SafeSteps: Learning Safer Footstep Planning Policies for Legged Robots via Model-Based Priors

Shafeef Omar, Lorenzo Amatucci, Victor Barasuol, Giulio Turrisi, Claudio Semini

Abstract— We present a footstep planning policy for quadrupedal locomotion that is able to directly take into consideration a-priori safety information in its decisions. At its core, a learning process analyzes terrain patches, classifying each landing location by its kinematic feasibility, shin collision, and terrain roughness. This information is then encoded into a small vector representation and passed as an additional state to the footstep planning policy, which furthermore proposes only safe footstep location by applying a masked variant of the Proximal Policy Optimization algorithm. The performance of the proposed approach is shown by comparative simulations and experiments on an electric quadruped robot walking in different rough terrain scenarios. We show that violations of the above safety conditions are greatly reduced both during training and the successive deployment of the policy, resulting in an inherently safer footstep planner. Furthermore, we show how, as a byproduct, fewer reward terms are needed to shape the behavior of the policy, which in return is able to achieve both better final performances and sample efficiency. Video results at https://sites.google.com/view/safe-steps-rl.

I. INTRODUCTION

Quadruped robots are increasingly entering new application domains, such as inspection, construction, and rescue. To be valuable in such real-world scenarios, robots should be able to map the surroundings via their onboard sensors, such as cameras or lidar, in order to choose the best footstep landing location autonomously and avoid stepping on potentially unsafe regions (e.g., collapsing surfaces, stairs edges, etc.) which can inevitably bring the systems to failure. Along with visual information, an additional component needed to traverse such harsh scenarios is the ability to perform a reactive stepping strategy for recovery, e.g., modifying the robot's footstep at need in the presence of an external disturbance. This requirement further increases the complexity of the problem, given the need to explicitly consider the robot's dynamics along with vision. In both the case of planning and recovery, the resulting chosen footsteps need to be kinematically feasible to be actuated by the robot, and the resulting motion needs to be collision-free in order to avoid additional ground reaction forces that can destabilize the motion.

Given the difficulty of achieving all these conditions concurrently, footstep placement techniques are still heavily studied in the robotic community. In [2], the authors presented one of the first applications of terrain awareness locomotion to enhance the traversing capabilities of a
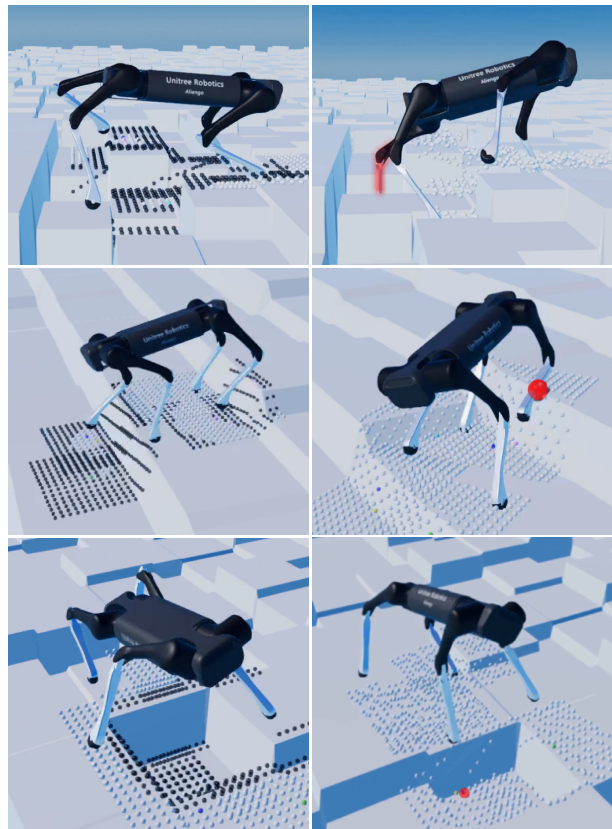
Fig. 1. Snapshots of Aliengo [1] walking over irregular terrains with our method (left) where footholds that bring safety violations are prohibited at the action level (black dots), and with a simpler RL policy (right) where these violations are only discouraged during training. In the last case, starting from the top, we depict a kinematic violation, a shin collision, and a slippage due to the foot being placed over a terrain edge.

quadruped robot. A set of heuristics were analyzed to select the best foothold locations, discarding possible candidates near edges, slopes, or holes. This idea was then extended in [3] by considering body collisions and introducing a supervised learning regression technique to ease the classification problem. Still, given the needed computational time, the method was unsuitable for reactive and agile motion. This reactiveness was then achieved by performing regression via fast Convolutional Neural Networks in [4]. In this case, candidate footsteps were chosen by analyzing terrain features, the kinematic limits of the robot, and possible shin collision. Given the achieved computational speed, the robot was able to withstand various disturbances applied during its motion. This method, which we refer to as Visual Foothold Adaptation (VFA), will be used as the core component of this work.

More recently, optimization-based control techniques, such as Model Predictive Control [5] (MPC), were coupled with vision-based footstep correction [6], [7], [8], [9], [10] to obtain highly-dynamic and optimal motions. Noticeably, in [7] and [8], the footstep position was considered as an additional control variable to be further optimized by the MPC, taking into consideration a convex constraint obtained by performing a vision-based segmentation technique.

Although effective in obtaining good performance and robustness, the above optimization-based approaches have two main limitations from the viewpoint of this paper. First, classifying the terrain by employing different criteria, such as edges analysis, kinematics, and collision, can create a non-convex foothold constraint. This can be solvable by employing a Mixed Integer Program, which, however, can be hard to run in real-time [11]. In fact, standard solutions, as in [7], only consider the nearest convex constraint. Second, usually, disturbances are not explicitly estimated. In both cases, the obtained footstep candidate can be suboptimal.

