Learning-based Foot-Shape-Aware Foothold Selection for Quadrupedal Robots

Simone Tolomei[1,2], Dominik Belter[3], Jakub Bednarek[3], Franco Angelini[1,2], Manolo Garabini[1,2]

***Abstract*—Mastering rough terrain locomotion is a tough challenge for robots due to its dynamic, unpredictable nature and frequent physical contact. Traditionally, robots rely on carefully planned foot placements to maintain grip and stability. Recent advancements in quadruped robot feet offer diverse shapes and high grip for various terrains. However, control systems and planners often struggle to leverage these varied capabilities, relying instead on simplified foot models e.g. ball-like, flat. The simplified feet models committed to the single shape of the foot can not be used on robots equipped with diverse feet or modern adaptive feet. This work proposes a novel foothold optimization method that efficiently searches for optimal contact points for different foot shapes using a polynomial approximation. The system leverages a Convolutional Neural Network (CNN) trained on simulated data to predict a cost for each candidate foothold. We show that a single neural network can work with different and new foot mechanical designs without retraining the system. We experimentally validate our system on the ANYmal robot using both ball feet and adaptive soft feet, in indoor and outdoor environments, finding that our system improves stability, in terms of pitch and roll angles of the base, with respect to a state-of-the-art method.**

***Index Terms*—Robotics, Legged Robots, Foothold Selection, Convolutional Neural Networks.**

Fig. 1. ANYmal robot walking on rough terrain employing the proposed foothold planning algorithm based on a Convolutional Neural Network. The method searches for contact points that ensure a high contact surface and a high grip depending on the shape of employed feet.

I. Introduction

ROUGH terrain locomotion is a challenging task that stands in the way of quadrupedal robots when it comes to their deployment for real-world scenarios, such as the natural ones [1]. Avoiding slippage is crucial to ensure that the robot remains in balance. Feet adherence results from the combination of foothold planning, foot design and terrain features. These factors are codependent on each other, and they cannot be considered as single factors.

Over the years, foot design has evolved to better exploit terrain shapes and features. The quadrupedal robots ANYmal [2] and HyQ [3] employ ball-like feet and are both able to walk efficiently in different types of terrain. Flat feet have been used widely in humanoid bipedal robots like WALKMAN [4] or TALOS [5], which are able to safely plan footstep both over flat terrains [6] and rough terrains [7], and recently sensorized flat feet have shown some advantages in quadrupedal locomotion [8]. Catalano et al. [9] showed an adaptive foot able to exploit terrain roughness to improve grip.

Alongside feet design, the planning pipeline needs to be capable of understanding the local terrain's characteristics and be aware of the feet's characteristics. To do so, local terrain features, often captured as elevation maps [10], must be incorporated into planning methods that optimize the robot's foothold selection based on the foot's geometry and properties.

In our previous work [11] we proposed a planning algorithm based on a Convolutional Neural Network (CNN) to prefer footholds suitable for the adaptive foot. However, to the author's knowledge, there is no previous work on a foothold planning algorithm able to select footholds for feet

This research is partially supported by the European Union's HORIZON-MSCA-2023-SE-01-01 - MSCA Staff Exchanges 2023 Program under the Grant Agreement No 101182891(NEUTRAWEED), Horizon 2020 Research and Innovation Programme under Grant Agreement No. 101016970 (Natural Intelligence), and No. 101017274 (DARKO), and Horizon Research and Innovation Programme under Grant Agreement No. 101070596 (euROBIN), in part by the European Union by the Next Generation EU project ECS00000017 'Ecosistema dell'Innovazione' Tuscany Health Ecosystem (THE, PNRR, Spoke 9: Robotics and Automation for Health), and in part by the Italian Ministry of Education and Research in the framework of the "CrossLab" and "FoReLab" (Future-oriented Research Lab) Projects (Departments of Excellence). D. Belter was supported by the National Science Centre, Poland, under research project no UMO-2019/35/D/ST6/03959. We acknowledge the support of the ARCADIT project, funded by the National Center of Research on High Performance Computing, Big Data and Quantum Computing, through an Innovation Grant assigned to UNIPISA and Italferr (CUP: I53C22000690001) S. Tolomei is within the Italian Doctorate in Robotics and Intelligent Machines

[1]Centro di Ricerca "Enrico Piaggio", Università di Pisa, Largo Lucio Lazzarino 1, 56126 Pisa, Italy.

[2]Dipartimento di Ingegneria dell'Informazione, Università di Pisa, Largo Lucio Lazzarino 1, 56126 Pisa, Italy.

[3]Institute of Robotics and Machine Intelligence, Poznan University of Technology, Poznan, Poland

Published version https://doi.org/10.1109/TSMC.2025.3563974

simone.tolomei@phd.unipi.it

of various shapes. With the method proposed in this article, a single neural network can work with different and new foot mechanical designs without retraining the system.

In [11], we used a polynomial approximation of the local terrain's elevation to estimate the ability of the SoftFoot-Q [9] to adapt to the terrain at a given point and to select footholds accordingly. We employed a CNN trained with simulated data to compute the cost of every foothold. However, no experimental validation was presented.

This work is a direct continuation of [11]. As the main contribution with respect to that work, we generalize the polynomial approximation approach to feet of different shapes through a unified polynomial model, we implement a new CNN model to predict the cost value starting from the local elevation map and the foot description. Furthermore, we validate the algorithm both in simulation and experimentally on the quadrupedal robot ANYmal C (Fig. 1). Our contribution to the state of the art can be summarized as follows:

- generalization of the polynomial approximation approach to the evaluation of footholds for feet of different shapes, using a compact polynomial representation to characterize foot-terrain interaction,
- a new unified CNN model to predict the cost value for foothold depending on the local elevation map and the various feet models, without any re-training,
- validation of the algorithm both in simulation and experimentally on the quadrupedal robot ANYmal C that shows that the proposed method applied on the quadruped robot equipped with adaptive feet allows avoiding slippages and stabilizes the robot's body motion compared to the *state-of-the-art* method.

The rest of this paper is organized as follows: in Sec. II, we present the state of the art in the fields of feet design and footstep planning for legged robots. In Sec. III, we describe the problem addressed, with a focus on foothold optimization. In Sec. IV, we present our proposed solution, with the neural network architecture, and we discuss in the detail the dataset generation and preprocessing, the training process of the neural network, and the training results. In Sec. V, we illustrate and discuss the results obtained in simulation, and in Sec. VI, we show the experimental validation both on indoor and outdoor tests, for different types of feet. In Sec. VII, an overall discussion of the results achieved is given, with a focus on the advantages and disadvantages of our approach, and in Sec. VIII we conclude with final remarks and present possible future works.

II. Related Works

A. Feet Design

Various types of robotic feet for legged robots have been proposed in the last years. Ball feet, under which the point contact hypothesis holds, have been used in the quadrupedal robots ANYmal by ANYbotics [2] and HyQ [3]. These robots have shown capable to traverse different types of terrains [12], [13] often relying on state estimation to detect contact state between terrain and feet [14]. Flat feet have been used in the bipedal robots such as WALKMAN [4] and TALOS [5] to provide a stable contact surface with the terrain. Similarly, Kaslin et al. [15] propose a flat foot for quadrupedal robots equipped with an Inertial Measurement Unit (IMU) on the sole. A complementary filter estimates the foot pose, which is used to provide reliable foot contact detection and increase the performance of walking controllers. Valsecchi et al. [8] propose a sensorized flat foot able to estimate local terrain slope and reaction forces. The controller utilizes this additional information to minimize local tangential forces and avoid slippage.

Finally, Piazza et al. [16] propose a bio-inspired adaptive foot for bipedal walkers to enable a natural walking behaviour. A quadrupedal adaptive foot inspired by [16] has been proposed by Catalano et al. [9], and it has shown the ability to adapt to rough terrain shapes and improve grip. This foot, namely SoftFoot-Q, is also equipped with IMUs to enable pose estimation and haptic exploration.

All these designs however require a specialized planning and control pipeline able to exploit their different capabilities.

B. Foothold Selection

Quadrupedal robot locomotion relies heavily on foothold choice, particularly on rough terrains. Exteroceptive sensing is often used to take into account terrain shape and proprieties. Barasuol et al. [17] use an elevation map to evaluate the size of the obstacles and select footholds based on a visual pattern classification. The system is validated on the quadrupedal robot HyQ [3] showing promising results with challenging terrains. The system however requires a feedback by an external motion capture system. Kalakrishnan et al. [12] use expert demonstration to learn optimal foothold choices using terrain templates. While showing good results, this approach is dependant on external expert input to evaluate the foothold cost. In [18], a high-level controller on the quadrupedal robot Little Dog utilizes a linear combination of features such as slope and collision information around the candidate foothold to compute a cost index. A low-level controller based on nonlinear programming then solves for the optimal foothold and computes the trajectory. Similarly, Fankhauser et al. [19] compute a score value for each candidate foothold starting from the slope, roughness, curvature of the terrain, and a confidence score evaluated from the local elevation map. The value is mapped through a threshold to a binary score to obtain a map of feasible footholds. Haptic exploration is used with the StarlETH robot [20] to classify untouched footholds and assess their robustness without human supervision. This approach relies on specialized hardware integrated in the shank to implement a force control when the robot is performing haptic exploration.

