{\mathbf{e}}_i = \frac{\mathbf{e}_i}{\|\mathbf{e}_i\|}$, $\hat{\boldsymbol{\pi}}_i = \frac{\boldsymbol{\pi}_i}{\|\boldsymbol{\pi}_i\|}$, and $\hat{\boldsymbol{\nu}}_i = \frac{\boldsymbol{\nu}_i}{\|\boldsymbol{\nu}_i\|}$, where $\boldsymbol{\pi}_i$ are the tip-over axis normals which intersect the vehicle's center of mass (COM) and $\boldsymbol{\nu}_i$ are the components of the force acting on the vehicle which contribute to the tip-over. These vectors can be calculated as:

$$\boldsymbol{\pi}_i = (\mathbf{I} - \hat{\mathbf{e}}_i\hat{\mathbf{e}}_i^T)(\mathbf{p}_{oc,i+1} - \mathbf{p}_{og}), \tag{9}$$

$$\boldsymbol{\nu}_i = (\mathbf{I} - \hat{\mathbf{e}}_i\hat{\mathbf{e}}_i^T)\boldsymbol{\nu}, \tag{10}$$

where $\mathbf{I}$ is the identity matrix and $\boldsymbol{\nu}$ is the total force acting on the vehicle. The angles between the forces and the tip-over axis normals can be computed as:

$$\theta_i = \sigma_i \arccos(\hat{\boldsymbol{\nu}}_i \cdot \hat{\boldsymbol{\pi}}_i), \tag{11}$$

where,

$$\sigma_i = \begin{cases} +1, & (\hat{\boldsymbol{\pi}}_i \times \hat{\boldsymbol{\nu}}_i) \cdot \hat{\mathbf{e}}_i < 0 \\ -1, & \text{otherwise}, \end{cases} \tag{12}$$

from which the tip-over stability criteria can be defined as:

$$\theta_i > 0 \implies \min\theta_i > 0, \forall i = 1, 2, 3, 4. \tag{13}$$

We define the costs associated with the tip-over stability criteria as:

$$\theta_{c,i} = \max(0, -\theta_i + \epsilon), \tag{14}$$

$$\theta_{\Delta c,i} = (\theta_i - \theta_{i+1})^2, \tag{15}$$

where ϵ is a small number. The total stability cost for the vehicle is defined as:

$$c_s = \sum_i (\theta_{c,i} + w_\theta\, \theta_{\Delta c,i}), \tag{16}$$

where w_θ is a scaling factor.

IV. Main Algorithmic Results

Our trajectory planning pipeline has the following components. First, we fit a functional form to the terrain digital elevation data, which is then used to formulate a differentiable wheel-terrain interaction model. Subsequently, we formulate a trajectory planning problem that leverages the differentiability of the wheel-terrain interaction.

A. Functional Form for the Terrain Model

Imagine that we are given the digital elevation data, which describes the height z^j of the terrain at any given point (x^j, y^j). For example, these can be obtained online from the point clouds obtained with on-board LiDAR. In this section, our aim is to fit a functional form to this data. That is, we want to obtain analytical relationships of the form:

$$z^j = f(x^j, y^j). \tag{17}$$

Our approach is built on approximating f in terms of Fourier basis functions. That is,

$$\begin{aligned} f(x^j, y^j) = \sum_{n=1}^{N} & a_n \cos\left(\omega_{1,n} x^j + \omega_{2,n} y^j\right) \\ & + b_n \sin\left(\omega_{3,n} x^j + \omega_{4,n} y^j\right), \end{aligned} \tag{18}$$

where $\omega_{1,n}, \omega_{2,n}, \omega_{3,n}, \omega_{4,n}$ are the frequencies of the Fourier basis functions, N is the number of frequencies, and a_n, b_n are the weights associated with each functions. These are obtained by solving the following regression problem

$$\sum_{j=1}^{M} \| f(x^j, y^j) - z^j \|_2^2, \tag{19}$$

where M is the total number of points in the digital elevation data or point-cloud. Fig.3 shows the fit obtained over circular terrain patches of radius $7m$.

Practical Considerations: The number of frequencies needed to get a good fit depends on the area of the terrain. Our implementation, which leverages warm-started gradient descent on GPUs, takes on an average of $0.5s$ to fit the Fourier approximation to a circular patch of radius $7m$. During practical implementation, we can adopt a receding horizon approach, where the vehicle can fit the terrain model over a certain distance and then plan a stable trajectory over it. Such an approach would ensure that the terrain fit is called sparingly.