To bypass the above limitations, Reinforcement Learning (RL) techniques can be viable solutions for achieving such optimality, since nonlinear and non-convex problems can be tractable by employing these methods. In [12], a footstep policy was learned from scratch together with a whole-body controller in a hierarchical fashion. Along the same lines, the authors in [13] proposed a similar architecture for the case of quadrupedal locomotion, meanwhile, in [14], [15] RL was proposed to modify leg frequencies, and to generate foot position or joint residuals, showing impressive real-world results. In [16], RL was employed to add robustness to a model-based controller by estimating external disturbances. Finally, similar to our approach, in [17] RL was used to learn a footstep planning policy while a model-based controller was employed to track the generated references. In the same way, here we employ an MPC controller to devise the robot motion, concentrating only on the footstep planning task.

However, all these works do not explicitly consider safety in the learned policy, which can, in some scenarios, still fail given the black-box nature of the approach. This paper is aimed to directly take this aspect into consideration.

In this paper, we build upon our visual foothold adaptation (VFA) technique [4] to devise a safer footstep planning policy for quadrupedal robots. The basic idea is to outsource the fulfillment of user-defined safety constraints by mixing the above model-based module with a specific policy architecture in order to guarantee with high probability their satisfaction. This is done by using the model-based priors given by the VFA, which perform terrain analysis and check for kinematic limits and shin collisions, both for defining the policy's input and modifying its output. The latter is done by employing a masking procedure over Proximal Policy Optimization (PPO) [18], a state-of-the-art model-free reinforcement learning algorithm, constraining the choice of the available actions to improve safety (Fig. 1). As a result, fewer reward terms (the ones that we relate to safety) are needed for shaping the behavior of the policy, which in return optimizes only dynamic conditions and terminal violations.

A. Contributions

The main contributions of this work are:

- the design of a footstep planning policy that incorporates a-priori safety information in its decision. To the best of the author's knowledge, no prior work addresses this specific topic for footstep planning in the context of RL;
- an extensive evaluation which shows that outsourcing safety translates into fewer reward terms for shaping the behavior of the policy, enabling the possibility of achieving better performances with lower sample complexity. Furthermore, we show that, with our approach, violations of the user-defined safety constraints are greatly reduced during the entire learning transient.

As a comparison, first we show the benefit of adopting an RL policy instead of simpler heuristics for the choice of the foothold locations, and second we provide an extensive evaluation of the performance of our method with respect to standard RL approaches that minimize the same safety conditions in the reward.

B. Outline

The paper is organized as follows. Section II introduces the footstep planning problem for quadrupedal robots, highlighting how vision-based and precise footstep placement can be used to perform locomotion in the presence of rough terrain. The proposed footstep planning policy is then presented in Sec. III, where we first discuss the learning procedure for the a-priori safety constraints, the policy architecture, and finally, the model-based controller adopted in this work. In Sec. IV, we report comparative simulation results, meanwhile, some general conclusions about the approach are drawn in Sec. V.

II. PROBLEM FORMULATION

In this work, we want to learn a footstep planning policy that is able to choose optimal and safe foot landing locations in order to maximize the robot's performance. Commonly, in the case of blind locomotion and flat terrain, this is done by computing the next footstep $\boldsymbol{p}_{\text{foot}} \in \mathbb{R}^3$ as

$$\boldsymbol{p}_{\text{foot}} = \boldsymbol{p}_{\text{hip}} + \frac{1}{2}T_{\text{stance}}(\boldsymbol{v}_{\text{com}}^{\text{usr}} + \boldsymbol{w}_{\text{com}}^{\text{usr}} \times {}^{\text{b}}\bar{\boldsymbol{p}}_{\text{hip}}) \quad (1)$$

by employing a modified version of the Raibert's heuristic [19], where $\bar{\boldsymbol{p}}_{\text{hip}} \in \mathbb{R}^3$ is the distance between the hip and center of the base, T_{stance} is a scalar representing the stance duration, and $\boldsymbol{p}_{\text{hip}} \in \mathbb{R}^3$ is the projection of the hip on the ground; finally, $\boldsymbol{v}_{\text{com}}^{\text{usr}}, \boldsymbol{w}_{\text{com}}^{\text{usr}} \in \mathbb{R}^3$ are respectively the linear and desired angular speed of the Center of Mass (CoM). All these quantities, if not explicitly specified, are expressed in the world frame W, meanwhile ${}^{\text{b}}\bar{\boldsymbol{p}}_{\text{hip}}$ is expressed in the base frame B.