Alongside these model-based methods, over the last few years, learning-based approaches have started to show promising results in application related to legged robots, such as legged locomotion [21] and perception [22] Li et al. [23] learn a parametric cost function from expert knowledge to evaluate foothold cost. In [24] a CNN is trained from simulated data to predict cost values of footholds. Similarly, Magana et al. [25] implement a self-supervised learning methodology for foothold selection based on a CNN. While demonstrated on the HyQ robot with both static and dynamic gaits, the learning process requires heuristic functions to be defined by experts. Chen et al. [26] use a CNN to find footholds that guide the robot into an energy-efficient stance, used to climb stairs. The network takes an elevation map as input and encodes the optimal foothold position into a one-hot output vector. In [27] the authors propose a footstep planning policy for quadrupedal robots that directly incorporates safety information during decision-making. By analyzing terrain and classifying landing locations based on safety criteria, the policy achieves significant reductions in safety violations and improves sample efficiency compared to prior methods. However, the mentioned methods focus only on ball feet. In [28] the authors discuss footstep planning for lower limb exoskeleton robots, which share similarities with quadrupedal foothold selection. The work presents a visual footstep planning system that considers human-robot interaction and footstep planning, including the use of a cubic Bezier curve to smooth target trajectories. Yao et al. [29] propose a system consisting of a planner algorithm based on Deep Reinforcement Learning (DRL) and an optimal control strategy for the low-level controller. Fully learning-based techniques have been shown in [30] where an end-to-end blind policy is trained to perform quadrupedal locomotion over rough terrains. These approaches, however, rely on simulated experience, that requires to model complex dynamics and interaction with the environments.

In our previous work [11], we proposed a CNN-based foothold selection algorithm for quadrupedal robots equipped with adaptive feet. The network is trained to compute foothold costs taking into account kinematic properties as well as the terrain shape in order to exploit the adaptive nature of the soft feet. The method proposed in [11] was however limited by the feet dimensions used for generating the dataset, and any change required re-training with a newly-crafted dataset. The same problem exists when the robot is trained to utilize only one and specific feet type, i.e. ball-like [23], [25], [27] or flat [8], [31], [32]. The method from the literature that deals with various shapes of feet is presented in [33]. In this approach, the highest point from the map below the footprint is chosen as a reference Center of Pressure but the the proposed planning strategy does not show how to optimize the foothold. In contrast, this is the core problem solved in this manuscript. The method proposed in this work generalizes the foothold selection planner to quadrupedal robots equipped with feet of various shapes. In particular, we propose an algorithm able to seamlessly find footholds for feet with a continuous curvature on the sole outline, such as flat feet, ball-shaped feet, and adaptive soft feet, without any re-training.

III. Problem Definition

The task of foothold planning is to determine the appropriate contact points, referred to as footholds, for a given motion task. Given the state of the robot x, a gait sequence and a reference base twist ξ_{B}, from which a reference base pose B_{n} is computed, the algorithm computes a sequence of feasible nominal footholds $(p_1, p_2, \ldots, p_n)$ that allows for tracking the reference base pose (Fig. 2).

In particular, the algorithm uses previous knowledge of a default step length to compute a set of stances such that the path from the start to the goal pose is divided into equally spaced intervals [19]. The leg sequence is determined based on the main direction, and the leg with the largest first step is chosen as the start of the walking sequence. Factors like the limits of the leg workspace, self-collisions, and the feasibility of the foothold in terms of the terrain shape, slope, and roughness need to be taken into account. Moreover, the shape of the foot affects the grip and the ability of the foot to maintain contact with the terrain, making it an essential factor to consider during planning. These terms are included in the footstep optimisation phase, which scans the neighbouring terrain around the nominal foothold for an optimal contact point. Once this is found, the leg's swing trajectory is computed using a predefined spline that moves the foot to the selected position and avoids collisions with the terrain [19].

This problem can be formalized as the minimization of a cost function that balances the deviation from the nominal foothold with the feasibility of the foothold under kinematic and terrain constraints. The result is an optimization problem that, at each step, searches an optimal foothold p^* over a set $\boldsymbol{P}$ of candidate footholds in the area around the nominal point, by solving

$$\min_{p \in \boldsymbol{P}} \quad \alpha_{\mathrm{K}} J_{\mathrm{K}}\left(p, x\right) + \alpha_{\mathrm{e}} J_{\mathrm{e}}\left(p\right) + \alpha_{\mathrm{d}} J_{\mathrm{d}}\left(p, p_{\mathrm{n}}\right), \tag{1}$$

where the term $J_{\mathrm{d}}\left(p, p_{\mathrm{n}}\right)$ is the distance between candidate p and nominal foothold p_{n}, the term $J_{\mathrm{e}}(p)$ encodes the foothold feasibility and cost in terms of terrain elevation and shape, and $J_{\mathrm{K}}(p, x)$ represents the kinematic constraints. To tune the optimization process each term is multiplied by a scalar weight, α_{e}, α_{K} and α_{d}, respectively for the feasibility, the kinematic cost and the distance cost. In this work, the set $\boldsymbol{P}$ of candidate footholds consists of a grid of points centered in the hip of the moving leg, with a fixed size and stride.

A. Distance Term

The distance term

$$J_{\mathrm{d}}\left(p, p_{\mathrm{n}}\right) = |p - p_{\mathrm{n}}| \tag{2}$$

is responsible for restricting the deviation from the nominal foothold, that would result in a deviation from the reference base trajectory.

B. Kinematic term

The purpose of the kinematic term is to avoid non-feasible footholds. Those are points that are outside the workspace of the moving leg and points that would bring the robot in self-collision.

This term can be formulated as

$$J_{\mathrm{K}}(p, x) = \begin{cases} K_{\mathrm{M}}(p) & p \in W_{\mathrm{L}}(x) \cap C_{\mathrm{free}}(x) \\ \infty & \text{elsewhere} \end{cases}, \tag{3}$$

where $W_{\mathrm{L}}(x)$ is the workspace of the moving leg, $C_{\mathrm{free}}(x)$ is the collision-free space, and $K_{\mathrm{M}}(p)$ is a kinematic margin for

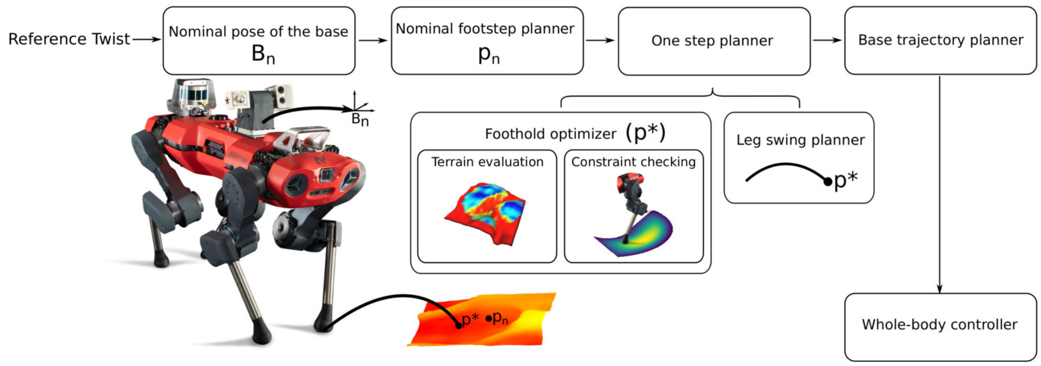


Fig. 2. The foothold planning algorithm starts from a reference twist, it computes a set of nominal foothold positions, and it optimizes them at each step. The foothold optimizer (neural network) simultaneously evaluates the local terrain patch and checks the kinematic constraints – workspace of the leg and collisions of the leg with the terrain. The optimized foothold, the leg swing trajectory and the base trajectory are then sent to the Whole-Body controller for tracking.

the candidate foothold p defined as a minimal distance between the candidate foothold p and the closest point located on the border of the workspace p_w [24],

$$K_{\mathrm{M}}(p) = \min_{p_w} \|p - p_w\| \,. \tag{4}$$

C. Terrain cost term

The stability of the foot-terrain contact interaction depends on a series of factors, among them the local curvature radii of the terrain and the foot sole [9]. When the foot curvature is the same as the local terrain in the contact point, the stability is granted. Catalano et al. [9] designed a compliant foot able to adapt to the terrain curvature in order to satisfy this condition. We obtain similar results by searching for contact points with a local curvature similar to the foot's sole. To do so, we introduce a mathematical model of the foot's sole, in the form of a two-dimensional second-order polynomial.We chose a two-dimensional second-order polynomial because it represents the shape of the various feet in a compact way (Fig. 3). Moreover, the parameters of the polynomial have physical meanings like slope and curvature. Finally, we can quickly check whether the foot curvature is the same as or close to the local terrain curvature in the contact point so the foothold guarantees stable support. With this model, any foot shape is characterized using a series of coefficients $(c_{\mathrm{x},i}, c_{\mathrm{y},i})$ as described by

$$h_{\mathrm{p}}(x,y) = \sum_{i=0}^{2} c_{\mathrm{x},i}\; x^i + \sum_{i=0}^{2} c_{\mathrm{y},i}\; y^i. \tag{5}$$

We can evaluate the grip of each candidate foothold by fitting a second-order polynomial on the underlying elevation map and computing a similarity value between the foot shape and the terrain patch

$$J_{\mathrm{e}} = \alpha_{\mathrm{f}} J_{\mathrm{f}} + \sum_{n=(x,y)} \sum_{i=0}^{2} \frac{\alpha_{\mathrm{n},i}}{c_{\mathrm{n},i}^{\max} - c_{\mathrm{n},i}^{\min}} \cdot \left| c_{\mathrm{n},i}^{\mathrm{foot}} - c_{\mathrm{n},i} \right| . \tag{6}$$