B. Differentiable Wheel-Terrain Interaction

This section presents the first core result of the paper. Let the yaw plane states of the vehicle at time k can be represented by the vector $\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \alpha_k \end{bmatrix}^T$, where (x_k, y_k) is the position, and α_k is the heading angle of the vehicle. For wheeled vehicles with no-active suspension mechanism, the evolution of a vehicle's yaw plane configuration, i.e., $\mathbf{x}_k$, can be directly controlled. The rest of the pose variables, i.e., z_k coordinate, roll β_k, and pitch γ_k, will be a function of the yaw plane configuration and the terrain geometry.

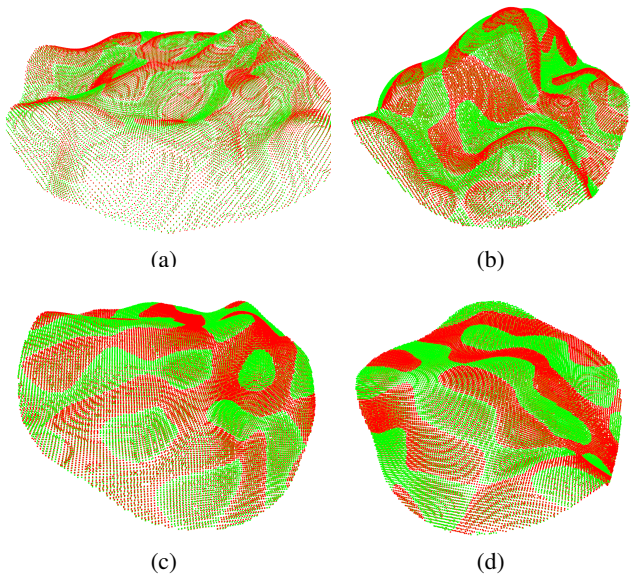


Fig. 3: Red points represent the ground truth terrain patch, and green points represent the terrain patch with predicted z coordinates.

Mathematically, this dependency can be represented in the following manner.

$$\begin{aligned} z_k &= s_1(x_k, y_k, \alpha_k), \\ \beta_k &= s_2(x_k, y_k, \alpha_k), \\ \gamma_k &= s_3(x_k, y_k, \alpha_k). \end{aligned} \tag{20}$$

In this section, we are interested in obtaining the functions s_i that decide the wheel-terrain interaction. To this end, we view the wheeled vehicle on uneven terrain as a parallel manipulator [21] and derive the so-called loop closure equations. To this end, refer to Fig. 2a and imagine that we have a global reference frame $\{O\}$ and a reference frame $\{G\}$ that moves along with the vehicle and is attached to its center. The reference frame $\{L\}$ is also similar, but its orientation is the same as the vehicle. With these notations in place, the vector describing the wheel-ground contact point $\mathbf{p}_{oc,i}$ can be directly obtained from the terrain equation. Alternately, the contact points can also be reached by first following through $\mathbf{p}_{og}$ that identifies the center of the vehicle and then moving along $\mathbf{p}_{gc,i}$. This insight can be encoded into the following equation, which resembles the loop closure equations obtained for parallel manipulators [21] [17].

$$\mathbf{p}_{og} + \mathbf{p}_{gc,i} = \mathbf{p}_{oc,i}, \tag{21}$$

$$\mathbf{p}_{gc,i} = \mathbf{R} \begin{bmatrix} \delta_i h & r_i w & -(l_i) \end{bmatrix}, \forall i = 1,2,3,4, \tag{22}$$

$$\delta_i = \begin{cases} 1, & i = 1,4, \\ -1, & i = 2,3, \end{cases}$$

$$r_i = \frac{2.5 - i}{|2.5 - i|},$$

$$\mathbf{p}_{og} = \begin{bmatrix} x_k & y_k & z_k \end{bmatrix}, \tag{23}$$

$$\mathbf{p}_{oc,i} = \begin{bmatrix} x_{c,i_k} & y_{c,i_k} & z_{c,i_k} \end{bmatrix}. \tag{24}$$

The term $\mathbf{p}_{gc,i}$ is expressed as the vector describing the wheel contact point in the vehicle local frame $\{L\}$ multiplied with the rotation matrix between the $\{O\}$ and $\{L\}$ (refer Fig. 2a). The variables r_i and δ_i have been incorporated to ensure the proper sign of w and h corresponding to each vertex of the chassis. l_i are the equivalent leg lengths, including the wheels' radius. The constants h and w are the half-width and half-breadth of the chassis.

