Quadrupedal Footstep Planning using Learned Motion Models of a Black-Box Controller

Ilyass Taouil[1,2], Giulio Turrisi[1], Daniel Schleich[2], Victor Barasuol[1], Claudio Semini[1], and Sven Behnke[2]

***Abstract*— Legged robots are increasingly entering new domains and applications, including search and rescue, inspection, and logistics. However, for such a systems to be valuable in real-world scenarios, they must be able to autonomously and robustly navigate irregular terrains. In many cases, robots that are sold on the market do not provide such abilities, being able to perform only blind locomotion. Furthermore, their controller cannot be easily modified by the end-user, requiring a new and time-consuming control synthesis. In this work, we present a fast local motion planning pipeline that extends the capabilities of a black-box walking controller that is only able to track high-level reference velocities. More precisely, we learn a set of motion models for such a controller that maps high-level velocity commands to Center of Mass (CoM) and footstep motions. We then integrate these models with a variant of the A^* algorithm to plan the CoM trajectory, footstep sequences, and corresponding high-level velocity commands based on visual information, allowing the quadruped to safely traverse irregular terrains at demand.**

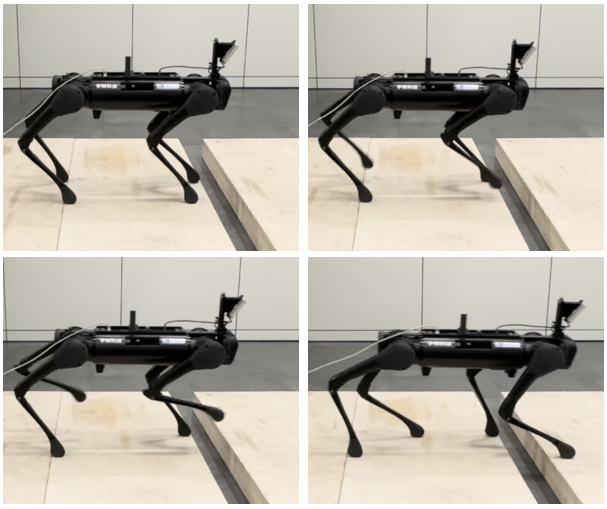

Fig. 1. The Aliengo quadruped robot stepping over a gap with the proposed approach.

I. Introduction

Autonomous systems are continuously entering new domains and applications. Significant leaps have been made in making these systems operational, demonstrating enormous potential for the future. Legged systems in particular offer high mobility and versatility given their ability to step at discontinuous locations. On the other hand, to be able to be deployed in the real world they must be able to reason about the surroundings in order to select the best contact locations. The work we present addresses the development of perceptive locomotion skills for legged robots whose manufacturer's controller can only track high-level velocity commands using proprioceptive data, and cannot receive target footstep locations. Examples of such a systems are Aliengo [1], Vision 60 [2], and Spot [3]. We seek to extend the capabilities of such controllers by planning the appropriate velocity commands that allow the legged system to overcome different types of uneven terrains (Fig. 1). We do so by using exteroceptive sensory data and a set of motion models learned from the controller's behavior, of which we do not know the implementation details, hence treating it as a black box.

In the literature, there exists several paradigms to tackle vision-based locomotion. In [4]–[10] the authors use a model-based approach to plan the required footstep sequences to overcome irregular terrains. Learning-based methods [11]–[13] have also shown great ability to couple control and perception in these scenarios. However, the aforementioned approaches, respectively require the ability to either be able to set the desired footstep location at the controller level, or to have available in simulation the behavior of the closed-loop system to learn the policy, something that in both cases is not available on the previously mentioned commercial platforms.

Works similar to ours are the one presented in [14]–[16]. In [14], the authors presented a footstep planner for the humanoid robot ASIMO [17] whose controller can only receive desired body displacements. The authors use the A^* [18] search algorithm to compute the optimal sequence of footstep locations up to a pre-defined planning horizon, using a mapping in the form of a lookup table where footstep displacements are defined based on the current state, action, and environment. The method, however, is limited to slow and static gaits given the high re-planning time. Furthermore, it can only cope with flat terrains. In [15], the authors presented a method that generates footstep trajectories for a humanoid to approach and kick a ball. Similar to the previous method, the authors only consider flat terrains and their A^* based planner requires several minutes to find an optimal solution, hence requiring a subsequent approximation of the final solution by policy learning. Finally, in [16], the authors presented an approach that plans a sequence of footsteps for the Atlas humanoid, which is commanded by a black-box controller. However, their method allows changing the footstep locations at the controller level, which in our study case is not possible.