Here, $\alpha_{\mathrm{n},i}$ and α_{f} are scaling coefficients and $c_{\mathrm{n},i}^{\mathrm{foot}}$ are the polynomial coefficients associated with the shape of the foot. The terms $c_{\mathrm{n},i}^{\max}$ and $c_{\mathrm{n},i}^{\min}$ are constants used to normalize the

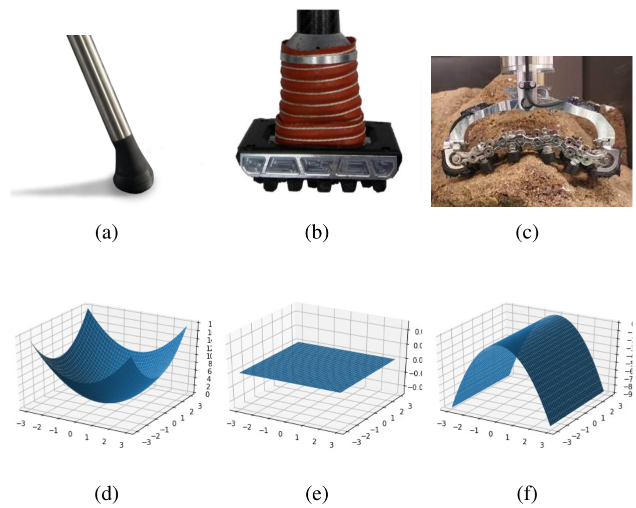


Fig. 3. Each foot is parameterized with the two coefficients C_{p} and C_{r}, described by (8), and associated with the curvature along the longitudinal and lateral axis. Ball foot has $C_{\mathrm{p}} = 2.4$, $C_{\mathrm{r}} = 2.4$ (14), flat foot has $C_{\mathrm{p}} = 0.0$, $C_{\mathrm{r}} = 0.0$ (15) and adaptive foot has $C_{\mathrm{p}} = -2.4$, $C_{\mathrm{r}} = 0.0$ (16). The foot length and width are $L = 0.159\,\mathrm{m}$, $W = 0.056\,\mathrm{m}$, respectively.

cost index. The term J_{f} is the quality of the polynomial fit over the terrain. Defining W and L as width and length of the foot sole, respectively, then J_{f} can be written as

$$J_{\mathrm{f}} = \sum_{x=-L/2}^{L/2} \sum_{y=-W/2}^{W/2} \left| h_{\mathrm{p}}(x,y) - h(x,y) \right| ; \tag{7}$$

where $h(x,y)$ is the terrain elevation and $h_{\mathrm{p}}(x,y)$ is the computed polynomial fitting. Using this parametrization, the coefficients needed to define a foot shape are associated with physical characteristics. In particular $c_{k,0}$ represents the offset, $c_{k,1}$ is the average slope, and $c_{k,2}$ is the curvature, with $k = x, y$. In this work, we focus on feet with a fixed size ($L = 0.159\,\mathrm{m}$, $W = 0.056\,\mathrm{m}$) that is related to the ANYmal C robot and SoftFoot-Q used for experimental verification. The constant parameters L and W imply that the trained model is valid only for a specific foot size. To further simplify the foot characterization, we neglect the offset and assume the same slope coefficient for all possible foot designs. This is done to contain the computational cost and the amount of data required. At these expenses, the assumptions can be relaxed to account for generic size and slope. By defining

$$c_{\mathrm{x},2} = C_{\mathrm{p}}, \\ c_{\mathrm{y},2} = C_{\mathrm{r}}, \tag{8}$$

every foot can be represented as a point in an $\mathbb{R}^2$ space, with C_{p} and C_{r} representing the surface curvature in the pitch and in the roll directions, respectively (Fig. 3).

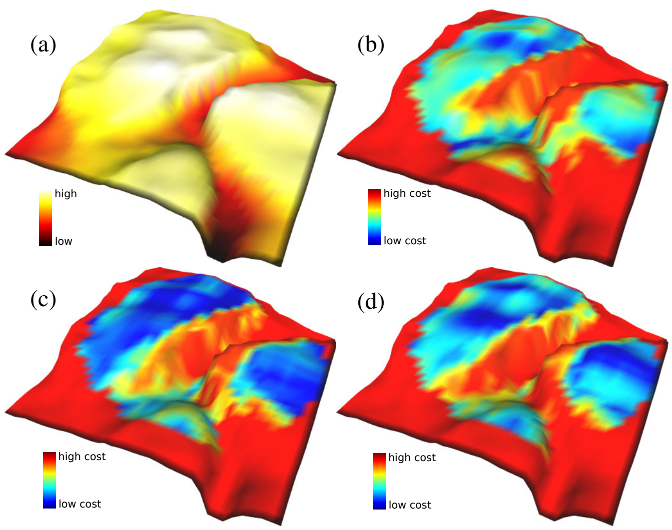


Fig. 4. Foothold cost evaluation for the example terrain patch (a) using the same proposed model (1) and various parameters for a ball-like foot ($C_{\mathrm{p}} = 2.4$, $C_{\mathrm{r}} = 2.4$) (b), flat foot ($C_{\mathrm{p}} = 0.0$, $C_{\mathrm{r}} = 0.0$) (c) and adaptive foot ($C_{\mathrm{p}} = -2.4$, $C_{\mathrm{r}} = 0.0$) (d).

In Fig. 4, we show the results of terrain patch evaluation taking into account the kinematic model of the robot's leg and the proposed terrain cost term. The position of the optimal foothold depends on the C_{p} and C_{r} values. Over the same terrain patch (Fig. 4(a)) robot with ball-like feet chooses small concavities of the terrain (Fig. 4(b)), the flat feet will be located in planar and horizontal parts of the terrain (Fig. 4(c)), while the adaptive feet are placed on small bumps (Fig. 4(d)).

While adaptive feet are robust by design and able to provide good footholds on a wide ranges of terrain patches, we design the algorithm in order to select, among the feasible footholds, the optimal point for the SoftFoot in terms of foot-terrain stability [9], [11].

IV. Solution approach

A. The neural network model

The algorithm described in Sec. III-C is able to evaluate the grip of candidate footholds. However, this approach can be computationally expensive to implement on an onboard computer of a quadrupedal robot.

To overcome this challenge, we use a Convolutional Neural Network (CNN) to evaluate the cost of candidate footholds. By using CNNs, we can provide a terrain patch located around the nominal foothold and evaluate them in a single step. Also, we can take into account time-consuming constraints like collision and kinematic margin that are computed iteratively in traditional footstep planners [19], [34]. Lastly, the trained neural network evaluates footholds in real-time on the board of the robot. The alternatives to CNN are Support Vector Machine [35], Multilayer Perceptron [36] or Gradient Boosting [37]. However, all these methods require low-dimensional input and hand-tuned features. In contrast, CNN learns to extract meaningful features from the input data during learning and reduces the expert involvement in the training process. We also avoid using Vision Transformers like Swin [38] or ViT [39] which are generally slower than CNNs during inference. Even though recent studies are focused on improving the computational efficiency of transformer-based neural networks, the comparison shows that CNNs are still better for lower-resolution images and real-time applications [40]–[42]. Moreover, we propose embedding the foot parameters into the latent space of the CNN. A similar strategy to inject additional information into token embeddings is not applicable to Vision Transformers.

In particular, we compute the terms J_{e} and J_{K} of (1) with a CNN, and we leave the term $J_{\mathrm{d}}(p, p_{\mathrm{n}})$ to be computed at inference time, thus formulating (1) as

$$\min_{p \in \boldsymbol{P}} \quad \underbrace{\alpha_{\mathrm{K}} J_{\mathrm{K}}(p, x) + \alpha_{\mathrm{e}} J_{\mathrm{e}}(p)}_{J_{\mathrm{CNN}}(p,x)} + \underbrace{\alpha_{\mathrm{d}} \left| p - p_{\mathrm{n}} \right|}_{J_{\mathrm{d}}(p,p_{\mathrm{n}})} . \tag{9}$$

The CNN takes as inputs an elevation map and the two coefficients $(C_{\mathrm{p}}, C_{\mathrm{r}})$, the information on collisions and kinematic margin are embedded in the training dataset. Note that we do not provide the state of the robot x as an input to the neural network. The state of the robot x is directly related to the position of the potential footholds located in the cells of the elevation map provided as an input. The configuration of the leg for the given position on the map is computed from the inverse kinematic model of the leg.

The map is centered in the moving leg's hip joint projected on the ground, it has a resolution of 20×20 mm^2 and a size of $s_{\mathrm{x}} \times s_{\mathrm{y}}$ cells, with $s_{\mathrm{x}} = s_{\mathrm{y}} = 40$ cells, where each cell represents a candidate foothold. The distance term is included later in the process, as shown in Fig. 5, with a proportional term α_{d}. This term acts as a tuning parameter for the performance of the robot in a specific task. By increasing α_{d} the robot tends to follow the reference pose more accurately, and by decreasing α_{d} the robot neglects the reference pose and focuses on selecting high-grip footholds.

The result of this process is a map of cost values, each associated with a candidate foothold on the elevation map. Due to the small size of the map, the optimal foothold is found simply by iterating over the costmap and searching for the point with minimum value. However, due to the uncertainty of the pose estimation and mapping algorithms, when the minimum cost point is surrounded by weak footholds, there is a risk that the robot will select a neighbouring potentially risky foothold. To avoid this problem, we filter the map with a moving average of size $(5, 5)$.

In our previous work [11], we proposed a model based on the Efficient Residual Network (ERF) [43] with the addition of the Markov Random Field (MRF) [44] on the output layer.