We also note that $\mathbf{p}_{oc,i}$ satisfy the terrain equation (17) and thus, it is possible to write:

$$z_{c,i_k} = f(x_{c,i_k}, y_{c,i_k}). \tag{25}$$

We can expand (21) using the Euler-angle parametrization for the rotation matrix $\mathbf{R}$. When the resulting expressions are stacked alongside (25), we obtain a set of 16 coupled non-linear equations in the following form:

$$g_j(\mathbf{x}_k, \mathbf{u}_k) = 0, \quad j = 1 \dots 16, \tag{26}$$

where $\mathbf{u}_k(\mathbf{x}_k) = \begin{bmatrix} z_k & \beta_k & \gamma_k & x_{c,i_k} & y_{c,i_k} & z_{c,i_k} \end{bmatrix}^T$. The pose prediction can now be formulated as a non-linear least squares (NLS) problem:

$$\mathbf{u}_k^*(\mathbf{x}_k) = \arg\min_{\mathbf{u}_k} \sum_j g_j(\mathbf{x}_k, \mathbf{u}_k)^2. \tag{27}$$

Remark 1: The residual associated with the $\mathbf{u}_k^*(\mathbf{x}_k)$ can be used to ascertain whether it is possible for the wheeled vehicle to ensure contact on all four wheels on a particular patch of terrain.

Implicit Differentiation For an efficient solution of the trajectory optimization problem introduced later, it is important to compute the Jacobian of $\mathbf{u}_k^*(\mathbf{x}_k)$ with respect to its input parameter $\mathbf{x}_k$. However, the relationship between $\mathbf{u}_k^*$ and $\mathbf{x}_k$ does not have an analytical form and thus requires tools from implicit differentiation. The derivation is based on Dini's implicit function theorem [22] applied to the first-order optimality condition. We define the following proposition that has been adapted from [18] for our pose prediction NLS.

Proposition 1: Consider the NLS problem (27). We can define $\mathbf{H} = \mathrm{D}^2_{\mathbf{u}_k\mathbf{u}_k}\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k(\mathbf{x}_k)) \in \mathbb{R}^{m\times m}$, where $\mathbf{g}(.)$ is obtained by stacking $g_j(.)$ and $\mathbf{B} = \mathrm{D}^2_{\mathbf{x_k}\mathbf{u_k}}\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k(\mathbf{x}_k)) \in \mathbb{R}^{m\times n}$. Then, the Jacobian of the optimal pose is

$$\mathrm{D}\mathbf{u}_k^*(\mathbf{x}_k) = -\mathbf{H}^{-1}\mathbf{B},$$

with the assumption that $\mathbf{H}$ is non-singular.

Proof: The first-order optimality condition of our NLS is given by $\mathrm{D}_{\mathbf{u}_k}\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{0}_{1\times m}$. Differentiating both sides of this optimality condition with respect to $\mathbf{x}_k$ provides us the following equation

$$\begin{aligned}\mathbf{0}_{m\times n} &= \mathrm{D}(\mathrm{D}_{\mathbf{u}_k}\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k))^T \\ &= \mathrm{D}^2_{\mathbf{x}_k\mathbf{u}_k}\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) + \mathrm{D}^2_{\mathbf{u}_k\mathbf{u}_k}\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k)\mathrm{D}\mathbf{u}_k(\mathbf{x}_k),\end{aligned}$$

which can be rearranged to

$$\mathrm{D}\mathbf{u}_k(\mathbf{x}_k) = -(\mathrm{D}^2_{\mathbf{u}_k\mathbf{u}_k}\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k))^{-1}\mathrm{D}^2_{\mathbf{x}_k\mathbf{u}_k}\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k), \tag{28}$$

when $\mathrm{D}^2_{\mathbf{u}_k\mathbf{u}_k}\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k)$ is non-singular. ■