[1]Dynamic Legged Systems Laboratory, Istituto Italiano di Tecnologia (IIT), 16163 Genova, Italy

[2]Autonomous Intelligent Systems, University of Bonn, 53115 Bonn, Germany

To summarize, the main contributions of this paper are the following:

- a novel motion planning pipeline for quadrupedal locomotion that extends blind locomotion capabilities of black-box controllers with perception-based skills
- a set of learned motion models that predict the behavior of a black-box quadrupedal locomotion controller
- extensive experimental tests with the Aliengo robot over different test scenarios that demonstrate the capability of the approach

A. Outline

The paper is organized as follows: Section II describes the problem formulation, whereas Section III delves into the details of the proposed method and describes the learned motion models and the footstep planning formulation. Section IV introduces the metrics used for the evaluation and presents the results obtained from the extensive experimentation in different indoor scenarios. Finally, Section V summarizes the presented method and offers some considerations on its limitations and possible future extensions.

II. Problem Formulation

In this work, we address the development of perceptive locomotion skills for purely reactive controllers. Such controllers solely rely on proprioceptive information in order to keep balance, thereby making them prone to failure in complex scenarios where vision is essential. The Aliengo robot, which will be under study in this work, is an example of such a system, as its built-in controller can only perform blind locomotion and can only be commanded with high-level velocity references.

Given the limited control authority that the user can have on such a system, the proposed solution is formulated as a motion planning problem that aims at finding the optimal footstep sequences by varying velocity commands. In fact, CoM velocities and footstep positions are quantities often correlated [19]–[21], meaning that this mapping, if learned, can be employed to allow Aliengo to traverse irregular terrain at demand.

The API of Aliengo provides multiple information about the robot states, such as the robot CoM position and velocity, defined as $\boldsymbol{p}_{\text{com}}$ and ${}^{b}\boldsymbol{v}_{\text{com}}$ and respectively defined in the world frame W and in the base frame B, the relative position of the feet $\bar{\boldsymbol{p}}_{\text{foot}}$ with respect to the CoM, the robot contact forces $\boldsymbol{f}$, and the rotation matrix that maps information from the robot base to the world frame ${}^{w}\boldsymbol{R}_{b}$. All these quantities, as will be explained in Section III, will be exploited to learn accurate motion models of the locomotion controller of Aliengo. We also have access to depth information from a top mounted RealSense D435f camera to reconstruct the environment.

Furthermore, the blind controller under study only allows a trotting gait, where two diagonal feet swing at a time. This will inevitably hinder the possible scenarios that can be traversed, constraining the two swing legs to have the same foot displacement at the same time. This limitation can only be alleviated by a planning procedure with a sufficient horizon length.

III. Proposed Approach

The motion planning pipeline consists of three blocks: 1) a motion models interface that uses the learned models to predict the robot's next CoM position, CoM velocity, and swing leg touchdown; 2) an elevation mapping module [22] that reconstructs the local terrain as a $2.5D$ grid map, which is then used to compute a foot costmap describing safe and unsafe stepping locations; 3) a motion planning module that implements a variant of the A^* algorithm and integrates the output of the previous two modules to plan footstep sequences and corresponding high-level velocities, at every detected full-stance phase, which are then fed to the robot controller.

At a high level, the above motion planning pipeline works as follows: the algorithm, given a user-defined CoM goal position $\boldsymbol{p}_{\text{com}}^{\text{goal}}$, acquires the actual robot state at the current time t, defined as

$$\boldsymbol{s}^{t} = (\boldsymbol{p}_{\text{com}}^{t}, {}^{b}\boldsymbol{v}_{\text{com}}^{t}, {}^{b}\boldsymbol{v}_{\text{com}}^{t-1}, {}^{b}\boldsymbol{v}_{\text{ref}}^{t}, {}^{b}\boldsymbol{v}_{\text{ref}}^{t-1}, {}^{b}\bar{\boldsymbol{p}}_{\text{foot}}^{t}) \tag{1}$$