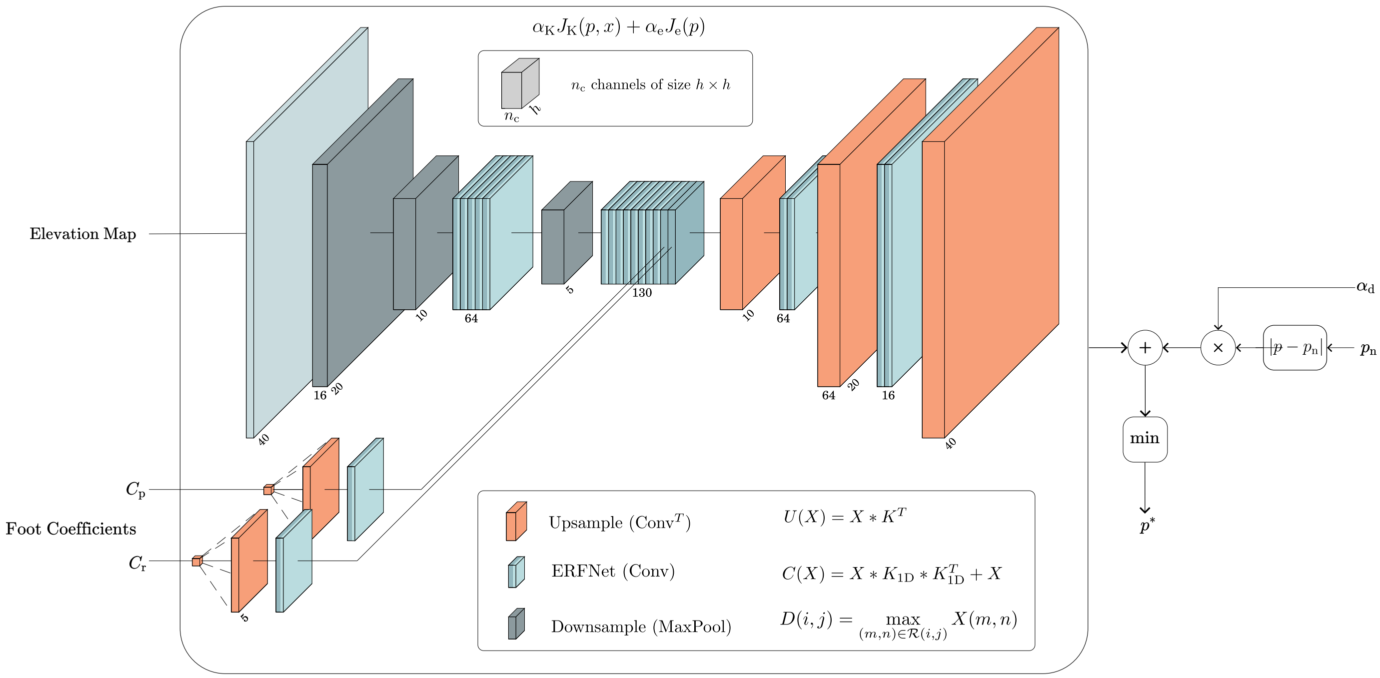


Fig. 5. The optimization process is split in two phases. A CNN computes the cost terms associated with kinematic and foot grip for all the possible footholds, while the distance from the nominal foothold is added later in the process. The result is a grid of cost values from which the minimum cost can simply be computed by iterating over all points. The CNN architecture exploits the latent space to add two additional scalar values as inputs. This enables the network to infer the cost values for every value of the coefficients (C_p, C_r). Downsample (and upsample) convolution layers use stride 2 to halve (and to double) input resolution. ERFNet layers use dilation 1. Finally, on top of every layer we perform Batch Normalization.

This approach has been shown capable of evaluating the foothold cost for the SoftFoot-Q [9], [11], and it employs ERF layers to improve performance on non-specialized hardware. This network works as an image segmentation model, where the output classes are mapped to specific intervals of the cost function.

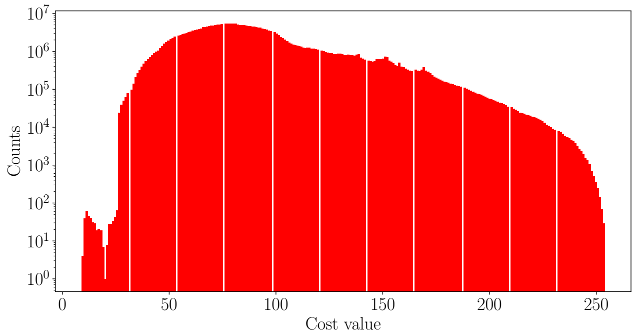
However, this network can only be trained to infer the cost value for a single specific type of foot. To create a model able to predict cost values for any foot described by the pair of coefficients (C_p, C_r), we adopted an architecture shown in Fig. 5. The shape coefficients (C_p, C_r) are upscaled using two deconvolution layers before being concatenated in the latent space [45]. The proposed architecture allows merging multi-dimensional input like an elevation map and low-dimensional parameters that represent the shape of the foot (C_p, C_r). This enables us to use the same model using feet of different curvatures just by changing the parameters during inference, without retraining the network. The resulting model has $2,067,934$ trainable parameters. To enable reproducing the results, we published the implementation of the neural network and a pre-trained model[1].

B. Dataset and preprocessing

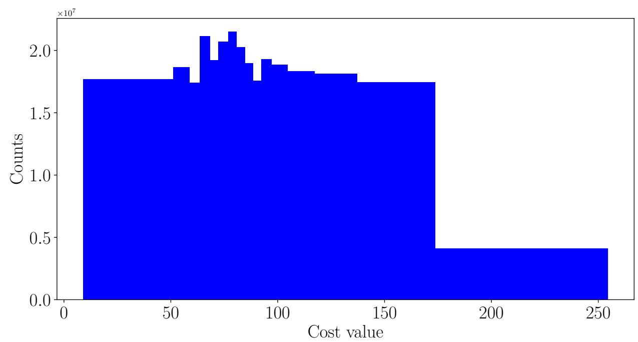
The neural network is trained on a simulated dataset. Firstly, we perform a sampling of the (C_p, C_r) parameters space, obtaining a set of $N_{\text{points}} = 13$ uniformly-spaced points. In each of these point we generated a set of $N_{\text{map}} = 20000$ random elevation maps with the resolution 40×40. The maps are generated by sampling the $12\,\text{m} \times 12\,\text{m}$ elevation map created offline by composing maps obtained during various experiments on the real robots. We also added the flat region, steps with various heights, concavities, and bumps to increase the variation of foothold examples. The maximal height of the obstacles is equal to 0.8 m and the lowest point on the map is 0.07 m below the ground level. We randomly sample the robot's position on the map, considering both the horizontal position and distance from the ground, to generate the dataset. The horizontal orientation of the robot (yaw angle) is randomly chosen from four primary orientations: $n \cdot \pi/2$, with $n \in [0, 1, 2, 3]$. To evaluate the kinematic margin, we use a Gaussian-Mixture (GM)-based method [34], which identifies a set of Gaussian-Mixture functions that represents the constrained regions. The margin is proportional to the minimum distance between the foot position and the boundary of the leg's workspace, evaluated with the GM. The resulting dataset consists of $N_{\text{points}} \times N_{\text{map}} = 260000$ training samples, each one consisting of the value of the input parameters (C_p, C_r), an input elevation map, and the corresponding output costmap.

Due to the random nature of the input costmaps and the uniform sampling of the parameter space, the produced dataset is balanced in the input values, but the same does not hold for the output samples (Fig. 6(a)). In Sec. IV-A we showed that this network can be treated as an image segmentation model, where all the metrics and algorithms used in classification tasks can be employed. Because of this, we can evaluate the balancing of the dataset by computing the Shannon Evenness

[1] https://github.com/tolomeis/learning-footstep-planner-cnn



(a)



(b)

Fig. 6. Histogram of the dataset output values. (a) before the preprocessing and (b) after the preprocessing. The number of samples with the maximum cost (i.e. points outside the leg's workspace) is 4110469.

TABLE I
DATA PREPROCESSING RESULTS.

	Raw dataset	Preprocessed dataset
Number of classes (N)	256	17
SEI	0.69	0.90
Average bin width	1	14.5

Index (SEI)

$$\text{SEI} = -\frac{\sum_{i=1}^{N} \frac{n_i}{n_\text{T}} \log\left(\frac{n_i}{n_\text{T}}\right)}{\log(N)}, \tag{10}$$

where N is the number of classes, n_i is the number of samples in the class i, and n_T is the total number of samples.

The SEI value in Tab. I shows that the dataset is biased towards higher cost values. We perform a preprocessing of the dataset, in order to reduce the number of output classes (cost value intervals) and at the same time to reduce the bias in the dataset. In particular, we change the mapping from cost value intervals into output classes using a non-uniform interval length, resulting in larger classes where the number of samples is lower, and vice-versa, as described in Algorithm 1. The SEI values in Tab. I show that the preprocessed dataset (Fig. 6(b)) is more balanced, at the cost of a lower number of classes (17), and thus a lower resolution on the cost values.

This process does not remove entirely the bias and does not modify the class associated with the maximum cost, i.e. the footholds that are outside the leg's workspace. To address this

Algorithm 1 Bins computation

Require: n_goal, B
  $i \leftarrow 1$
  $B_n[0] \leftarrow B[0]$
  $c \leftarrow 0$
  for $b \in B$ **do**
    $c \leftarrow c + 1$
    if $c \geq n_\text{goal}$ **or** $i == n_\text{max}$ **then**
      $B_n[i] \leftarrow b$
      $c \leftarrow 0$
    end if
  end for
  return

remaining bias, we introduce weights in the loss function to weight the classes during training.