A drawback of the above equation is that it is only suitable for blind locomotion, and does not consider disturbances, limiting the robot's performance and resilience in a practical scenario. For performing vision-based footstep planning, in this work, we build upon the VFA module which computes a set of heuristic criteria to evaluate the safety of each possible
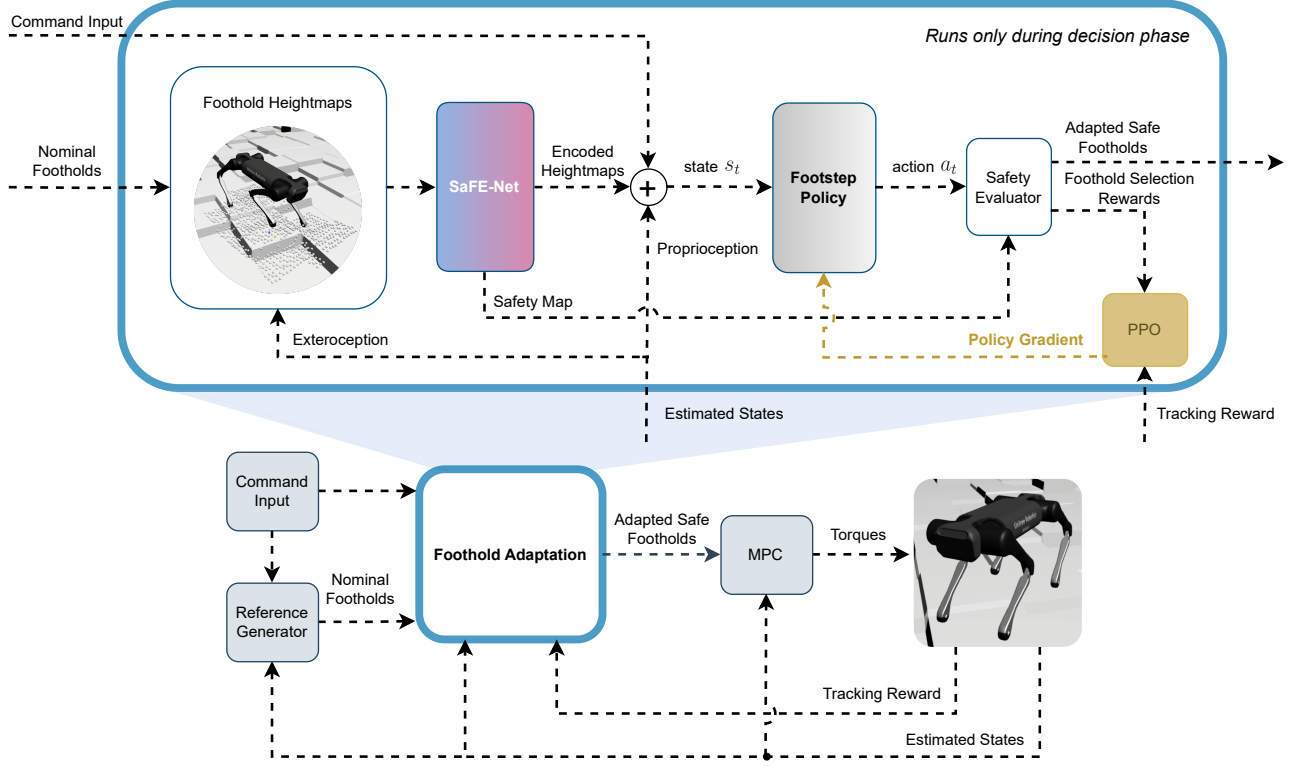
Fig. 2. Block diagram of the proposed approach. Starting from the left, the user commands a velocity input, which is then tracked by an MPC controller and an optimal footstep policy. The last, described in Sec. III, considers the safety information provided by SaFe-Net both in its input state and at the output level.

landing location in the vicinity of the proposed foothold p_{foot} in (1).

The VFA takes a tuple T as an input, evaluates it based on multiple criteria, and outputs a boolean matrix $\boldsymbol{\mu}_{\text{safe}}$ representing which locations are considered safe. The input tuple for a generic leg in swing is defined as

$$T = \left(\boldsymbol{H}, z, \phi, {}^b\boldsymbol{v}_{\text{com}}, {}^b\boldsymbol{w}_{\text{com}} \right)$$

where $\boldsymbol{H} \in \mathbb{R}^{h_x \times h_y}$ is the heightmap of dimensions h_x and h_y centered around p_{foot}, and ${}^b\boldsymbol{v}_{\text{com}}, {}^b\boldsymbol{w}_{\text{com}} \in \mathbb{R}^3$ are respectively the linear and angular speed of the CoM in the base frame. Furthermore, $\phi \in [0,1]$ is a scalar representing the gait phase, which continually increases from 0 to 1 as the leg lifts off until the next touchdown, while $z \in \mathbb{R}$ is the hip height. Each cell of the heightmap $\boldsymbol{H}$ contains the terrain height with respect to the hips of the robot.

In this work, we only consider the following heuristic criteria: Terrain Roughness (TR), Leg Collision (LC) and Kinematic Feasibility (KF), which are detailed below:

a) Terrain Roughness (TR): this criterion checks edges in the heightmap that are unsafe for the robot to step on. For each candidate foothold p_c in $\boldsymbol{H}$, we evaluate the height difference relative to its neighboring footholds, and we consider a threshold to decide whether p_c is safe or not.

b) Leg Collision (LC): this criterion selects footholds that do not result in a leg collision with the terrain at touch-

down. To do so, we create a bounding region around the leg configuration corresponding to the candidate foothold p_c and the future hip location. Then, if the bounding region collides with the terrain, we discard the candidate foothold.

c) Kinematic Feasibility (KF): this criterion selects kinematically feasible footholds, checking whether a candidate foothold p_c will result in a trajectory that remains within the workspace of the leg during the entire gait cycle. For this, we check if the candidate foothold p_c is within the workspace of the leg during touchdown and next lift-off.

Each criterion C outputs a boolean matrix $\boldsymbol{\mu}_C$, and the final output $\boldsymbol{\mu}_{\text{safe}}$ is computed by performing the element-wise logical AND ($\wedge$) of all the criteria, such as

$$\boldsymbol{\mu}_{\text{safe}} = \boldsymbol{\mu}_{\text{TR}} \wedge \boldsymbol{\mu}_{\text{LC}} \wedge \boldsymbol{\mu}_{\text{KF}}.$$

The output $\boldsymbol{\mu}_{\text{safe}} \in \mathbb{R}^{h_x \times h_y}$ is a boolean matrix with the same size as the input heightmap $\boldsymbol{H}$ and indicates the safety of the candidate footholds.

In previous work [6], we chose as the best foothold the one *nearest* to the nominal p_{foot} flagged as safe in $\boldsymbol{\mu}_{\text{safe}}$, but as detailed in the next section, this choice can result in a suboptimal robot behavior.