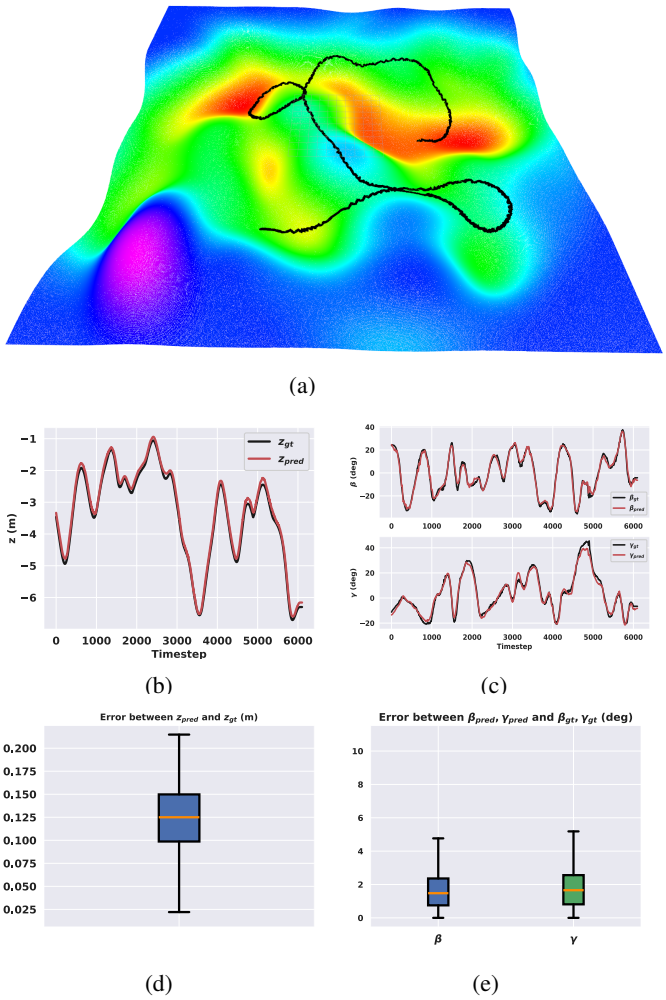


Fig. 4: (a) Trajectory obtained from manually driving Husky on a synthetic terrain in Gazebo. (b) z ground truth and predicted values. (c) β and γ ground truth vs predicted. (d) Error statistics for z. (e) Error statistics for β and γ.

C. Bi-Level Optimization Based Trajectory Planning

We assume a differentially flat non-holonomic vehicle. Thus, its heading, velocity, and curvature can be uniquely defined by the position-level trajectory. For example, the heading can be obtained as $\arctan(\dot{y}_k, \dot{x}_k)$. With this insight, we propose the following bi-level optimization problem

$$\min \sum_k \left(c_r(\ddot{x}_k, \ddot{y}_k, \dot{x}_k, \dot{y}_k) + c_s(\mathbf{u}_k^*(\mathbf{x}_k))\right), \tag{29a}$$
$$(x_0, y_0, \dot{x}_0, \dot{y}_0, \ddot{x}_0, \ddot{y}_0) = \mathbf{b}_0, \tag{29b}$$
$$(x_n, y_n, \dot{x}_n, \dot{y}_n, \ddot{x}_n, \ddot{y}_n) = \mathbf{b}_n, \tag{29c}$$
$$\underline{x} \le x_k \le \overline{x}, \tag{29d}$$
$$\underline{y} \le y_k \le \overline{y}, \tag{29e}$$
$$\mathbf{u}_k^*(\mathbf{x}_k) = \arg\min_{\mathbf{u}_k} \sum_j g_j(\mathbf{x}_k, \mathbf{u}_k)^2. \tag{29f}$$

$$c_a(\ddot{x}_k, \ddot{y}_k) = \ddot{x}_k^2 + \ddot{y}_k^2, \tag{29g}$$
$$c_c(\ddot{x}_k, \ddot{y}_k, \dot{x}_k, \dot{y}_k) = \ddot{y}\dot{x} - \ddot{x}\dot{y}/(\dot{x}^2 + \dot{y}^2 + \epsilon)^{\frac{3}{2}}, \tag{29h}$$
$$c_r(.) = c_a(.) + c_c(.). \tag{29i}$$

The first term ($c_r(.)$) in the cost function (29a) is the kinematics cost that ensures smoothness in the planned trajectory by penalizing high accelerations and sharp turns in the trajectory. The last term ($c_s(.)$) is the tip-over stability cost defined in (16). Equality constraints (29b) and (29c) ensure that the planned trajectory satisfies the initial and final boundary conditions. Consequently, $\mathbf{b}_0, \mathbf{b}_n$ are the stacked vectors of initial and final positions, velocities, and accelerations. The constants $\underline{x}, \underline{y}, \overline{x}, \overline{y}$ are the minimum and maximum bounds for x_k, y_k. The position level is critical for restricting the trajectory to lie within the patch that is currently observable by the vehicle's sensors.

Using the trajectory parametrization given in Sec III-B, the Bi-level optimization problem can be written in a compact form as given below.

$$\min_{\boldsymbol{\xi}} c_r(\boldsymbol{\xi}) + c_s(\mathbf{u}^*(\boldsymbol{\xi})), \tag{30a}$$

$$\mathbf{u}(\boldsymbol{\xi}) = \min_{\mathbf{u}} \|\mathbf{g}(\boldsymbol{\xi}, \mathbf{u})\|_2^2, \tag{30b}$$

$$\mathbf{A}_{eq}\boldsymbol{\xi} = \mathbf{b}_{eq}, \quad \mathbf{A}\boldsymbol{\xi} \leq \mathbf{b}. \tag{30c}$$

where $\boldsymbol{\xi} = [\mathbf{c}_x, \mathbf{c}_y]$ and $\mathbf{u}^*$ is formed by stacking $\mathbf{u}_k^*$ at different instants of time. The matrices $\mathbf{A}_{eq}, \mathbf{A}$ and the vectors $\mathbf{b}_{eq}, \mathbf{b}$ comes from (29b)-(29e).