As a last stage, the entire dataset is divided randomly in three slices, which are 70% for training, 15% for validation, and 15% for testing. The split is performed randomly on all the points in the (C_p, C_r) space. This ensures that every point is present in the dataset, but no significant bias is present towards a specific shape of foot. The validation set is used during training to test the current solution and to avoid overfitting the neural network. The test set is used to check the generalization properties of the CNN and present the results.

C. Training

To train the network proposed in this paper, we use a weighted sparse-softmax cross-entropy loss [46]

$$\mathcal{L}(y, \hat{y}, w) = -\frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} \sum_{j=1}^{H} w_j y_{ij} \log(\hat{y}_{ij}), \tag{11}$$

where y is the ground truth label, $\hat{y}$ is the sparse-softmax value of the predicted label, w is the weight term, M is the number of samples, and H is the number of output classes. The term y_{ij} is a binary indicator (0 or 1) that specifies whether sample i belongs to class j. The term $\hat{y}_{ij}$ is the predicted probability that sample i belongs to class j.

The weight term w_j is

$$w_j = \left(\log \frac{n_j}{\sum_i n_i \; + \eta} + w_\text{c}\right)^{-1}, \tag{12}$$

where η and w_c are weight parameters. The weight w_j reflects the imbalance in the number of samples in each class. The higher the number of samples in a class, the lower the weight associated with it, such that the contribution of each class is inversely proportional to the number of samples. This helps compensate for unbalancing in the output values.

We train the network using Stochastic Gradient Descent (SGD) with momentum, and we update the learning rate using the Adam optimizer [47] with Decoupled Weight Decay Regularization [48]. Tab. II shows the training hyperparameters. In particular, we use Weight decay and dropout to avoid incurring in overfitting.

TABLE II
TRAINING HYPERPARAMETERS

Learning Rate	α	5×10^{-4}
Batch Size	B	8
Max number of epochs	E	400
Weight Decay	λ	10^{-4}
Dropout		0.3
Momentum		0.7

D. *Training results*

Fig. 7 shows the validation accuracy of the network during training. We trained the CNN for 300 epochs until the validation loss stabilizes. We stop training before the validation loss drops, which may indicate overfitting and the lack of generalization of the obtained neural network.

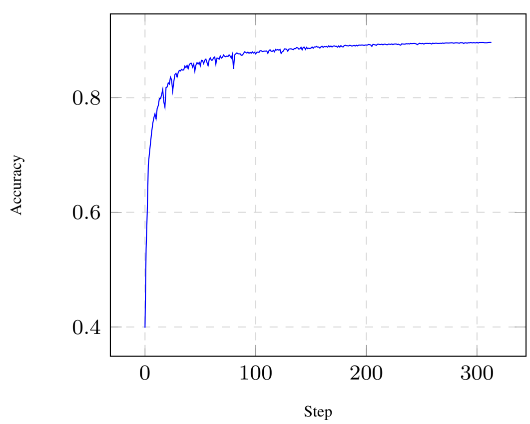


Fig. 7. Validation Accuracy over epochs, during the training process.

It is helpful to look at the Root Mean Square Error (RMSE), computed in the testing dataset as

$$\mathrm{RMSE} = \sqrt{\frac{1}{N_{\mathrm{map}}} \sum_{i=1}^{N_{\mathrm{map}}} \frac{1}{s_{\mathrm{x}}} \sum_{x=1}^{s_{\mathrm{x}}} \frac{1}{s_{\mathrm{y}}} \sum_{y=1}^{s_{\mathrm{y}}} (\hat{c}_{i,x,y} - c_{i,x,y})^2}, \quad (13)$$

where s_{x} and s_{y} indicate the size of the maps, while N_{map} is the number of maps.

Eq. (13) indicates the mean error committed when evaluating the cost, where an error of 1 means that the network is misclassifying between adjacent classes in the cost value-class mapping.

Fig. 8 shows the RMSE for all the values of the input parameters $(C_{\mathrm{p}}, C_{\mathrm{r}})$ that are in the dataset. This value quantifies the ability of the network to generalize on unseen terrain patches for each foot in the dataset.

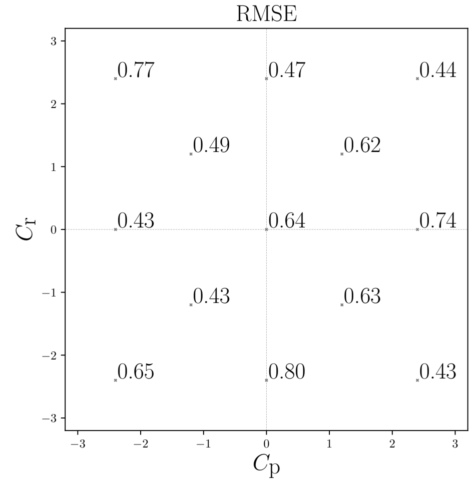
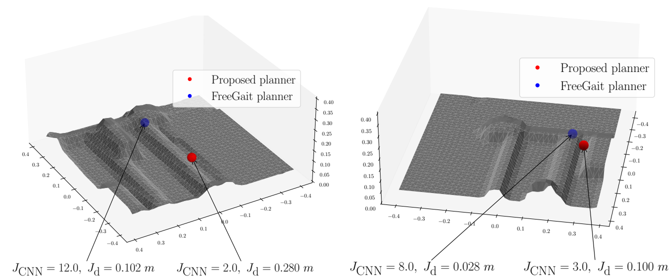
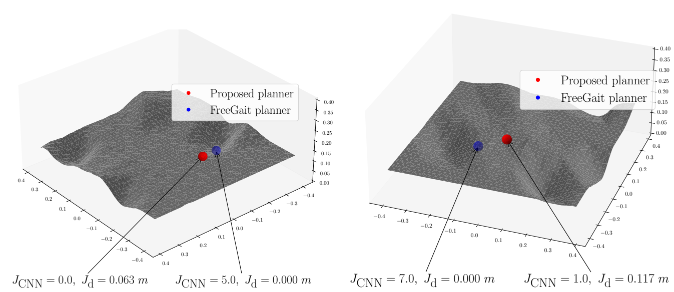


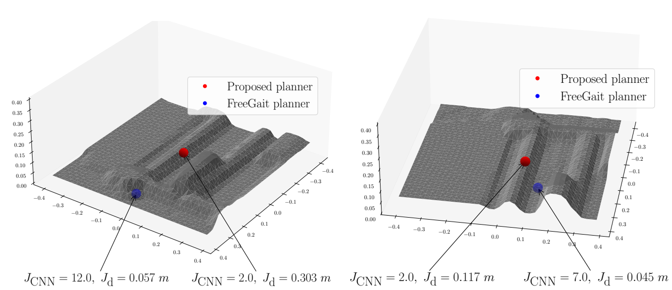
Fig. 8. RMSE in the space of the parameters $(C_{\mathrm{p}}, C_{\mathrm{r}})$.



(a) Two examples of selected footholds for ball feet.



(b) Two examples of selected footholds for flat feet.



(c) Two examples of selected footholds for soft feet.

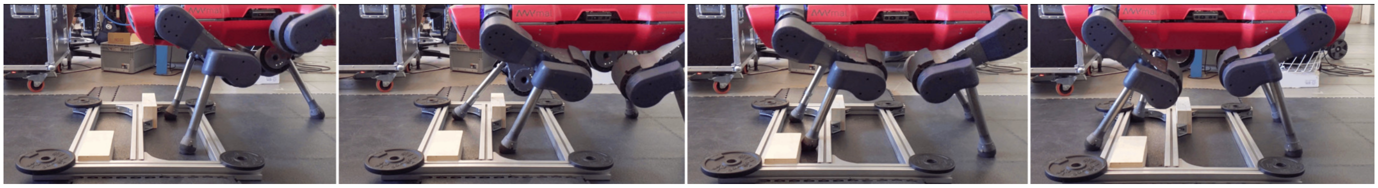
Fig. 9. Samples of footholds acquired in simulations. (a) for ball-shaped feet, (b) for flat feet, (c) for soft feet. For each sample, the values (9) are reported. Axes are in meters.

V. SIMULATION RESULTS

We test the obtained model and the planning algorithm on simulation in a controlled environment, to assess the behaviour of the planner. The obtained results are compared with the cost index of the footholds generated by the state-of-the-art planner FreeGait [19].

A. *Ball feet*

To evaluate the performance with the Ball feet we use coefficients that are related to the radius of the sphere. In



(a)

(b)

Fig. 10. Photo-sequences of the ANYmal C robot equipped with ball-shaped feet walking over a custom-made terrain platform using (a) the proposed algorithm and (b) a state-of-the-art planner [19]

particular, we choose

$$\begin{aligned} C_{\mathrm{p}} &= 2.4 \\ C_{\mathrm{r}} &= 2.4. \end{aligned} \tag{14}$$

With these parameters, the planner is searching for terrain patches that are concave, in which a ball foot can have the maximum contact surface. Fig. 9(a) shows some of the footholds selected in simulation by the planner in this configuration.

B. Flat feet

For the flat feet, the coefficients are set to zero to represent the flat surface of the sole,

$$\begin{aligned} C_{\mathrm{p}} &= 0.0 \\ C_{\mathrm{r}} &= 0.0. \end{aligned} \tag{15}$$

With these parameters, the planner is searching for terrain patches that are flat, in which this type of foot can exert the maximum contact surface. Fig. 9(b) shows some of the footholds selected in simulation by the planner in this configuration.