III. THE PROPOSED APPROACH

The presence of the robot's dynamics and disturbances affect considerably the footstep planning procedure. First,
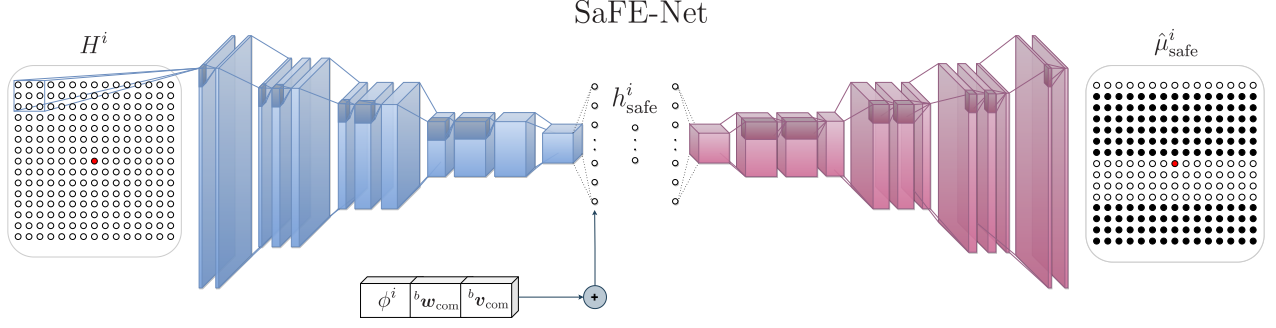
Fig. 3. Image representing the architecture of SaFE-Net (Sec. III-B). For a generic leg in swing i, on the left, we represent the heightmap $\boldsymbol{H}^i$, while on the right its prediction $\hat{\boldsymbol{\mu}}^i_{\text{safe}}$. The center of the heightmap, $\boldsymbol{p}^i_{\text{foot}}$ coming from (1), is indicated with a red circle.

only applying the VFA module and choosing the nearest foothold location to $\boldsymbol{p}_{\text{foot}}$ can result in worse tracking performance of the user-commanded twist. For example, in the case of stair climbing, the robot could choose to step multiple times in the same area before proceeding. Second, even when the chosen foothold is optimal, the robot's motion can be destabilized by the presence of external disturbances, and another better step location might exist to aid a recovery maneuver.

In this section, we describe our safe-footstep planning policy that is able to reduce the effect of the above conditions. At its core, there is a Neural Network (Sec. III-A) which classifies the heightmap patches $\boldsymbol{H}$ by learning the evaluation of the heuristic criteria described in Sec. II via supervised learning. The network employs an encoder-decoder structure to confine the classification information into a smaller subspace $\boldsymbol{h}_{\text{safe}}$, which is then fed as input to an RL policy (Sec. III-B), which continuously proposes the best safe landing location by applying a masked variant of PPO. Finally, an MPC (Sec. III-C) realizes the robot's motion.

A block diagram of the proposed approach is shown in Fig. 2.

A. Visual Foothold Classification and Encoding

A common approach used to embed visual information inside an RL policy is by employing a separate neural network with an encoder-decoder architecture [17]. This step is usually performed to compress the information into a smaller subspace to simplify the learning process, which otherwise will be more prone to local minima in the presence of redundant and unexplored state information.

In this work, we follow this same reasoning by condensing the VFA heuristics criteria detailed in Sec. II to give the policy a model-based prior on the footholds' safety. We perform regression via supervised learning by employing a denoising convolutional encoder-decoder style architecture, which we call as SaFE-Net (Safe Foothold Evaluation Network) to compute a fast and precise segmentation of the foothold heightmap $\boldsymbol{H}$. The encoder of SaFE-Net consists of a Convolutional Neural Network whose output is linearised

and appended with some state information required to infer the safety of the foothold, followed by a linear layer, whose output is the encoded representation $\boldsymbol{h}_{\text{safe}}$. The decoder consists of a linear layer, and its output is further reshaped and deconvolved to obtain a segmentation map $\hat{\boldsymbol{\mu}}_{\text{safe}}$ where each cell corresponds to each foothold's safety. Fig. 3 provides a visual description of the architecture of SaFE-Net mentioned above.

To train SaFE-Net, we collect our heightmap dataset in RaiSim [20] over different rough terrain scenarios (see Fig. 1). It consists of the foothold heightmap of the leg in swing $\boldsymbol{H}$, the subsequent safety evaluated using the heuristics $\boldsymbol{\mu}_{\text{safe}}$, the gait phase of the leg in swing ϕ, the linear and angular velocities ${}^b\boldsymbol{v}_{\text{com}}$ and ${}^b\boldsymbol{w}_{\text{com}}$ of the center of mass. We minimise a combination of the Binary Cross-Entropy [21] (BCE) and Generalized Dice [22] losses, such as

$$L_{\text{BCE}} = \frac{1}{N} \sum_{i=1}^{N} ((\hat{\mu}_{\text{i}} \log \sigma(\mu_{\text{i}}) + (1 - \hat{\mu}_{\text{i}}) \log(1 - \sigma(\mu_{\text{i}}))$$

$$L_{\text{dice}} = 1 - (2 \cdot \frac{\sum_{i=1}^{N}(\mu_i \hat{\mu}_i)}{\sum_{j=1}^{N} \mu_j + \sum_{d=1}^{N} \mu_d})$$

where $\hat{\mu}_i$ and μ_i are respectively the single element network's prediction and ground truth for the i-th element of the matrices $\hat{\boldsymbol{\mu}}_{\text{safe}}$ and $\boldsymbol{\mu}_{\text{safe}}$; N is the number of elements in the flattened matrix $\boldsymbol{H}$, and σ represent the sigmoid activation function. This combination of losses is commonly used in segmentation tasks to consider pixel-wise accuracy and global spatial coherence. The binary cross-entropy loss encourages accurate predictions at the pixel level, while the dice loss encourages spatially coherent segmentations by penalizing false negatives and false positives.