D. Projected gradient descent

The optimal value for the parameter $\boldsymbol{\xi}$ is obtained through projected gradient descent. The gradient descent update rule is given as

$${}^{t+1}\overline{\boldsymbol{\xi}} = {}^{t}\boldsymbol{\xi} - \eta \left(\nabla_{\boldsymbol{\xi}} c_r({}^{t}\boldsymbol{\xi}) + \mathrm{D}c_s \left(\mathbf{u}^* \left({}^{t}\boldsymbol{\xi} \right) \right) \right), \tag{31}$$

where η is the learning rate. The parameter $\boldsymbol{\xi}$ is projected onto the constraint set using the following Quadratic Programming problem at each iteration t of the gradient descent.

$${}^{t+1}\boldsymbol{\xi} = \arg\min_{\boldsymbol{\xi}} \frac{1}{2} \left\| {}^{t+1}\overline{\boldsymbol{\xi}} - {}^{t+1}\boldsymbol{\xi} \right\|_2^2, \tag{32}$$

$$\mathbf{A}_{eq}\, {}^{t+1}\boldsymbol{\xi} = \mathbf{b}_{eq}, \tag{33}$$

$$\mathbf{A}\, {}^{t+1}\boldsymbol{\xi} \leq \mathbf{b}. \tag{34}$$

The term $\mathrm{D}c_s \left(\mathbf{u}^* \left({}^{t}\boldsymbol{\xi} \right) \right)$ will be computed through implicit differentiation discussed in the previous section. However, since the implicit gradient is computed for a scalar cost c_s, we never actually have to explicitly form the Jacobian of $\mathbf{u}^*$ with respect to $\boldsymbol{\xi}$. Rather, we just need the Jacobian-vector product that can be efficiently computed through existing autodifferentiaton libraries like JAX [23].

V. Results

In this section, we validate our proposed wheel-terrain interaction and pose predictor along with the bi-level optimizer through extensive simulations. We mainly focus on answering the following research questions with the help of qualitative and quantitative (statistical) results.

- **Q1:** How well does our NLS based pose predictor fare vis-a-vis high-fidelity physics simulator Gazebo [24]?
- **Q2:** How does the stability cost affect the trajectory taken by the vehicle?
- **Q3:** How does our gradient descent approach compare to the state-of-the-art Cross Entropy Method (CEM) [5] sampling based planner?

A. Implementation Details

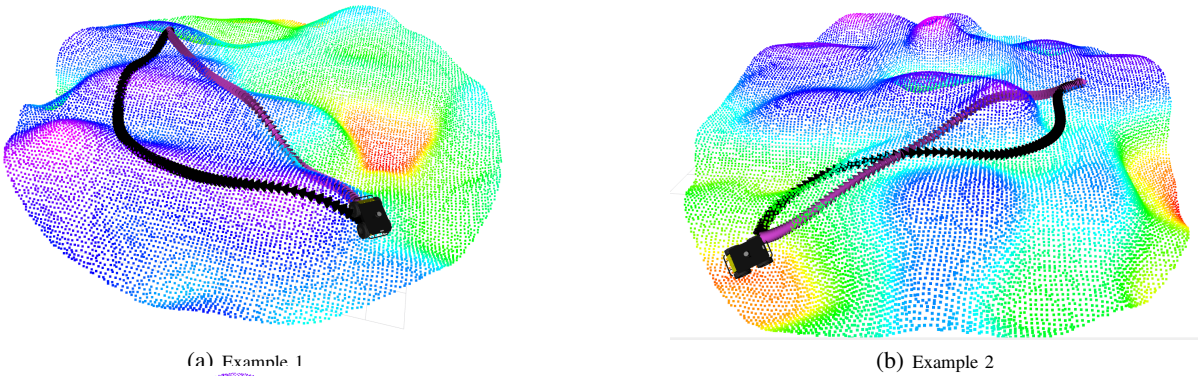
The trajectory planner comprising of the bi-level optimization (30a)-(30c) and projection (33)-(34) is implemented in Python using the JAX [23] library for GPU-accelerated computing. The matrix $\mathbf{W}$ in (7) is constructed from a 10^{th} order polynomial. Our simulation pipeline consists of open-loop analyses carried out with stand-alone Python code, and the closed-loop simulations were performed on the Gazebo simulator [24]. The terrains were created using Blender [25] and were converted to the required formats for the Python and Gazebo simulations. All simulations were conducted for 100 time-steps with a sampling time of $0.2s$ on a computer with Intel i9 CPU and Nvidea RTX 3080 GPU. For modeling the terrain at each instant of time, we take a patch of radius $7m$ centered at the vehicle position. The number of Fourier frequencies N in the terrain model was taken as 100 to balance the trade-off between computation time and accuracy.