where ${}^{b}\boldsymbol{v}_{\text{ref}}^{t}$, ${}^{b}\boldsymbol{v}_{\text{ref}}^{t-1}$ are the last two commanded velocities, and ${}^{b}\boldsymbol{v}_{\text{com}}^{t-1}$ is the previous CoM velocity. The planner then iterates over a set of discrete velocity references and starts building a graph describing the evolution of the robot states until $\boldsymbol{p}_{\text{com}}^{\text{goal}}$ is reached or N sequential safe footholds are found. For this, three different regressor models are used to predict the components of $\boldsymbol{s}$ during each node expansion: a CoM Displacement Model (CDM), a Footstep Displacement Model (FDM), and a CoM Velocity Model (CVM). Finally, once the solution is obtained, only the first velocity command is applied to the robot and then the procedure is restarted in a receding horizon fashion.

A block diagram of the proposed approach can be seen in Fig. 2.

In the following sections, we describe the learning procedure to obtain the models (Section III-A), detailing the necessity of the extended state representation adopted in (1), and the proposed planning algorithm (Section III-B).

A. Motion Models Learning

The motion models learned in this work map an input vector describing a reduced representation $\boldsymbol{x}^{t}$ of the robot state in (1), to feet and CoM displacement as well as the CoM velocity at time $t+1$, fundamentally creating three different regression problems. Different approaches could be used to learn these models, including a simple *mapping table* or a *Neural Network*. However, given that in our experiments we observed a strong linear character exhibited by the controller (see Sect. IV), a better approach is found in the *Multivariate Linear Regression* (MLR) [23], where a linear combination in the form of a first-degree polynomial that fits the data is computed.

The most fundamental variable we want to predict in our planning pipeline is the next footstep placement $\boldsymbol{p}_{\text{foot}}^{t+1}$. In our application, the FDM regressor outputs a displacement

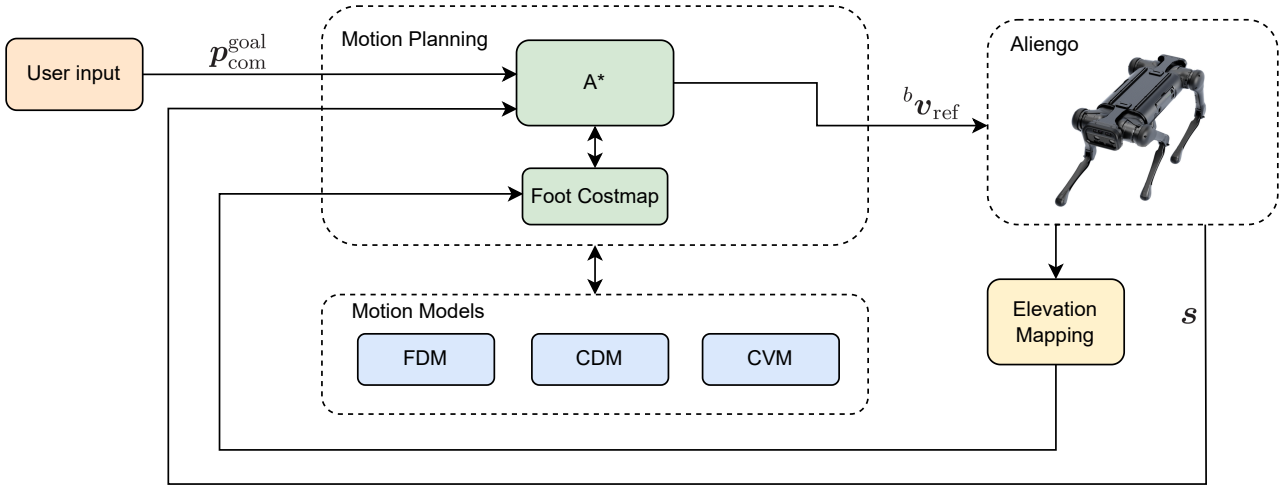

Fig. 2. Block diagram of the overall planning procedure. Every time a full-stance phase is detected, a new solution is computed in a receding horizon fashion. For clarity, we dropped the dependence on the time t. The motion models are: Footstep Displacement Model (FDM), CoM Displacement Model (CDM), and CoM Velocity Model (CVM).