SaFE-Net provides two important components that form the basis of safety for training the subsequent footstep policy. The encoded representation of the heightmap, $\boldsymbol{h}_{\text{safe}}$, which inherently embeds safety and is deployed as an additional state to the RL agent, and the segmented safety map $\hat{\boldsymbol{\mu}}_{\text{safe}}$, which is used for action masking as described in the following section.

B. Footstep Planning Policy

The RL agent, acting as a centralized planner, determines the optimal footstep positions exclusively for the legs in swing, with the agent being queried only twice during this phase. These are during the lift-off of the foot and when it reaches the apex of the swing trajectory, which is generated by a separate model-based module. No more additional corrections are computed after the apex in order to avoid aggressive tracking maneuvers which can destabilize the robot's motion.

The following describes the policy's state space, action space, the adopted safety masking procedure, reward functions, and training strategy.

1) State: The policy of our foothold adaptation strategy has states that represent both proprioceptive (x_{prop}) and exteroceptive (x_{ext}) information, such as

$$x_{\text{prop}} = ({}^b v_{\text{com}}^{\text{usr}}, {}^b w_{\text{com}}^{\text{usr}}, {}^b v_{\text{com}}, {}^b w_{\text{com}}, \, q_{\text{hist}}, \, R, \, \phi^{\text{legs}}, \, a_{\text{prev}})$$
$$x_{\text{ext}} = h_{\text{safe}}^{\text{legs}}$$

where $q_{\text{hist}} \in \mathbb{R}^{24 \cdot t}$ is a sparse representation of t past joint position and velocity values, which helps to reconstruct possible disturbance during motion; $R \in \mathbb{R}^{3 \times 3}$ is a rotation matrix for retrieving the roll, pitch, and yaw of the robot; $\phi^{\text{legs}} \in \mathbb{R}^4$ is a vector of phase variables; finally, a_{prev} is the previous action from the policy and $h_{\text{safe}}^{\text{legs}} \in \mathbb{R}^{32 \cdot 4}$ is obtained by stacking the encoded representations from SaFE-Net for the heightmaps of each leg.

2) Actions: The output of the policy is defined over a multidimensional continuous action space, where each possible action a^i is responsible for a precise xy-foothold coordinate for the i-th leg in the swing. In our case, the robot is constrained to perform a trotting gait. Hence only two legs can lift simultaneously, and the policy output $a \in \mathbb{R}^4$. The gait phases vector ϕ helps the policy to understand which legs can be commanded at the current time instant.

3) Safety Masking: After querying the agent, the continuous actions are converted to exact discrete foothold choices in the heightmaps H^i of the i-th legs currently in swing. One can normally assign a negative reward signal for each unsafe foothold choice to encourage safe foothold candidates. This approach, however, does not provide any safety guarantee for the policy to sample safe actions, even after the training procedure. Furthermore, unsafe foothold selections can happen rarely during training, hardening the optimization problem [23]. To bypass the above limitations, we additionally employ an invalid action masking strategy to constrain the possible foothold's choice. Both during training and deployment, the output of SaFE-Net is employed to modify the action given by the policy if it is flagged as unsafe by choosing the *nearest* foothold marked safe in $\hat{\mu}_{\text{safe}}$.

4) Reward Functions: Given the safety criteria considered in this work, we decided to employ a small set of reward functions to describe the desired behavior of the robot. These terms are related to the tracking performance of the user-commanded linear velocities ${}^b v_{\text{usr}}$ and a regularization behavioral component, defined respectively by the following functions:

$$r_{\text{track}} = \exp\left(- \frac{({}^b v_{\text{com}}^{\text{usr}} - {}^b v_{\text{com}}))}{0.2 \cdot (1 + |{}^b v_{\text{com}}^{\text{usr}} - {}^b v_{\text{com}}|)} \right)^2$$
$$r_{\text{reg}} = \sum_{i=1}^{N_{\text{swing}}} ||H^i(a^i) - p_{\text{foot}}^i||$$

where, in r_{reg}, $H^i(\cdot)$ is the foot position in the i-th heightmap proposed by the policy a^i converted to the corresponding discrete values, and p_{foot}^i is the nominal foot position (1) at the center of the heightmap. Both these values refer to the i-th leg in swing. This reward term helps to stabilize the learning progress since we know that (1) is a good prior in the case of blind locomotion. Furthermore, we add some terminal sparse negative reward r_{terminal} in the case of self-collision or other catastrophic behavior, such as hitting the ground with the trunk or exceeding some pitch and roll thresholds, and a negative constant reward in the case a^i is converted to unsafe footholds.

5) Training Procedure: We trained our agent by applying external disturbances during random intervals to the CoM by sampling randomly in the range $[-35\text{N}, 35\text{N}]$, both in the longitudinal and lateral direction of the robot. We applied a curriculum strategy to increase the difficulty of the terrain during training. Furthermore, the terrain properties are randomized, and a new environment is sampled on every episode reset to encourage generalization. By employing PPO in the same simulation environment described in Sect. III-A, this training procedure is performed sequentially after a satisfactory precision of SaFE-Net is achieved.