C. Soft feet SoftFoot-Q

We evaluated the performance of the soft feet SoftFoot-Q [9] using

$$\begin{aligned} C_{\mathrm{p}} &= -2.4 \\ C_{\mathrm{r}} &= 0.0. \end{aligned} \tag{16}$$

This means that the planner is searching for terrain patches that are convex in the longitudinal direction. Fig. 9(c) shows some of the footholds selected in simulation by the planner in this configuration.

VI. Experimental Validation

We evaluate our proposed foothold optimization system on the ANYmal C [2] robot on a custom-made terrain platform and compare it to the state-of-the-art planner that relies on the point-contact foot hypothesis [19]. The platform is designed to have both convex surfaces and holes that act as concave terrain patches. This helps showing our planner's ability to select more suitable footholds for the type of feet in use. We performed four sets of experiments, two with standard ball feet and two with adaptive soft feet. For each foot, we tested both the state-of-the-art planner and our proposed planner. In each experiment, the robot has to walk over the platform using a constant reference speed, equal for all experiments, and starting from the same initial position with respect to the platform. During the experiment, we measured the pitch and roll angles of the base, finding that our planner consistently produces lower oscillations of the robot's base. The state-of-the-art planner is used as-is with both the standard and the adaptive feet. In our planner, we changed the shape coefficients to embed information on the feet shape in the planner. In

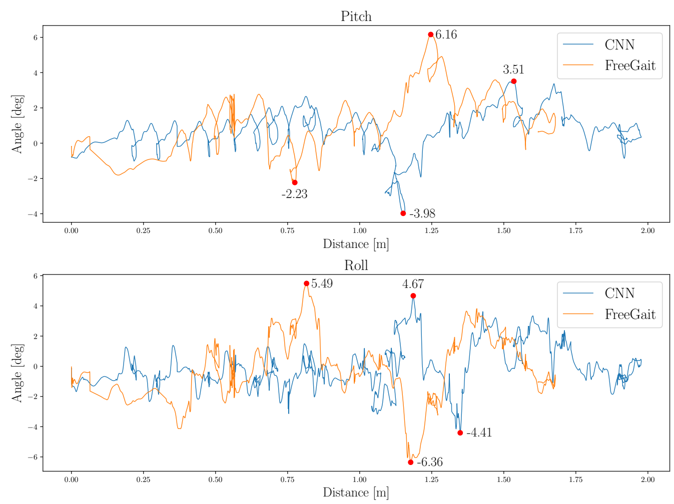


Fig. 11. Comparison of the robot base roll and pitch while using ball-shaped feet by our planner and the state-of-the-art planner [19] over the same terrain. To account for different velocities, the figure is plotted with the travelled distance in meters on the x-axis. For the sake of readability, a moving average is applied at the travelled distance to reduce the effect of the base back-and-forth oscillations typical of a static walk.

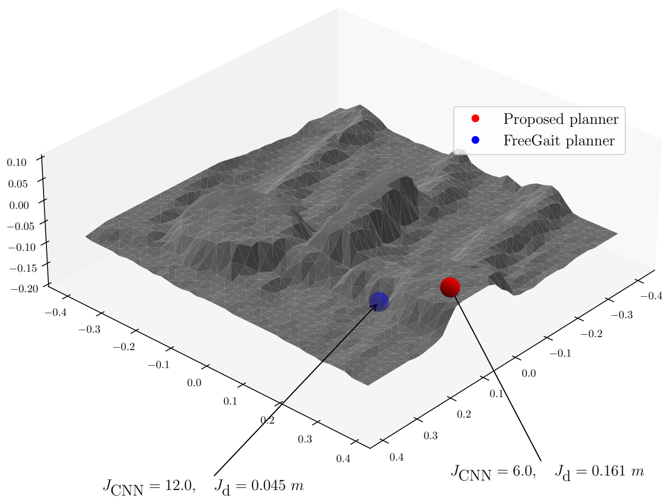


Fig. 12. Example of the foothold positions generated by our planner (red dots) and the state of the art planner (blue dots) in the elevation map for ball feet. Axes are in meters.

particular, we used values presented in Sec. V. We first present the results for the ball feet, and then show the results for the soft feet. For all experiments, we used a distance weighting term $\alpha_{\mathrm{d}} = 35$, and scaling coefficients of $\alpha_{\mathrm{n},i} = 0.25$ and $\alpha_{\mathrm{f}} = 2$, as defined in (6). We also provide an example of the foothold positions generated by our planner in the elevation map.

A. Ball feet

Fig. 12 shows the elevation map with the two foothold positions generated by our planner (red dot) and the one generated by the state-of-the-art planner (blue dot) for comparison. Fig. 10 shows photo-sequences of the ANYmal C robot equipped with ball feet walking over the custom-made terrain platform using the proposed algorithm and a state-of-the-art planner [19]. The stability of the base during the experiment, measured as roll and pitch angles, is improved when using our planner, as shown in Fig. 11. In particular, with [19] the robot registered a maximum pitch and roll angles, in absolute values, respectively of $6.16\,\mathrm{deg}$ deg and $6.36\,\mathrm{deg}$. While using our planner the robot showed maximum values of $3.98\,\mathrm{deg}$ in pitch and $4.67\,\mathrm{deg}$ in roll.

B. Soft feet

Fig. 13 shows a photo-sequence of the ANYmal C robot equipped with the SoftFoot-Q walking over the custom-made terrain platform using the proposed algorithm. Our planner consistently generates footholds with an equal or lower cost index than the state-of-the-art planner. As an example, Fig. 15 shows the elevation map with the two foothold positions generated by our planner (red dot) and the one generated by the state-of-the-art planner (blue dot) with the foothold cost, for comparison. These results are obtained using the parameters described in (16).

During the experiment, with FreeGait planner, the robot got stuck repeating the same step due collision with an obstacle. This did not happen with the proposed algorithm. Moreover, the overall stability of the base during the experiment, measured as roll and pitch angles, is improved when using our planner, as shown in Fig. 14. In particular, with [19] the robot registered a maximum pitch and roll angles, in absolute values, respectively of $5.40\,\mathrm{deg}$ and $7.34\,\mathrm{deg}$. While using our planner the robot showed maximum values of $4.28\,\mathrm{deg}$ in pitch and $3.97\,\mathrm{deg}$ in roll.

C. Outdoor test with SoftFeet-Q

To assess the performance and usability of the planner, we tried the SoftFoot-Q configuration outdoor in two different scenarios. Firstly we tested under direct sunlight (Fig. 16(a)) which is an unfavourable lighting condition for the Intel Realsense D435 [49] used by the robot to perceive the environment. We then tested our planner in a cluttered scenario (Fig. 16(b)) to further validate the applicability of our system. In both scenarios we tested the robot with adaptive feet using our planner against the state-of-the-art planner. Despite the challenging terrain, our system was able to successfully traverse the environment.

We also provide an example of the foothold positions generated by our planner in the elevation map. Fig. 17 shows the elevation map with the two foothold positions generated by our planner (red dot) and the one generated by the state-of-the-art planner (blue dot) for comparison.

VII. Discussion

A. CNN performance

The performance of the neural network can be evaluated from Fig. 8. The RMSE changes throughout the parameters space, this could be due to biases in the dataset that emerge only from specific values of the parameters. The RMSE is always less than 1, this means that most of the errors are committed between adjacent classes in the output. We found that this behaviour does not significantly affect the performance of the foothold selection algorithm.

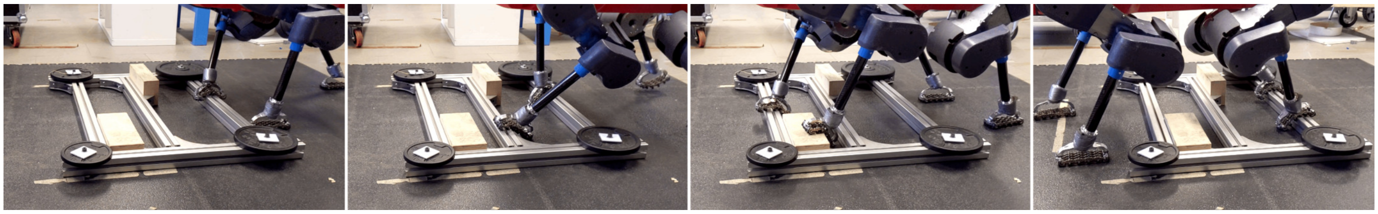
B. Planner performance

Among the goals of a static walk controller, is to minimize roll and pitch oscillations of the base during the walk sequence [19]. The results in Fig. 11 and 14 show that using the proposed planner the maximum values of pitch and roll over the same terrain are reduced, for both adaptive soft feet and ball-shaped feed. This holds even in the case of outdoor testing with unfavourable lighting conditions.

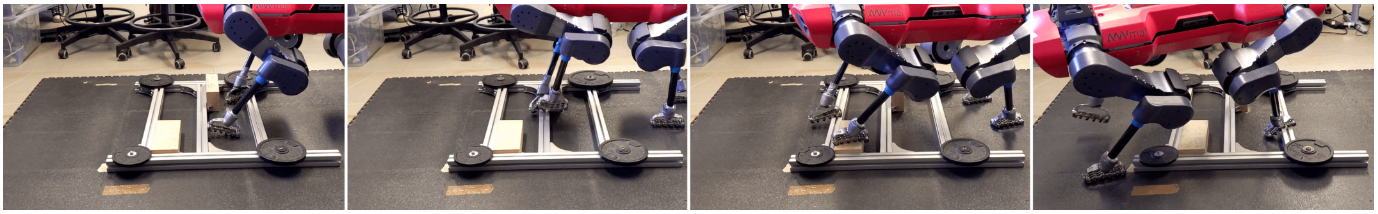
The elevation map in Fig. 12 shows that the planner is able to search for a foothold with the correct reference shape, that is concave for the ball feet. The same is true for the SoftFoot-Q, as visible in Fig. 15 where the planner selects a foothold that is convex in the longitudinal direction.