B. Validation using the Gazebo simulator

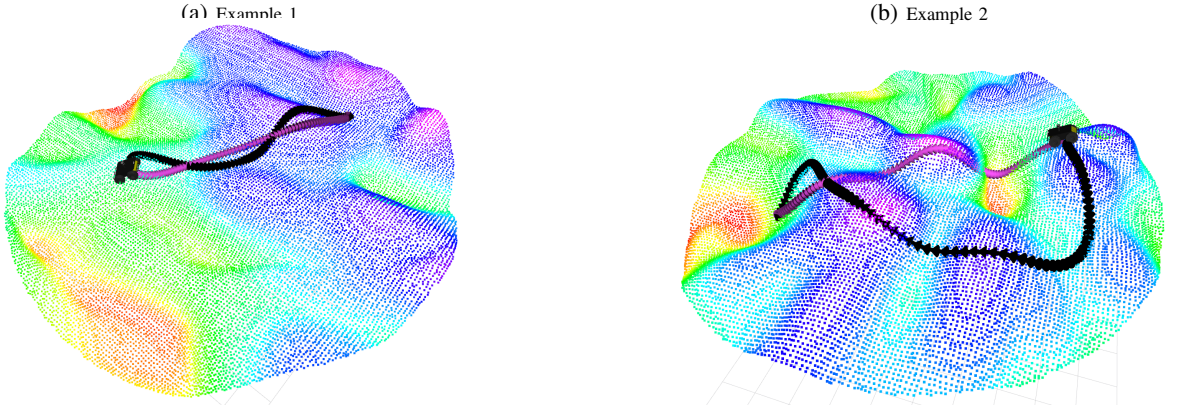
In this section, we analyze the accuracy of our pose predictor by comparing the ground truth data obtained from Gazebo with the values obtained by solving the NLS problem (27) with the help of (17). To this end, we drive the Husky robot in Gazebo on synthetic terrain models and collect the odometry data, as shown in Fig. 4a. We then compare the ground truth values z_{gt}, β_{gt}, and γ_{gt} from Gazebo with the predicted z_{pred}, β_{pred}, and γ_{pred} from our pose optimizer. Several runs of the robot on different terrains were performed, and some qualitative results are shown in Figures 4b and 4c. We can see that the predicted pose values closely match the ground truth values. The statistical results presented in Figures 4d and 4e show the minimum, maximum, and median error obtained across several runs. It can be seen that our NLS pose optimizer is highly accurate in predicting the pose of the robot on different terrains, and the error values are minimal.

C. Validation of the Safe Planning on Uneven Terrain

In this subsection, we evaluate the performance of our trajectory optimizer by considering the tip-over stability metric as an indicator of safe trajectories. Fig 5 shows four example trajectories generated by the planner with and without the stability cost $c_s(.)$ defined in (16). From the figures, we can see that the trajectory computed by the planner with stability cost closely follows the terrain gradients, while the one without the stability cost cuts across hills and valleys in the terrain. The former behavior results in a safer and more stable trajectory toward the goal point, whereas the latter is prone to tipping along the way. The variations in the roll angle β and the pitch angle γ for the trajectories are plotted in Fig. 6. It can be seen that the magnitude of