${}^b\boldsymbol{\delta p}_{\text{foot}}^t$ over the actual foot position expressed in the base frame. Usually, this variable is strictly correlated with the commanded and actual CoM velocity of the robot, but other variables can also play an important role in its derivation. For this reason, we experimented with different input spaces for this model, and the best accuracy was obtained by choosing it as

$$\boldsymbol{x}_{\text{FDM}}^t = ({}^b\boldsymbol{v}_{\text{com}}^t, {}^b\boldsymbol{v}_{\text{com}}^{t-1}, {}^b\boldsymbol{v}_{\text{ref}}^t, {}^b\boldsymbol{v}_{\text{ref}}^{t-1}, {}^b\bar{\boldsymbol{p}}_{\text{foot}}^t). \tag{2}$$

As will be explained in Sect. IV, this input space choice is by no means dependent on the type of the regressors, since we observed that employing both linear and nonlinear basis functions does not impact the overall final accuracy.

The input space $\boldsymbol{x}_{\text{FDM}}^t$ is characterized by the presence of the commanded and actual CoM velocities at time t and time $t-1$ as well. We found that this state history information is needed in order to mimic the non-ideal behavior of the controller of Aliengo, which is unable to track the commanded velocity in just one foot swing (Fig. 3).

In our approach, we plan with a look-ahead horizon of N footsteps in order to reduce myopic behaviors in the final solution. Thus, we need to predict the future state variables ${}^b\boldsymbol{v}_{\text{com}}^{t+i}$, ${}^b\bar{\boldsymbol{p}}_{\text{foot}}^{t+i}$, with $i = 1, ..., N$, for the FDM state representation (2) as well. For this, the CVM regressor provides directly the future CoM velocity ${}^b\boldsymbol{v}_{\text{com}}^{t+i}$, while the CDM regressor predicts the displacement ${}^b\boldsymbol{\delta p}_{\text{com}}^t$ with respect to the actual CoM position. This value, together with the previous output of the FDM regressor, is then used to retrieve the last component of $\boldsymbol{x}_{\text{FDM}}^{t+1}$ as the following: first, we compute the next CoM and foot position in the world frame as

$$\boldsymbol{p}_{\text{foot}}^{t+1} = {}^w\boldsymbol{p}_{\text{foot}}^t + {}^w\boldsymbol{R}_b\,{}^b\boldsymbol{\delta p}_{\text{foot}}^t \tag{3}$$

$$\boldsymbol{p}_{\text{com}}^{t+1} = \boldsymbol{p}_{\text{com}}^t + {}^w\boldsymbol{R}_b\,{}^b\boldsymbol{\delta p}_{\text{com}}^t \tag{4}$$

and then we evaluate their relative displacement as

$${}^b\bar{\boldsymbol{p}}_{\text{foot}}^{t+1} = {}^b\boldsymbol{R}_w(\boldsymbol{p}_{\text{foot}}^{t+1} - \boldsymbol{p}_{\text{com}}^{t+1}) \tag{5}$$

Both CVM and CDM regressors share the same input space with the FDM.

For training the above models, we first performed a data acquisition step gathering the quadruped's high-level state information offered by the API, while it executes various motion commands. The data collection was performed on a flat ground with a step height of 10cm. To acquire the data we use Aliengo's controller interface that receives as input a 3-dimensional velocity command. Based on this interface, we defined a strategy to gather the data such that they reflect all possible behaviors of interest to the planning process. This entails exhausting all possible input permutations that can be commanded to the robot in terms of velocity magnitudes. More precisely, we send several reference commands over varying periods of time to learn the controller's behavior during acceleration, deceleration, and continuous motions. Finally, we fit the corresponding MLR models by applying a simple least square loss.

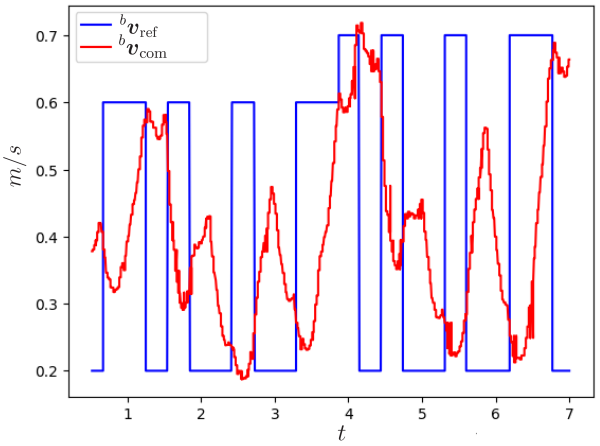

Fig. 3. The CoM velocity reference tracking behavior in x of the Aliengo's built-in controller. In this work, the controller is set in the "Sport Mode 3.0".