C. Comparison with state-of-the-art planner

During execution, our planner was able to perform inference in 8.5 ms, evaluating 1600 candidate points. In contrast, [19] takes $0.5 - 2.5$ ms to evaluate a candidate point. The increased



(a)



(b)

Fig. 13. Photo-sequences of the ANYmal C robot equipped with SoftFoot-Q walking over a custom-made terrain platform using (a) the proposed algorithm and (b) a state-of-the-art planner [19]

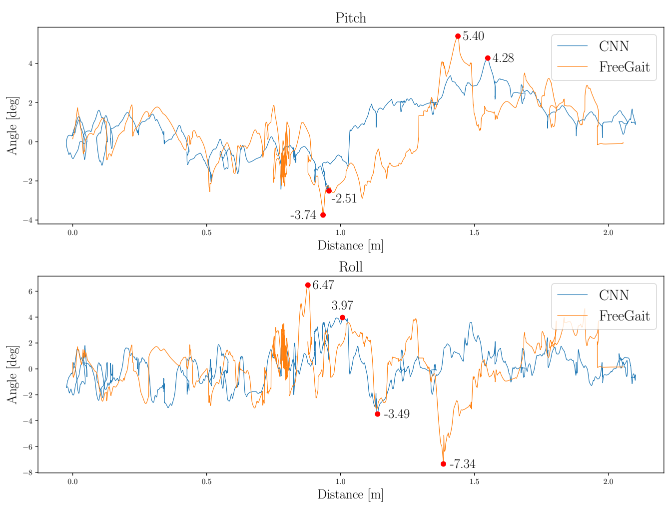


Fig. 14. Comparison of the robot base roll and pitch while using the adaptive feet SoftFeet-Q [9] by our planner and the state of the art planner [19] over the same terrain. To account for different velocities, the figure is plotted with the travelled distance in meters on the x-axis. For the sake of readability, a moving average is applied at the travelled distance to reduce the effect of the base back-and-forth oscillations typical of a static walk.

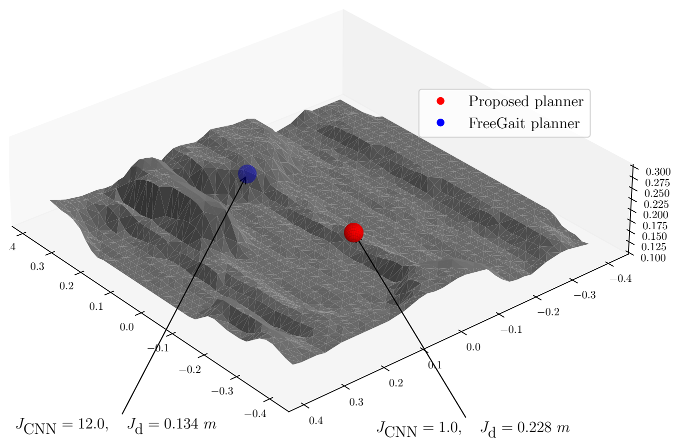


Fig. 15. Example of the foothold positions generated for the adaptive foot [9] by our planner (red dots) and the state of the art planner (blue dots) in the elevation map.

computational time resulted in lower velocity tracking for our planner. For all the experiments, the reference velocity for the statically stable FreeGait [19] is $0.400\,\mathrm{m/s}$. Using the proposed planner, the robot walked at an average velocity of $0.0197\,\mathrm{m/s}$, while with [19] the average velocity is $0.0288\,\mathrm{m/s}$.

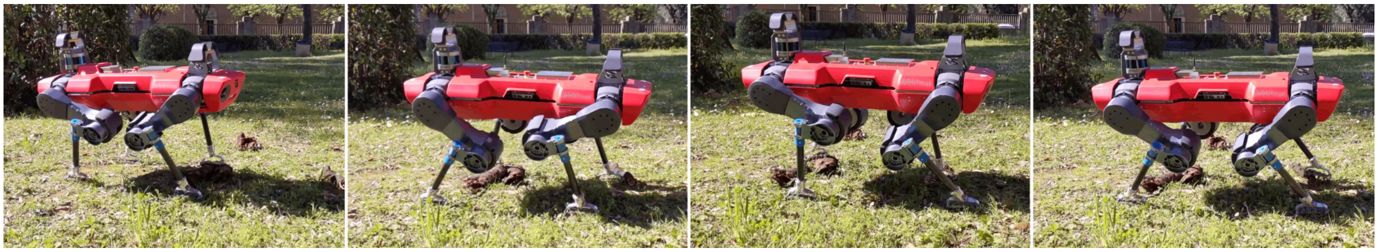
D. Limitations

The elevation map used in the system conveys information only about the terrain's shape, and not about its material properties and compliance. Therefore, the robot is unable to distinguish between different obstacles that share similar geometric features, such as a root that is firmly fixed to the ground and a rock with a similar shape. This may lead to suboptimal foothold planning. To address this issue, future research should explore the integration of additional sensing modalities that can provide information about the terrain's compliance. Furthermore, to address the lower velocity tracking, the reference velocity could be used to augment the state of the CNN, which has been shown to help increase the overall velocity tracking [50].

Moreover, the development of advanced machine learning algorithms that can process multi-modal sensory data in real-time may enable more robust and adaptive navigation strategies.

VIII. Conclusions

In this paper, we proposed a generalization of our previous method [11] to select footholds for legged robots equipped



(a)

(b)

Fig. 16. Photo-sequence of the ANYmal C robot equipped with SoftFoot-Q adaptive feet [9] walking in an outdoor natural environment (a) and in a rough cluttered scenario (b).

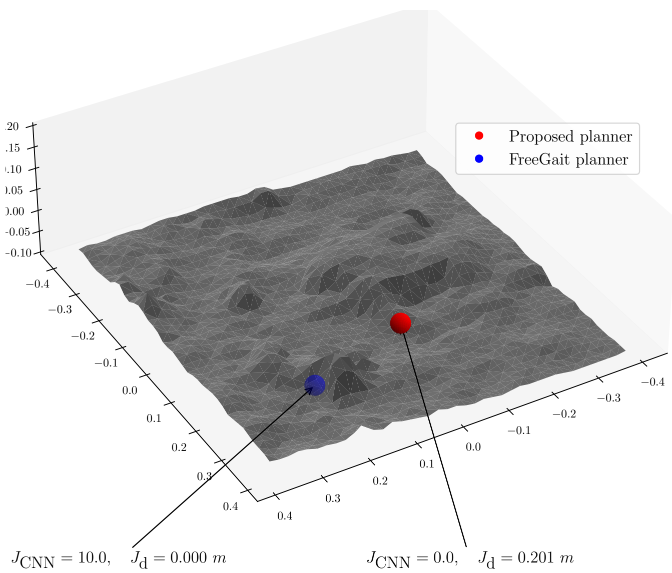


Fig. 17. Example of the foothold positions generated for the adaptive feet [9] by our planner (red dots) and the state-of-the-art planner (blue dots) in the elevation map during an outdoor test.

with different types of feet. The method utilizes a Convolutional Neural Network to predict the cost of candidate footholds. The network predicts the cost based on a polynomial approximation of the foot sole and searches for matching terrain patches in the surroundings of the robot to maximize contact surface and grip. The cost computed by the network takes also into account the kinematic model of the robot, self-collisions and collisions of the leg with the terrain. The training data is generated in a simulated environment, by generating a set of random terrain elevation maps and computing the cost maps. This process is repeated for a set of predetermined coefficients for the foot shape, in order to avoid bias in the dataset. Finally, the proposed neural network is used on-line to plan footholds for the robot ANYmal-C. The method is experimentally validated on a terrain platform and in outdoor natural environments, and it is compared with a state-of-the-art foothold planner. The proposed algorithm is able to select high-grip footholds for various type of feet, and is robust to outdoor environments and harsh lighting conditions.

In future works, we will focus on integrating additional sensing capabilities to the planner in order to consider the terrain material and compliance, to avoid stepping on slippery obstacles. The ability to change feet parameters online could also be exploited to fully take advantage of the capabilities of adaptive feet [51], changing online the preferred terrain outline from flat to convex depending on the characteristics of the terrain. Furthermore, we will also study the possibility of integrating this foothold selection approach to other learning-based base trajectory and control algorithms.

REFERENCES

[1] F. Angelini, P. Angelini, C. Angiolini, S. Bagella, F. Bonomo, M. Caccianiga, C. Della Santina, D. Gigante, M. Hutter, T. Nanayakkara *et al.*, “Robotic monitoring of habitats: The natural intelligence approach,” *IEEE Access*, 2023.

[2] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, “ANYmal - a highly mobile and dynamic quadrupedal robot,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 38–44.

[3] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, “Design of HyQ – a hydraulically and electrically actuated quadruped robot,” *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.

[4] F. Negrello, M. Garabini, M. Catalano, P. Kryczka, W. Choi, D. Caldwell, A. Bicchi, and N. Tsagarakis, “WALK-MAN humanoid lower body design optimization for enhanced physical performance,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1817–1824.

[5] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, J.-P. Laumond, L. Marchionni, H. Tome, and F. Ferro, “TALOS: A new humanoid research platform targeted for industrial applications,” in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 689–695.

[6] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz, "Anytime search-based footstep planning with suboptimality bounds," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pp. 674–679.