C. Model Predictive Control

To actuate the optimized foothold position chosen by the policy, we employ a well-studied formulation used for the real-time motion planning and control problem known as a Model Predictive Control (MPC). The formulation we utilized implements a linearized reduced order model to approximate the robot dynamics, called the Single Rigid Body Model (SRBM), which has shown its effectiveness, especially in the case of quadruped robots [24] where the inertia of the legs can be usually neglected. Using this approximation, the system dynamics can be written as

$$\ddot{p}_{\text{com}} = \frac{1}{m} \sum_{i=1}^{n_{\text{leg}}} f_i + g$$
$$\dot{R} = R^b \hat{w} \quad R \in SO(3) \tag{2}$$
$$^b I^b \dot{w} = R^T (\sum_{i=1}^{n_{\text{leg}}} \hat{r}_i f_i) - {}^b \hat{w}^b I^b w$$

where $\ddot{p}_{\text{com}}$ is the center of mass acceleration f_i is the Ground Reaction Force (GRF) acting on the i-th leg, $\dot{R}$ is the derivative of rotation matrix between the world and the robot base, while w and $\dot{w}$ are the angular velocity and its derivative. r_i is the vector connecting the base and foot
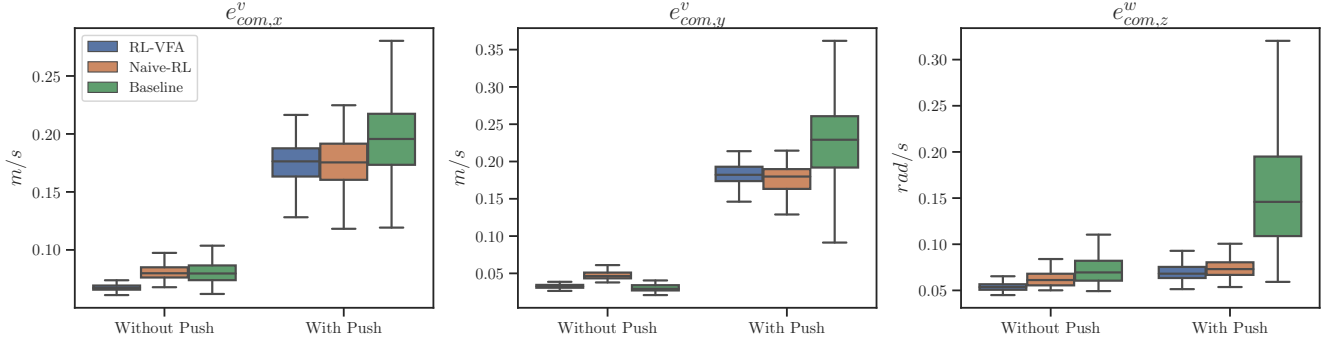
Fig. 4. Median and variance of the linear (x-y) and angular (yaw) velocities tracking errors with and without the presence of external disturbances. These results were obtained by commanding the robot with a forward velocity of 0.3 m/s.

$r_i = H^i(a^i) - p_{\text{com}}$, the operator $\widehat{(\cdot)}$ maps the vector to a screw-symmetric matrix. Finally, I and m are the inertia matrix and mass of the robot body, and g is the gravity vector. Eq. (2) is non-linear in the angular part. We use the variation-based linearization scheme presented in [25]. We express the rotational error in the $SO(3)$, considering the variation to the operating point to be free from singularities in the representation. We then perform a first-order Taylor expansion of the matrix exponential to then vectorize the error expressed in $SO(3)$ as $\boldsymbol{\xi} \in \mathbb{R}^3$ such that $\hat{\boldsymbol{\xi}} = \delta R$. The linearized dynamics is finally discretized using the forward Euler scheme. The system state is defined as $\boldsymbol{x} = \left[\boldsymbol{p}_{\text{com}}; \dot{\boldsymbol{p}}_{\text{com}}, \boldsymbol{\xi}, {}^b\boldsymbol{w} \right] \in \mathbb{R}^{12}$ and the control input as $\boldsymbol{u} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3, \boldsymbol{f}_4] \in \mathbb{R}^{12}$. Given the simplified model, system state and control input we can defined the optimal control problem (OCP) as:

$$
\begin{aligned}
\min_{\boldsymbol{x}, \boldsymbol{u}} \quad & \mathcal{L}_T(\boldsymbol{x}(N)) + \sum_{k=0}^{N-1} \mathcal{L}(\boldsymbol{x}_k, \boldsymbol{u}_k) \\
s.t. \quad & \boldsymbol{x}_{k+1} = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k + \boldsymbol{c}_k \\
& \boldsymbol{u}_k \in U_k \\
& k = 0, 1, ..., N-1 \\
& \boldsymbol{x}_{k=0} = \boldsymbol{x}_{\text{op}}
\end{aligned}
\tag{3}
$$

where $\mathcal{L}(\boldsymbol{x}(\cdot))$ is a convex quadratic cost over the user commanded velocities ${}^b\boldsymbol{v}_{\text{com}}^{\text{usr}}$, ${}^b\boldsymbol{w}_{\text{com}}^{\text{usr}}$ and body posture. $\boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{c}_k$ are the linearized dynamics, and U_k is the set of feasible ground reaction forces constrained by the outer pyramid approximation of the friction cone to guarantee non-slipping conditions. $\boldsymbol{x}_{\text{op}}$ is the state variable at the operating point. To solve the OCP problem we used a specialized quadratic programming solver [26], that exploits the sparse structure of the problem. Finally, the GRFs obtained by solving the optimization problem (3) are then converted into motor torques by applying

$$
\boldsymbol{\tau} = -\boldsymbol{J}^\top(\boldsymbol{q})\boldsymbol{u}
$$

where $\boldsymbol{q}$ is the vector containing the actual robot joints position, and $\boldsymbol{J}(\cdot)$ is the contact jacobian.

IV. RESULTS

The proposed approach has been validated through simulations on Aliengo [1], an electric quadruped robot developed by Unitree. In the following, we show simulation results by comparing three different approaches, which are

1) MPC-VFA (baseline): the optimal foothold is chosen as described at the end of Sect. II;
2) RL-VFA: where we employ the masking procedure explained in Sect. III-B, encoding the safety information both in the policy input and output space;
3) Naive-RL: where the safety information is only encoded in the reward function.