B. Footstep Planning with Learned Models

The motion planning process consists of finding the best sequence of high-level velocity commands that can be fed to the quadruped's internal controller in order to overcome complex terrains. To achieve this task, we utilize a variant of the graph-based A^* algorithm [24] that is able to better cope with the continuous state-space nature of the footstep planning problem in hand. It should be noticed that other planning methods could have been employed as well, such as randomized algorithms [25], but graph-based search algorithms usually show a superior converge rate to a solution for tasks with small-dimensional state spaces.

Starting from a generic full-stance phase defined by the state $\boldsymbol{s}^{\text{start}}$, where all the feet of the robot are in contact with the ground, the planner searches for the best N future footsteps needed to reach a user-defined goal position $\boldsymbol{p}_{\text{com}}^{\text{goal}}$. For this, first we expand a set of discretized velocities to generate M new nodes $\boldsymbol{s}'$, with M the possible velocity command values that we can apply to our robot. In our application, we discretize the action space with a step-size of 0.1 m/s, and we set the minimum and maximum allowed command velocity, respectively at 0.0 m/s and 1 m/s. Afterwards, we compute the state information associated with all the expanded nodes. This step is needed to check if those are associated with states that were visited before and if the associated footholds are safe.

For this, we first compute the CoM position $\boldsymbol{p}'_{\text{com}}$ of every single node in the world frame by applying the CDM regressor and Eq. (4). We then discretize $\boldsymbol{p}'_{\text{com}}$ with a 0.01 m step-size. To check if the new nodes are redundant, we maintain a list of visited nodes V. The expanded nodes are directly discarded if there is a node with the same velocity command and discretized CoM position value, otherwise a safety check is performed on the proposed foothold locations. For this, we compute the associated footstep displacements ${}^b\boldsymbol{\delta p}_{\text{foot}}$ via the FDM regressor and compute the respective world coordinate $\boldsymbol{p}'_{\text{foot}}$ by Eq. (3).

The safety of the footholds is then retrieved by analyzing the value of $\boldsymbol{p}'_{\text{foot}}$ in the foot costmap, where pixels that exhibit a large height variation compared to their neighborhood are flagged as unsafe. We then calculate the associated traversability costs for each new node to choose the next candidate to visit. This comprises a Euclidean distance cost d_{goal} of the robot CoM position with respect to the goal, and a feet configuration cost d_{foot} needed to maximize the feet distances to the closest unsafe cell, such as

$$d_{\text{foot}} = \begin{cases} d_{\max} - d_{\text{foot}}, & \text{if } d_{\text{foot}} < d_{\max} \\ 0, & \text{otherwise} \end{cases}$$

where $d_{\max}$ is a user-defined safety margin threshold. In fact, solely relying on the Euclidean cost and the footstep validity check might cause the robot to step too close to the edges due to prediction errors or motion inaccuracies. However, by maximizing the feet distances, we can mitigate such errors. We add the nodes in the open list O for further expansion if there are no nodes with the same CoM discretized positions and with lower cost, otherwise we maintain in O only the best candidates. Finally, we retrieve the last components of the nodes state ${}^b\boldsymbol{v}'_{\text{com}}$, ${}^b\bar{\boldsymbol{p}}'_{\text{foot}}$ by respectively querying the CVM regressor and computing Eq. (5).

We can then proceed to extract from O the next node to visit by choosing the one with the lowest cost.

The above planning procedure continues until the goal is reached, N safe footsteps are found, or the maximum computational time is reached. In the last two cases, only the first planned velocity is commanded. The planning procedure is then repeated in a receding horizon fashion as soon as another full-stance phase is detected.

The pseudo-code of the proposed footstep planning algorithm can be found in Algorithm 1.

