A Hybrid Primitive-Based Navigation Planner for the Wheeled-Legged Robot CENTAURO

Alessio De Luca[1,2], Luca Muratore[1] and Nikos G. Tsagarakis[1]

***Abstract*— Wheeled-legged robots have the potential to navigate in cluttered and irregular scenarios by altering the locomotion modes to adapt to the terrain challenges and effectively reach targeted locations in unstructured spaces. To achieve this functionality, a hybrid locomotion planner is necessary.**

In this work we present a search-based planner, which explores a set of motion primitives and a 2.5D traversability map extracted from the environment to generate navigation plans for the hybrid mobility robot CENTAURO. The planner explores the map from the current robot position to the goal location requested by the user, considering the most appropriate composition and tuning of locomotion primitives to build up a feasible plan, which is then executed by the robot. The available primitives are prioritized and can be easily modified, added or removed through a configuration file. Our approach was evaluated both in simulation and on the real wheeled-legged robot CENTAURO, demonstrating traversing capabilities in cluttered environments with various obstacles.

***Index Terms*: Sensor-based Control, Motion and Path Planning, Legged Robots**

I. INTRODUCTION

Today, real world applications are still imposing high level challenges to robotics, demanding enhanced loco-manipulation skills combined with high level of autonomy from robots that are attempting to address such applications. Navigating efficiently and autonomously through cluttered terrains is certainly one of the most required skills for permitting a mobile robot to reach a target location within the environment to perform a task. To successfully navigate while negotiating terrain features and obstacles, mobile platforms need first to perceive the terrain features and the involved obstacles, then select an appropriate locomotion mode/strategy to tackle these terrain challenges based on their physical mobility capabilities that can be wheeled, legged or hybrid based.

To obtain the necessary information from the environment in order to navigate and interact with it safely, several methods have been proposed. The majority of these techniques considered the use of 2.5D elevation maps extracted from the point cloud of the environment. Few examples are [3], [4] and [5], where robots built the map from scratch using the on-board sensors in order to then perform path planning actions. These methods have demonstrated good results and are suitable to perform a search-based planning that is generally faster than sample-based methods in which a tree of whole-body actions is built like in [6]. In addition, when dealing with highly challenging tasks, as for example in the fields of disaster response or inspection and maintenance of damaged infrastructures, high mobility flexibility is required to adapt to potentially very unstructured terrains. Hybrid mobility robots merge the benefits of both legged and wheeled locomotion and can provide a richer repertoire of locomotion actions in such challenging applications. Several hybrid robots have been proposed in the last decade. CENTAURO [1], is a recent platform, which supports both locomotion modes thanks to its 4 articulated legs ending in 360° steerable wheels, while CHIMP [2] robot incorporates track elements in its limbs and can change the locomotion mode as needed to navigate and reach a target location. In DARPA Robotics Challenge (DRC), the winner of the 2015 edition, the DRC-HUBO robot [7] used wheels attached to the knees permitting it to change locomotion mode moving to a kneeing position. Another example is the hybrid robot MOMARO [8], which has a humanoid upper body and four legs ending in wheels. During DRC, the MOMARO robot was controlled remotely via teleoperation but in the following years the MOMARO team also introduced a framework for semi-autonomous locomotion and manipulation based on a set of motion primitives [9]. Here the researchers applied the A*-search on a grid map to perform path planning, in which

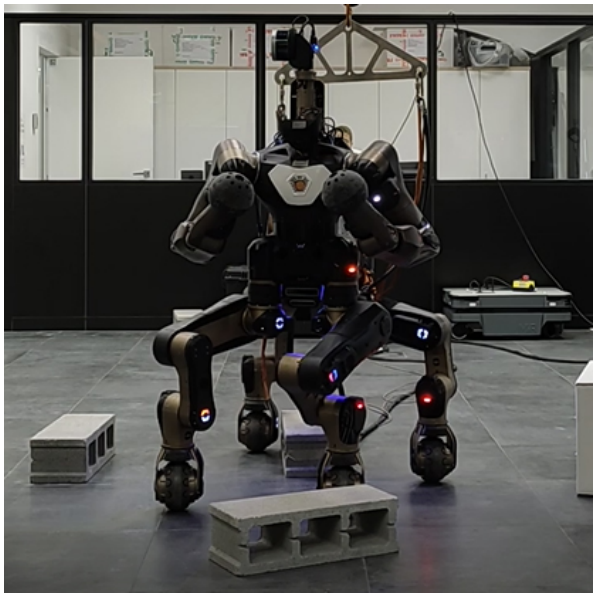

Fig. 1: CENTAURO robot performing single-wheel movements to negotiate the obstacles.