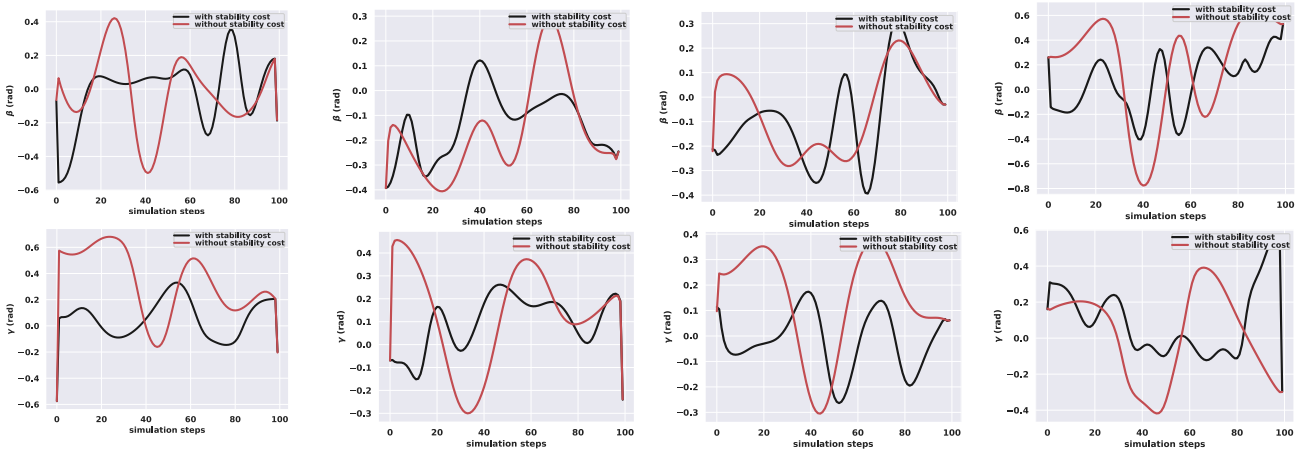


(a) Example 1 (b) Example 2



(c) Example 3 (d) Example 4

Fig. 5: Trajectories with and without stability cost. Black colour represents trajectory with stability cost and pink colour without the stability cost.



(a) Example 1 (b) Example 2 (c) Example 3 (d) Example 4

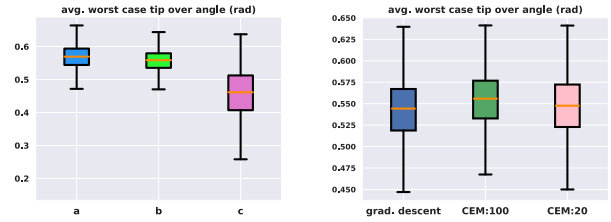
Fig. 6: Roll and pitch angle variations with and without the stability cost.

the angles is higher for the trajectory without the stability cost. Also, there are sharp variations in the angles, which means that the vehicle is traversing through a very uneven terrain. Sudden changes in the vehicle's orientation may lead to the loss of stability. With the stability cost in effect, the magnitude and variations in the angles become lower, and the vehicle becomes more stable.

Next, we present a statistical result in Fig. 7a. Here, we also highlight the significance of the weighting term w_θ in the stability cost 16. Several trajectories were simulated on different terrains, and the average worst-case tip-over angles were recorded. It can be seen that for the trajectories without stability cost, the value is lower. In the case of tip-over angle, higher positive values are preferable, as shown in (13). The values with stability cost are far higher than those without stability cost, which confirms that the trajectories taken by the vehicle with stability cost have a higher safety margin. It can also be seen that as the weight w_θ increased from 0.05 to 0.2, the values of the angles increased, which in turn increased the tip-over stability margin.

D. Comparison with CEM sampling-based planner

This section investigates the importance of differentiability of the wheel-terrain interaction/pose predictor model. If the



(a) a: with c_s, $w_\theta = 0.2$, b: with c_s, $w_\theta = 0.05$, c: without c_s

(b) gradient descent compared with CEM with batch sizes 100, 20.

Fig. 7: (a) Comparison of the effect of the stability cost, c_s. (b) Comparison between gradient descent and CEM.

implicit gradients are not available, then one can use a black-box sampling-based optimizer such as CEM. However, it will require an order of magnitude larger calls of the NLS pose predictor. In fact, it may not even be possible to solve our bi-level optimizer through a sampling-based approach on resource-constrained hardware. Nevertheless, in this section, our aim is to show that our gradient-based approach is competitive with a more computationally demanding approach. The results are summarized in Fig 7b, which shows the statistical results of the average worst-case tip-over angle obtained with our gradient-based approach and CEM. Two different batch sizes (100 and 20) were used for CEM. The results show that our gradient descent-based planner performs as well as a CEM planner with batch size 100 and is slightly better than a CEM with batch size 20. The average computation time for our approach was $0.42s$, whereas the CEM with batch sizes 100 and 20 took $2.12s$ and $1.03s$, respectively. It is also to be noted that the CEM planners are memory-hungry; thus, implementation on onboard computers might be difficult, if at all possible.

VI. CONCLUSIONS

We presented, for the first time, a differentiable wheel-terrain interaction/pose predictor model that is derived purely from the first principles and yet generalizes to arbitrary terrains. The fidelity of our model was shown to be close to that of a state-of-the-art physics engine. This opens up exciting new possibilities. For example, our pose predictor can be used as a differentiable and parallelizable world model within trajectory optimization or even reinforcement learning pipelines. We presented one such approach based on bi-level optimization, wherein we leveraged implicit gradients from the NLS solver to efficiently compute the optimal trajectory. The proposed optimizer can accommodate arbitrary stability metrics that depend on the vehicle's pose on uneven terrains. As an example, we used the force-angle measure and demonstrated stable trajectory planning on uneven terrains. Our future endeavors are focused on extending the NLS-based approach to handle wheel-terrain dynamics.