Algorithm 1 Footstep planner at the generic time t

procedure PLAN($\boldsymbol{p}^t_{\text{com}}$, ${}^b\boldsymbol{v}^t_{\text{com}}$, ${}^b\boldsymbol{v}^{t-1}_{\text{com}}$, ${}^b\boldsymbol{v}^t_{\text{ref}}$, ${}^b\boldsymbol{v}^{t-1}_{\text{ref}}$, ${}^b\bar{\boldsymbol{p}}^t_{\text{foot}}$)
  $i \leftarrow 0$
  $s^{start} \leftarrow (\boldsymbol{p}^t_{\text{com}}, {}^b\boldsymbol{v}^t_{\text{com}}, {}^b\boldsymbol{v}^{t-1}_{\text{com}}, {}^b\boldsymbol{v}^t_{\text{ref}}, {}^b\boldsymbol{v}^{t-1}_{\text{ref}}, {}^b\bar{\boldsymbol{p}}^t_{\text{foot}}, i)$
  $O \leftarrow \{s^{start}\}$
  $V \leftarrow \{\}$
  $path \leftarrow \{\}$
  $s \leftarrow$ **visit least-cost state representation from O**
  if $n == s^{goal}$ or $i == N$ or $O == \emptyset$ **then**
    return $path \leftarrow$ **best sequence of velocities**
  end if
  for each velocity **do**
    compute Footstep position by applying Eq. (3)
    if invalid feet locations **then**
      goto 11.
    end if
    compute CoM position by applying Eq. (4)
    compute CoM velocity ${}^b\boldsymbol{v}'_{\text{com}}$
    update number of node parents $i' = i + 1$
    construct state transition representation s'
    compute total cost $d_{goal} + d_{foot}$ for s'
    if s' is a duplicate in V **then**
      discard s'
    else if s' is a duplicate in O **then**
      if s' has lower cost **then**
        $O \leftarrow O \cup s'$
      end if
    else
      $O \leftarrow O \cup s'$
    end if
  end for
  $V \leftarrow V \cup s$
  goto 7.
end procedure

IV. Results

In this section we present the results of the prediction models and of the motion planning pipeline explained in Section III. All the experiments are executed on a system equipped with a top-mounted Intel RealSense D435f camera (Fig. 1) and an external motion capture system to obtain

Fig. 4. The proposed test scenarios (top) with the related foot costmaps (bottom). Starting from the left, we show Scenario I, Scenario II, and Scenario III. Furthermore, in the foot costmaps the areas which are not considered safe are shown in red and the traversable areas are shown in black.

odometry information and facilitate the computation of the elevation map from which we then compute the related foot costmap.

First, we present an empirical evaluation of the prediction models and a comparison between the different design choices of the regressors' input space. Afterward, we evaluate the proposed motion planning pipeline using the Aliengo system on three different indoor scenarios, where we neglect rotational movements.

A. Motion Models

We use the Root Mean Squared Error (RMSE) as a metric for the models' evaluation. In particular, for each model we compare the obtained RMSE for different input choices on the test set. Moreover, we also compare the best obtained linear model with a non-linear one.

For each model, we use as a baseline the input definition given in Section III, and compare it to three other formulations, which are:

1) without considering ${}^b\boldsymbol{v}_{\text{ref}}^{t-1}$, ${}^b\boldsymbol{v}_{\text{com}}^{t-1}$
2) without considering $\boldsymbol{p}_{\text{foot}}^{t}$
3) 2nd-order polynomial of the state (2)

where 1) and 2) respectively remove ${}^b\boldsymbol{v}_{\text{ref}}^{t-1}$, ${}^b\boldsymbol{v}_{\text{com}}^{t-1}$ and $\bar{\boldsymbol{p}}_{\text{foot}}^{t}$ from the input definition given in Section III, and 3) uses both a linear and a quadratic expression of the input space. Table I shows respectively the comparison of the RMSE for the aforementioned formulations.

TABLE I

REGRESSOR MODELS COMPARISON.