[1]Humanoids and Human-Centered Mechatronics Research Line, Istituto Italiano di Tecnologia (IIT), Via Morego 30, Genova 16163, Italy.E-mail:(alessio.deluca@iit.it; luca.muratore@iit.it; nikos.tsagarakis@iit.it)

[2]DIBRIS, Università di Genova, Italy, 16145.

stepping actions were included as abstract steps and later expanded in a trajectory. However, in this case there was still the need for a human operator that controls remotely the robot.

Similar to these results, in [10] a control framework to tackle the hybrid locomotion based on some motion modes (driving, walking and hybrid) was proposed. Also in this case the resulting framework did not show autonomous capabilities, in fact reinforcement learning techniques will be subsequently explored to select the motion modes and the velocities based on the terrain geometry, improving the foot placement planning.

Fully autonomous capabilities can be seen in [11], where the team from ETH introduced a new version of their quadrupedal robot ANYmal, adding wheels to the legs, proposing a hierarchical whole-body controller (WBC) and a motion planner. This work was extended in [12], which enabled the robot to perform stepping and driving simultaneously, decomposing the optimization problem in separate wheels and base Trajectory Optimization (TO). The results were also shown in the DARPA Subterranean challenge, revealing the advantages of wheeled-legged robots in real-world applications. The team also presented a combined sampling and TO based-planner for hybrid robots in [13], where the sampling stage manages the whole-body configurations, while the optimization stage satisfies the system constraints.

In our previous work [14] we demonstrated the autonomous capabilities of the CENTAURO robot (Fig. 1) with the use of a set of motion primitives concatenated by a Finite State Machine (FSM) to perform obstacle crossing tasks, being limited to deal with only rectangular shaped objects placed in series, decoupling driving and crossing tasks. In the work presented in this article instead, we extend the framework implementing a primitive-based planner, built upon the Anytime Repairing A* (ARA*) [15], allowing the robot to perform a search on the 2.5D traversability map extracted from the environment, enabling to deal with more complex scenarios. During the search, the nodes are expanded based on the wheels and pelvis position and the primitives available, selecting when feasible, the driving action over the stepping ones. In addition we developed a module that simplifies the planned solution by merging, whenever possible, multiple planned actions into one speeding up the execution.

The use of motion primitives permits us to describe parametrized atomic actions that the robot is allowed to consider during the planning and enable or disable them based on the capabilities of the robot platform used or the characteristics of the terrain. To summarize, the features and contribution of the proposed framework are:

- the definition and use of a set of motion primitives provided by the hybrid mobility robot CENTAURO and appropriately parametrized, permitting to generate the leg end-effectors positions in the defined reachable space,
- the introduction of a highly flexible, adaptable and extendable search-based planner built upon the ARA*, based on a set of motion primitives acquired at run-time from a configuration file, to perform a search on the 2.5D elevation and validity maps acquired from the perception system,
- the combination of multiple single actions to build more complex behaviours, called macro actions, speeding up the execution of the task,
- the autonomous navigation execution through terrains occupied by obstacles that necessitate mixed wheel and stepping actions to be traversed.

We demonstrate our framework in simulation environment and on the real CENTAURO robot evaluating the ability of the planner to find a solution in different real case scenarios based on the primitives available. The rest of the paper is organized as follows: in Section II we give an overview of the implemented framework; in Section III we describe how the hybrid planner works and in more details the motion primitives defined; in Section IV the validation studies and results are discussed and finally in Section V we outline the conclusions and future work directions.

II. Framework Overview

The proposed framework is composed of three main components: the traversability extractor, the hybrid primitive-based planner and the plan executor. A visual representation of the framework can be seen in Fig. 2.

For the traversability extractor we explored the method in [16] to create a 2.5D elevation map that is transformed into a binary map which we will call foothold validity map. This map takes into account the elevation and edges, extracted through the application of different filters together with the inflation radius specifying the minimum distance that we want to keep from the obstacles.

At the beginning of each test, the map is built online from scratch, rotating the robot to capture the elements around it. Of course, it is also possible to provide a pre-saved representation of the environment together with a correct initial localization of the robot. An example of the foothold validity map can be seen in Fig. 3A, where the traversable areas are presented in blue while those zones that are not considered safe due to edges or proximity to obstacles are indicated in red. The values of the filters applied take also in consideration the radius of the wheels in order to tune correctly the inflation radius near obstacles. This map is used by the proposed hybrid primitive-based planner in an offline way, without considering new updates of the map while the robot is moving. The planner takes also into account the available primitives and finds the sequence of primitive actions that allows the robot to achieve its goal. At this point, the identified plan is refined and some primitives are combined into one single *Macro* action. The macro that we considered up to now is the *Reshape*, which allows to execute simultaneously the single-wheel drive movements enabling to move independently the four wheels via driving, instead of moving just one at a time, speeding up the execution.