To minimize the considered safety violations in the naive RL approach (3), we added some additional reward terms. Specifically, we designed two negative sparse reward terms in the case of kinematic violations and shin collisions, and a distance-based reward term if the chosen foothold is placed near a terrain edge. For the last computation, we use the same TR heuristic explained in Sect. II. Furthermore, similar to the footstep policy in [17], we fed to the naive RL approach (3) an encoded representation of the heightmap (without providing any safety information) using a convolutional autoencoder architecture which was trained using the same heightmap data collected for SaFE-Net. The two policies mentioned above use the same training strategy as explained in Section III-B and an extensive hyperparameter search has been applied using Optuna [27] to optimize their results.

In the following, we want to highlight the benefit of employing a learned footstep planning policy within a model-based control framework, and the performance increment that can be achieved by explicitly considering safety information. For this, we tested the robot over irregular and randomized terrains and averaged the obtained results over 100 episodes (each of them consisting of 250 footstep placements).

In Fig. 4, we compare the two learning methods against MPC-VFA, showing the linear and angular velocity tracking error e_{com}^v, e_{com}^w, achieved with and without the application of additional external disturbances, while commanding the robot with a forward velocity of 0.3 m/s. As shown in the figures, RL-VFA is able to achieve better results (in median and variance) compared to the other methods, reaching a

mean reduction in the x and yaw tracking error of $\approx 20\%$ compared to the baseline. A similar result can be observed in the presence of external disturbances due to the ability of the policy to recover faster after a push. Similarly, the simpler naive implementation of RL is able to achieve good tracking performance compared to the baseline. However, given the additional reward terms and the intrinsic difficulty in achieving the global optimum, this method achieved worse performance compared to our approach.

This difficulty is clearly visible in the next result. To show that the proposed method can enable robust locomotion in the presence of severe disturbances, we analyzed the external forces at the CoM level withstood by all three methods (Fig. 5) applying the for a duration ranging from 1 to 3 seconds. The previous analysis can be applied here as well. Similar results are depicted in Fig. 6 where we plot the success rate (i.e. percentage of episodes that do not terminate with a body collision) achieved by all three methods during training. Here we superimpose the result coming from *three of the best* hyperparameters found during our extensive hyperparameter search. The outsourcing of the safety information enables both better sample efficiency and a more robust locomotion. The last can be observed in the accompanying video, where shin collisions, kinematic violations, and slippage over terrain edges clearly require additional effort from the locomotion controller.

Lastly, we compare in Table I how the defined safety conditions hold in our approach and in the naive RL policy case. For this, we test both agents at the beginning of their
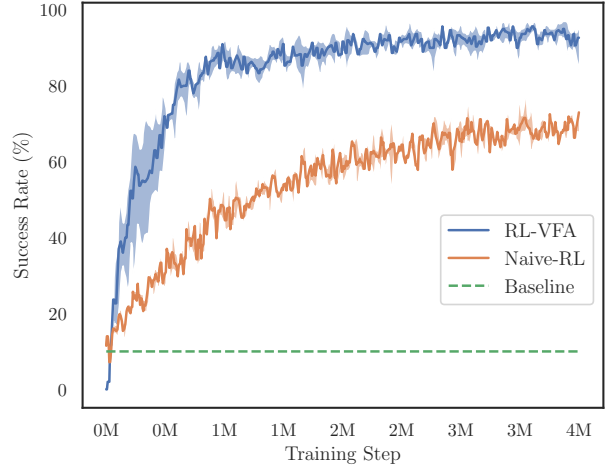
Fig. 6. Comparison of the episode success rate of Naive-RL and RL-VFA using three different hyperparameters during training on irregular terrains with fixed disturbances (+/-35N in the x-y directions) applied at random intervals. To highlight the benefit of a learned footstep policy, we plot the mean results obtained by the baseline (MPC-VFA).

training (10000 steps) and at their last episode (4000000 steps), and we calculate the percentage of safety violations over 100 episodes. Our approach consistently outperforms Naive-RL, especially in the case of terrain violations, where footholds are placed near a terrain edge ≈ 100 times more often compared to our approach. The same situation emerges in the case of shin collisions, where we achieved a reduction rate of $\approx 10x$. Still, some violations remain even in our case given the inevitable classification error of SaFE-Net. These results can be explained by analyzing the number of violations of RL during the policy evaluation at the beginning of training: shin collision and kinematic violations are represented by sparse events, over which optimization is inevitably complicated. Given the sparse nature of these criteria, simply penalizing such actions that cause violations as in [17] is not necessarily sufficient to learn to avoid them. Furthermore, we want to highlight how there is no significant difference in the number of safety violations between the training episodes in our approach, since, given our masking procedure, their enforcement is not hinged on any particular learning curve.

The reader can refer to the accompanying website for the videos of the above results. Furthermore, in the video, we show another simulation on Gazebo [28] and experiments on the real robot to prove that our method does not suffer from any particular domain adaption problem. For this, we trained our policy by randomizing the robot's mass and inertia, and we added noise both to the MPC and to the policy states. Even in this case, we obtain similar comparative results to the ones described in this section.