Regression Models	RMSE (m)
CDM w/o ${}^b\boldsymbol{v}_{\text{ref}}^{t-1}$, ${}^b\boldsymbol{v}_{\text{com}}^{t-1}$	0.0193
CDM w/o $\boldsymbol{p}_{\text{foot}}^{t}$	0.0115
CDM	0.0110
2nd order CDM	0.0116
FDM w/o ${}^b\boldsymbol{v}_{\text{ref}}^{t-1}$, ${}^b\boldsymbol{v}_{\text{com}}^{t-1}$	0.0406
FDM w/o $\boldsymbol{p}_{\text{foot}}^{t}$	0.0252
FDM	0.0199
2nd order FDM	0.0249
CVM w/o ${}^b\boldsymbol{v}_{\text{ref}}^{t-1}$, ${}^b\boldsymbol{v}_{\text{com}}^{t-1}$	0.118
CVM w/o $\boldsymbol{p}_{\text{foot}}^{t}$	0.0540
CVM	0.0492
2nd order CVM	0.0610

We note how the trend is similar for all three models. More precisely, we observe that the worst performing linear model is the one that does not consider ${}^b\boldsymbol{v}_{\text{ref}}^{t-1}$ and ${}^b\boldsymbol{v}_{\text{com}}^{t-1}$. This can be explained by the fact that Aliengo's controller does not track aggressively the reference velocity given at time t and its behavior is influenced by the velocity history at time $t-1$. Similarly, we also note how not considering $\bar{\boldsymbol{p}}_{\text{foot}}^{t}$ decreases the models' accuracy, although not as significantly for the CDM as for the rest of the models. This can be explained by the fact that the controller's behavior and consequently the CoM displacement, feet displacement, and velocity change depends on the postural configuration brought by the feet.

Finally, we also compare the best linear model obtained with a second-order polynomial to see if a non-linear formulation could improve the accuracy of the regressor. For all three models we obtain the same result, where the linear model has a higher accuracy than the non-linear one. This emphasizes the fact that there is a strong linear relationship between the input variables and output variables. Hence, even trying highly non-linear models would not bring any useful benefit to the models.

B. Motion Planning Pipeline

The motion planning pipeline is evaluated on three different real environments with varying degrees of complexity. Their morphology along with their respective foot costmap can be observed in Fig. 4.

The first scenario (Scenario I) consists of two symmetrical gaps across the longitudinal axis of the robot, respectively of 10 cm and 15 cm, while the second scenario (Scenario II) contains two asymmetrical gaps of 15 cm and a small step of 6 cm. These scenarios are proposed in order to highlight the particular capability of the built-in controller. As explained in Sect. II, the robot can only perform a trotting gait, and thus the swing feet are constrained to achieve the same foot swing distance during normal operation hindering the capability of the robot to traverse asymmetric scenarios. Finally, the third and last experiment (Scenario III) comprises one symmetrical gap of 10 cm, two asymmetrical gaps of 12 cm and 13 cm, and two small steps of 6 cm.

The obtained results, which can be observed in Table II,

TABLE II

EXPERIMENTAL RESULTS OF THE PROPOSED APPROACH ON THREE DIFFERENT INDOOR SCENARIOS.

	Scenario I			**Scenario II**			**Scenario III**		
Horizon	**3**	**5**	**7**	**3**	**5**	**7**	**3**	**5**	**7**
Success (%)	100%	100%	100%	100%	100%	100%	100%	100%	100%
Planning Time (ms)	0.239	0.924	1.75	0.475	0.765	0.975	0.234	0.825	1.20
Cumulative Cost	32.6	23.96	20.95	25.2	20.5	18.1	32.3	26.3	24.3
CDM MAE (m)	0.00786			0.00360			0.00339		
FDM MAE (m)	0.0198			0.0203			0.0170		
CVM MAE (m/s)	0.0317			0.0369			0.0445		

are collected during three different attempts for each scenario. The robot is able to successfully complete all the proposed scenarios, and as expected, varying the length of the planning horizon has a direct impact of the optimality of the final solution. For this, we report in Table II the cumulative cost obtained considering only the first step performed by the robot at the end of each re-planning procedure. Furthermore, we also observe how the planner takes on average less than 2ms for the re-planning phase even with longer horizons, making it suitable for reactive motions. Finally, we analyze the Mean Absolute Errors (MAE) of the three learned models on the three proposed scenarios. Even if their values are small, the data demonstrates that re-planning is necessary in order to avoid the inevitable error propagation.