[7] D. Kanoulas, A. Stumpf, V. S. Raghavan, C. Zhou, A. Toumpa, O. Von Stryk, D. G. Caldwell, and N. G. Tsagarakis, "Footstep Planning in Rough Terrain for Bipedal Robots Using Curved Contact Patches," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4662–4669.

[8] G. Valsecchi, R. Grandia, and M. Hutter, "Quadrupedal Locomotion on Uneven Terrain With Sensorized Feet," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1548–1555, 2020.

[9] M. G. Catalano, M. J. Pollayil, G. Grioli, G. Valsecchi, H. Kolvenbach, M. Hutter, A. Bicchi, and M. Garabini, "Adaptive feet for quadrupedal walkers," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 302–316, 2021.

[10] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic Terrain Mapping for Mobile Robots With Uncertain Localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3019–3026.

[11] J. Bednarek, N. Maalouf, M. J. Pollayil, M. Garabini, M. G. Catalano, G. Grioli, and D. Belter, "CNN-based Foothold Selection for Mechanically Adaptive Soft Foot," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10 225–10 232.

[12] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal, "Learning locomotion over rough terrain using terrain templates," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 167–172.

[13] M. Hutter, H. Sommer, C. Gehring, M. Hoepflinger, M. Bloesch, and R. Siegwart, "Quadrupedal locomotion using hierarchical operational space control," *The International Journal of Robotics Research*, vol. 33, no. 8, pp. 1047–1062, 2014.

[14] M. Camurri, M. Fallon, S. Bazeille, A. Radulescu, V. Barasuol, D. G. Caldwell, and C. Semini, "Probabilistic Contact Estimation and Impact Detection for State Estimation of Quadruped Robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1023–1030, 2017.

[15] R. Käslin, H. Kolvenbach, L. Paez, K. Lika, and M. Hutter, "Towards a Passive Adaptive Planar Foot with Ground Orientation and Contact Force Sensing for Legged Robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2707–2714.

[16] C. Piazza, C. Della Santina, G. M. Gasparri, M. G. Catalano, G. Grioli, M. Garabini, and A. Bicchi, "Toward an adaptive foot for natural walking," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 1204–1210.

[17] V. Barasuol, M. Camurri, S. Bazeille, D. G. Caldwell, and C. Semini, "Reactive trotting with foot placement corrections through visual pattern classification," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5734–5741.

[18] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, pp. 811–818.

[19] P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter, "Robust Rough-Terrain Locomotion with a Quadrupedal Robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5761–5768.

[20] M. A. Hoepflinger, M. Hutter, C. Gehring, M. Bloesch, and R. Siegwart, "Unsupervised identification and prediction of foothold robustness," in *2013 IEEE International Conference on Robotics and Automation*, pp. 3293–3298.

[21] Y. Gu, Y. Cheng, C. L. P. Chen, and X. Wang, "Proximal policy optimization with policy feedback," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 7, pp. 4600–4610, 2022.

[22] L. Pang, Z. Cao, J. Yu, P. Guan, X. Rong, and H. Chai, "A visual leader-following approach with a t-d-r framework for quadruped robots," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 4, pp. 2342–2354, 2021.

[23] X. Li, H. Gao, F. Zha, J. Li, Y. Wang, Y. Guo, and X. Wang, "Learning the Cost Function for Foothold Selection in a Quadruped Robot," *Sensors*, vol. 19, p. 1292, 2019.

[24] D. Belter, J. Bednarek, H.-C. Lin, G. Xin, and M. Mistry, "Single-shot Foothold Selection and Constraint Evaluation for Quadruped Locomotion," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7441–7447.

[25] O. Villarreal, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. G. Caldwell, and C. Semini, "Fast and Continuous Foothold Adaptation for Dynamic Locomotion through CNNs," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2140–2147, 2019.

[26] L. Chen, S. Ye, C. Sun, A. Zhang, G. Deng, T. Liao, and J. Sun, "CNNs based Foothold Selection for Energy-Efficient Quadruped Locomotion over Rough Terrains," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1115–1120.

[27] S. Omar, L. Amatucci, V. Barasuol, G. Turrisi, and C. Semini, "Safesteps: Learning safer footstep planning policies for legged robots via model-based priors," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, 2023, pp. 1–8.

[28] X. Wu, J. Li, L. Liu, and D. Tao, "The visual footsteps planning system for exoskeleton robots under complex terrain," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 8, pp. 5149–5160, 2023.

[29] Q. Yao, J. Wang, D. Wang, S. Yang, H. Zhang, Y. Wang, and Z. Wu, "Hierarchical Terrain-Aware Control for Quadrupedal Locomotion by Combining Deep Reinforcement Learning and Optimal Control," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4546–4551.

[30] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, p. eabc5986.

[31] D. Song, P. Fernbach, T. Flayols, A. D. Prete, N. Mansard, S. Tonneau, and Y. J. Kim, "Solving footstep planning as a feasibility problem using l1-norm minimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5961–5968, 2021.

[32] B. Mishra, D. Calvert, S. Bertrand, J. Pratt, H. E. Sevil, and R. Griffin, "Efficient terrain map using planar regions for footstep planning on humanoid robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 8044–8050.

[33] A. Stumpf and O. von Stryk, "A universal footstep planning methodology for continuous walking in challenging terrain applicable to different types of legged robots," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 420–10 427.

[34] D. Belter, "Efficient Modeling and Evaluation of Constraints in Path Planning for Multi-Legged Walking Robots," *IEEE Access*, vol. PP, pp. 1–1, 2019.

[35] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[36] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, D. E. Rumelhart and J. L. Mcclelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[37] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features." in *NIPS*, 2018.

[38] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 9992–10 002.

[39] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[40] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM computing surveys (CSUR)*, vol. 54, no. 10s, pp. 1–41, 2022.

[41] D. Karimi, H. Dou, and A. Gholipour, "Medical image segmentation using transformer networks," *IEEE Access*, vol. 10, pp. 29 322–29 332, 2022.

[42] G. Xu, X. Zhang, X. He, and X. Wu, "Levit-unet: Make faster encoders with transformer for medical image segmentation," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer, 2023, pp. 42–53.

[43] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2018.

[44] C. Li and M. Wand, "Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2479–2486.

[45] K. Piaskowski, R. Staszak, and D. Belter, "Generate What You Can't See - a View-dependent Image Generation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5903–5909.

[46] S. Sun, Z. Zhang, B. Huang, P. Lei, J. Su, S. Pan, and J. Cao, "Sparse-softmax: A Simpler and Faster Alternative Softmax Transformation," *arXiv preprint arXiv:2112.12433*, 2021.

[47] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
[48] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization,” *arXiv preprint arXiv:1711.05101*, 2017.
[49] Y. W. Kuan, N. O. Ee, and L. S. Wei, “Comparative study of intel r200, kinect v2, and primesense rgb-d sensors performance outdoors,” *IEEE Sensors Journal*, vol. 19, no. 19, pp. 8741–8750, 2019.
[50] D. Esteban, O. Villarreal, S. Fahmi, C. Semini, and V. Barasuol, “On the influence of body velocity in foothold adaptation for dynamic legged locomotion via cnns,” in *International Conference on Climbing and Walking Robots (CLAWAR), Moscow, Russia*, 2020, pp. 353–360.
[51] C. Piazza, C. D. Santina, M. G. Catalano, G. Grioli, and A. Bicchi, “Analytical model and experimental testing of the softfoot: an adaptive robot foot for walking over obstacles and irregular terrains,” 2024.

Simone TOLOMEI received the B.S. degree in electrical engineering and the M.S. degree (cum laude) in robotics and automation engineering from the University of Pisa, Pisa, Italy, in 2019 and 2022, respectively, where he is currently pursuing the Ph.D. degree in Robotics and Intelligent machines. His main research interests are reinforcement learning for robotics, mobile manipulation, and locomanipulation.

Dominik BELTER received a Ph.D. degree in robotics from Poznan University of Technology (PUT) in 2012. He received a DSc. degree in robotics from the same University and has been an Associate Professor since 2021. He spent a year working as a postdoc in the Intelligent Robotics Laboratory at the University of Birmingham from 2013 to 2016. Dominik Belter has been taking part as an investigator in 3 EC and 8 national projects leading four of them. He is currently the Head of the Institute of Robotics and Machine Intelligence at PUT. His research interests include walking robots, machine learning, vision, robot manipulation.

Jakub BEDNAREK graduated from Poznan University of Technology (PUT) receiving BSc and MSc in Computer Science. He is a Ph.D. student at the Faculty of Computer Science at PUT, conducting research on artificial intelligence, and currently, he is a Research Assistant in the Institute of Robotics and Machine Intelligence at PUT. His research focuses on the development of artificial intelligence methods, including research on how to explain their operation, and its application in robotics.

Franco ANGELINI received the B.S. degree in computer engineering in 2013 and M.S. degree (cum laude) in automation and robotics engineering in 2016 from the University of Pisa, Pisa, Italy. The University of Pisa also granted him a Ph.D. degree (cum laude) in robotics in 2020. Franco is currently an Assistant Professor at the University of Pisa. His main research interests are control of soft robotic systems, robotic environmental monitoring, grasping, and impedance planning.

Manolo GARABINI received the graduation degree in mechanical engineering and the Ph.D. degree in robotics from the University of Pisa, Pisa, Italy, in 2010 and 2014, respectively. He is currently a Professor at the University of Pisa. He is currently the Principal Investigator with the THING H2020 EU Research Project for the University of Pisa and the Coordinator of the Dysturbance H2020 Eurobench Subproject. Finally, he is the Coordinator of the H2020 EU Research Project Natural Intelligence. His main research interests include the design, planning, and control of soft adaptive robots.