REFERENCES

[1] V. Wiberg, E. Wallin, T. Nordfjell, and M. Servin, "Control of rough terrain vehicles using deep reinforcement learning," *IEEE robotics and automation letters*, vol. 7, no. 1, pp. 390–397, 2021.

[2] A. Gattupalli, V. P. Eathakota, A. K. Singh, and K. Madhava Krishna, "A simulation framework for evolution on uneven terrains for synchronous drive robot," *Advanced Robotics*, vol. 27, 2013.

[3] R. Agishev, T. Petříček, and K. Zimmermann, "Trajectory optimization using learned robot-terrain interaction model in exploration of large subterranean environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3365–3371, 2022.

[4] E. Papadopoulos and D. A. Rey, "The force-angle measure of tipover stability margin for mobile manipulators," *Vehicle System Dynamics*, vol. 33, no. 1, pp. 29–48, 2000.

[5] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and computing in applied probability*, vol. 1, pp. 127–190, 1999.

[6] M.-R. Bouguelia, R. Gonzalez, K. Iagnemma, and S. Byttner, "Unsupervised classification of slip events for planetary exploration rovers," *Journal of Terramechanics*, vol. 73, pp. 95–106, 2017.

[7] S. Banerjee, J. Harrison, P. M. Furlong, and M. Pavone, "Adaptive meta-learning for identification of rover-terrain dynamics," *arXiv preprint arXiv:2009.10191*, 2020.

[8] A. Ugenti, F. Vulpi, A. Milella, and G. Reina, "Learning and prediction of vehicle-terrain interaction from 3d vision," in *Multimodal Sensing and Artificial Intelligence: Technologies and Applications II*, vol. 11785. SPIE, 2021, pp. 167–173.

[9] A. Datar, C. Pan, and X. Xiao, "Learning to model and plan for wheeled mobility on vertically challenging terrain," *arXiv preprint arXiv:2306.11611*, 2023.

[10] V. Šalanskỳ, K. Zimmermann, T. Petříček, and T. Svoboda, "Pose consistency kkt-loss for weakly supervised learning of robot-terrain interaction model," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5477–5484, 2021.

[11] V. Eathakota, G. Aditya, and M. Krishna, "Quasi-static motion planning on uneven terrain for a wheeled mobile robot," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 4314–4320.

[12] Z. Xu, Y. Chen, Z. Jian, J. Tan, X. Wang, and B. L. Liang, "Hybrid trajectory optimization for autonomous terrain traversal of articulated tracked robots," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 755–762, 2023.

[13] L. Xu, K. Chai, Z. Han, H. Liu, C. Xu, Y. Cao, and F. Gao, "An efficient trajectory planner for car-like robots on uneven terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 2853–2860.

[14] W. Feng, L. Ding, R. Zhou, C. Xu, H. Yang, H. Gao, G. Liu, and Z. Deng, "Learning-based end-to-end navigation for planetary rovers considering non-geometric hazards," *IEEE Robotics and Automation Letters*, 2023.

[15] H. Hu, K. Zhang, A. H. Tan, M. Ruan, C. Agia, and G. Nejat, "A sim-to-real pipeline for deep reinforcement learning for autonomous robot navigation in cluttered rough terrain," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6569–6576, 2021.

[16] K. Weerakoon, A. J. Sathyamoorthy, U. Patel, and D. Manocha, "Terp: Reliable planning in uneven outdoor environments using deep reinforcement learning," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 9447–9453.

[17] A. K. Singh and K. M. Krishna, "Feasible acceleration count: A novel dynamic stability metric and its use in incremental motion planning on uneven terrain," *Robotics and Autonomous Systems*, vol. 79, pp. 156–171, 2016.

[18] S. Gould, R. Hartley, and D. Campbell, "Deep declarative networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 3988–4004, 2021.

[19] Z. Han, Y. Wu, T. Li, L. Zhang, L. Pei, L. Xu, C. Li, C. Ma, C. Xu, S. Shen *et al.*, "An efficient spatial-temporal trajectory planner for autonomous vehicles in unstructured environments," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[20] D. Rey and E. Papadoupoulos, "Online automatic tipover prevention for mobile manipulators," in *Proc. of the IEEE International Conference on Intelligent Robot and Systems*, vol. 3, 1997, pp. 1273–1278.

[21] N. Chakraborty and A. Ghosal, "Kinematics of wheeled mobile robots on uneven terrain," *Mechanism and machine theory*, vol. 39, no. 12, pp. 1273–1287, 2004.

[22] A. L. Dontchev and R. T. Rockafellar, *Implicit functions and solution mappings*. Springer, 2009, vol. 543.

[23] "Jax," https://github.com/google/jax.

[24] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ international conference on intelligent robots and systems*, 2004, pp. 2149–2154.

[25] "Blender," https://www.blender.org/.