Videos of the experiments and comparisons with the blind built-in controller are included in the accompanying video [1]. As expected, without visual information, the robot is more statistically prone to failure in such scenarios. Furthermore, we include in the accompanying video an additional experiment in a stair-like scenario to show the potentiality of the proposed method.

V. CONCLUSIONS

In this paper, we proposed a vision-based planning method able to enhance the built-in blind controller of Aliengo. To this end, the black-box nature of the controller is learned by means of three different regressors, fully integrated into the motion planning pipeline. Finally, we conducted experiments on scenarios with different complexity to highlight the capability of the approach.

The limitations of the proposed approach lie on the limited authority that the user can have on the black-box controller. In fact, to traverse some scenarios, it could be needed not only to modify the footstep location but even the step height (e.g. for high stairs) or the trunk pose [26] in advance. Both of these possibilities, in the case of Aliengo, are limited or precluded for the control modality used in this approach.

Future works will focus on reducing the conservativeness of the method. The safety margin (Sect. III) introduced for coping with prediction errors can be modified during planning, for example, by applying Bayesian regression techniques [27] and actively considering the additional information on the regressor uncertainty. Furthermore, we plan to enhance the ease of use of the proposed approach. The goal position could be in fact be computed directly by joystick commands, giving the possibility to the user to control the robot in the easiest and most reactive way possible.

[1] https://tinyurl.com/4rwsthr7

REFERENCES

[1] Unitree Robotics, "Aliengo," *https://www.unitree.com/aliengo*.

[2] Ghost Robotics, "Vision 60," *https://www.ghostrobotics.io*.

[3] Boston Dynamics, "Spot," *https://www.bostondynamics.com*.

[4] D. Belter, P. Łabecki, and P. Skrzypczyński, "Adaptive motion planning for autonomous rough terrain traversal with a walking robot," *Journal of Field Robotics*, vol. 33, no. 3, pp. 337–370, 2016.

[5] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping, and whole-body control," *IEEE Transactions on Robotics*, vol. 36, no. 6, pp. 1635–1648, 2020.

[6] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *Proc. 2018 IEEE International Conference on Robotics and Automation*, 2018, pp. 5761–5768.

[7] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt, "Footstep planning for autonomous walking over rough terrain," in *Proc. 2019 IEEE-RAS International Conference on Humanoid Robots*, 2019, pp. 9–16.

[8] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "Mpc-based controller with terrain insight for dynamic legged locomotion," in *Proc. 2020 IEEE International Conference on Robotics and Automation*, 2020, pp. 2436–2442.

[9] D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim, "Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot," in *Proc. 2020 IEEE International Conference on Robotics and Automation*, 2020, pp. 2464–2470.

[10] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model predictive control," *arXiv preprint arXiv:2208.08373*, 2022.

[11] 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.

[12] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics*, vol. 36, no. 4, 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] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid," in *Proc. of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 629–634.

[15] A. Schmitz, M. Missura, and S. Behnke, "Real-time trajectory generation by offline footstep planning for a humanoid soccer robot," in *RoboCup 2011: Robot Soccer World Cup XV*. Springer, 2012, pp. 198–209.

[16] A. Stumpf, S. Kohlbrecher, D. C. Conner, and O. von Stryk, "Supervised footstep planning for humanoid robots in rough terrain tasks using a black box walking controller," in *Proc. 2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 287–294.

[17] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo: System overview and integration," in *IEEE/RSJ international conference on intelligent robots and systems*, vol. 3. IEEE, 2002, pp. 2478–2483.

[18] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
[19] S. Hong, J.-H. Kim, and H.-W. Park, "Real-time constrained nonlinear model predictive control on so(3) for dynamic legged locomotion," pp. 3982–3989, 2020.
[20] 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.
[21] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.
[22] 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, 2018.
[23] X. Su, X. Yan, and C.-L. Tsai, "Linear regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, no. 3, pp. 275–294, 2012.
[24] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.
[25] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," *Algorithmic and Computational Robotics: New Directions*, no. 5, pp. 293–308, 2001.
[26] S. Fahmi, V. Barasuol, D. Esteban, O. Villarreal, and C. Semini, "Vital: Vision-based terrain-aware locomotion for legged robots," *IEEE Transactions on Robotics*, pp. 1–20, 2022.
[27] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for Machine Learning*. Cambridge University Press, 2020.