V. Conclusions

In this paper, we have proposed a method for imposing safety constraints to a learned footstep planning policy via
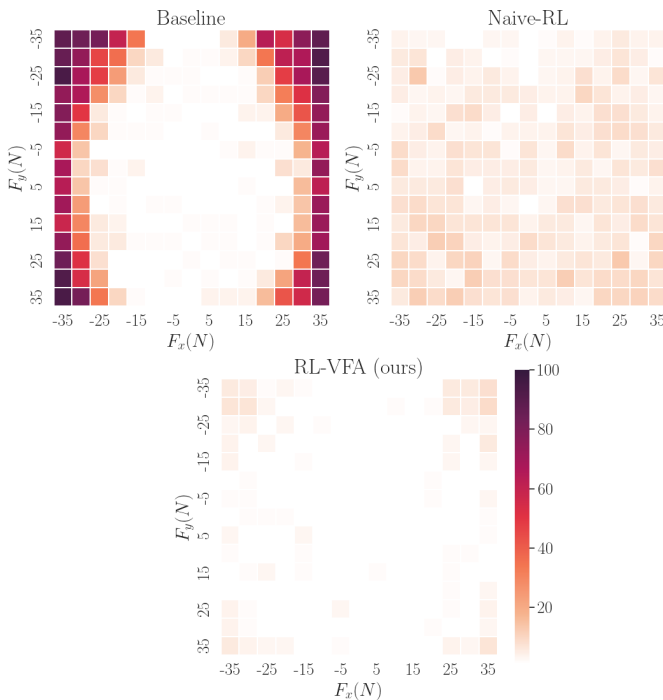

Fig. 5. Disturbance rejection analysis over external forces in the range $[-35N, 35N]$. Lighter squares depict higher robustness (zero episode termination in white).

external model-based priors. The method hinges upon our visual foothold adaption technique (VFA) which classifies unsafe footstep locations by analyzing their kinematic feasibility, shin collision, and vicinity to terrain edges, information that is then used by the policy to plan only safe footholds.

The proposed approach was statistically validated in simulation and tested on a real quadrupedal robot. In particular, numerical simulations of our method show that a low number of violations of the imposed safety conditions happen during training, resulting in an inherently safer footstep planning policy. Furthermore, we show how this approach is able to attain better final performances compared to standard RL approaches, thanks to a reduced number of reward functions needed to shape the behavior of the policy.

An interesting feature of the presented work is that additional constraints can be easily incorporated into the pipeline at demand just by modifying the training data for the encoder-decoder network. Given this feature, future work will consider the problem of coupling the proposed approach with dynamic constraints during the entire learning transient, for example, by explicitly taking into consideration criteria based on the Zero Moment Point [29] or the Instantaneous Capture Point [30], in order to increase the final robot's performance. Furthermore, we plan to test the proposed approach on the real hardware.

TABLE I

PERCENTAGE OF SAFETY VIOLATIONS OVER 100 EPISODES

Policy	Terrain (TR)	Leg (LC)	Kinematic (KF)
RL-VFA (@10K steps)	0.9	0.7	0.8
RL-VFA (@4M steps)	**0.6**	**0.3**	**0.6**
Naive-RL (@10K steps)	48.4	5.8	1.77
Naive-RL (@4M steps)	36.5	4.5	1.5

REFERENCES

[1] Unitree Robotics, "Aliengo," *https://www.unitree.com/aliengo*.
[2] J. R. Rebula, P. D. Neuhaus, B. V. Bonnlander, M. J. Johnson, and J. E. Pratt, "A controller for the littledog quadruped walking on rough terrain," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 1467–1473.
[3] 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*, 2008, pp. 811–818.
[4] 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*, pp. 1–1, 2019.
[5] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.
[6] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "Mpc-based controller with terrain insight for dynamic legged locomotion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2436–2442.
[7] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model-predictive control," *IEEE Transactions on Robotics*, pp. 1–20, 2023.
[8] S. Omar, L. Amatucci, G. Turrisi, V. Barasuol, and C. Semini, "Fast convex visual foothold adaptation for quadrupedal locomotion," in *4th Italian Conference in Robotics and Intelligent Machines (I-RIM)*, 2022, pp. 143–146.
[9] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8352–8358.
[10] N. Rathod, A. Bratta, M. Focchi, M. Zanon, O. Villarreal, C. Semini, and A. Bemporad, "Model predictive control with environment adaptation for legged locomotion," *IEEE Access*, vol. 9, pp. 145 710–145 727, 2021.
[11] F. Risbourg, T. Corbères, P.-A. Léziart, T. Flayols, N. Mansard, and S. Tonneau, "Real-time footstep planning and control of the solo quadruped robot in 3d environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 12 950–12 956.
[12] X. B. Peng, G. Berseth, K. Yin, and M. van de Panne, "Deeploco: dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transaction on Graphics*, vol. 36, pp. 1–13, 2017.
[13] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.
[14] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, 2022.
[15] S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. H. Lee, and J. Hwangbo, "Learning quadrupedal locomotion on deformable terrain," *Science Robotics*, vol. 8, no. 74, p. 2256, 2023.
[16] A. Pandala, R. T. Fawcett, U. Rosolia, A. D. Ames, and K. A. Hamed, "Robust predictive control for quadrupedal locomotion: Learning to close the gap between reduced- and full-order models," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6622–6629, 2022.
[17] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2908–2927, 2022.
[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.
[19] M. H. Raibert, *Legged Robots That Balance*. MIT Press, 1986.
[20] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.
[21] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[22] C. H. Sudre, W. Li, T. K. M. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," *Deep learning in medical image analysis and multimodal learning for clinical decision support : Third International Workshop*, vol. 2017, pp. 240–248, 2017.
[23] D. Rengarajan, G. Vaidya, A. Sarvesh, D. Kalathil, and S. Shakkottai, "Reinforcement learning with sparse rewards using guidance from offline demonstration," in *International Conference on Learning Representations*, 2022.
[24] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.
[25] Y. Ding, A. Pandala, C. Li, Y.-H. Shin, and H.-W. Park, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1154–1171, 2021.
[26] A. G. Pandala, Y. Ding, and H.-W. Park, "qpswift: A real-time sparse quadratic program solver for robotic applications," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3355–3362, 2019.
[27] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.
[28] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2149–2154 vol.3.
[29] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, "Introduction to humanoid robotics," in *Springer Tracts in Advanced Robotics*, 2014.
[30] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*, 2006, pp. 200–207.