The output of the planner is a sequence of parametrized actions composed of pelvis and wheels position, together

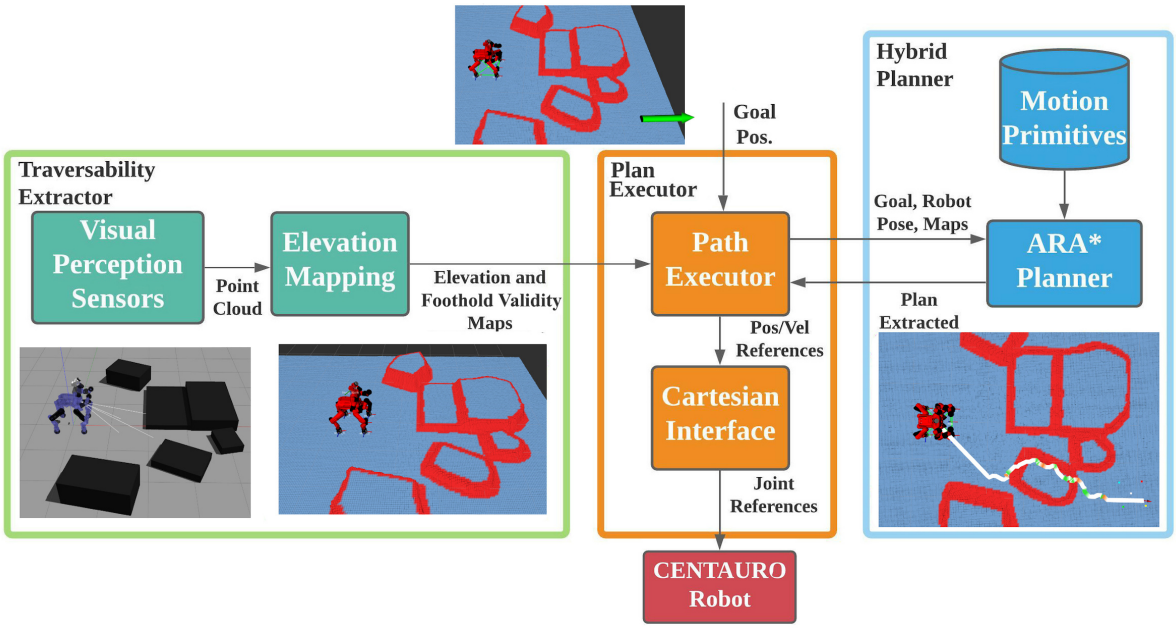

Fig. 2: Overview of the framework pipeline, in green the perception component, in orange the motion execution module and in blue the hybrid planner component.

with the corresponding action selected and wheel(s) involved. This sequence is used by the plan executor to generate the appropriate references (position or velocity) that are sent to the cartesian controller CartesI/O [17], which computes the Inverse Kinematic and sends the respective commands to the robot joints through the software middleware XBot [18]. The velocity of the movements is proportional to the distance traveled by the end-effector, considering also a maximum value defined by the user. In the following section we will illustrate in more details the proposed hybrid planner.

III. Hybrid Primitive-Based Planner

For the implementation of the hybrid planner we decided to employ a search-based approach. In particular, we implemented the ARA* to perform the search on the foothold validity map. The search is carried out by evolving the robot state in the map through the application of the primitives available to the planner, considering their costs, priorities and feasibility. The state-of-the-art ARA* was extended to a primitive-based one, gaining the ability to adapt the search according to the constraints defined and the motion primitives acquired during the initialization. The use of the Anytime version of the A* will permit us in the future to realize an online implementation of the planner, since the ARA* finds, in a small amount of time, the first sub-optimal solution which is refined until the available time expires.
In the ARA*, at each iteration of the main loop, the costs of the nodes are considered to expand the node with the smallest cost. In details the ARA* minimizes:

$$f(n) = g(n) + \epsilon\, h(n) \tag{1}$$

where n is the node, $g(n)$ is the cost from the starting position to the node n, obtained by adding the cost of the current action (explained in the next sections) to the previous node cost $g(n-1)$, $h(n)$ is the heuristic function, which estimates the cheapest path from n to the goal and it is evaluated as the sum of the distances along x and y in grid cells. Finally ϵ is used to provide sub-optimal solution and it is decreased in the following iterations. We start the search with $\epsilon = 4$, which offers, in our case, a good trade-off between planning time and optimality of the solution found.
For each node visited during the search, the neighbors are expanded based on the reachable space defined for the selected primitive. If the new f score of the neighbor, which is computed from the node that we are currently visiting, is smaller than the one saved in the previous iterations, then the neighbor is added to the list of nodes to be visited.

A. Motion Primitives

The use of a set of motion primitives provides great flexibility permitting to shape and extend the framework in order to adapt it to the primitive actions offered by a robotic platform. The primitives considered in this work for the case of the CENTAURO robot platform, are whole robot driving and single-wheel motion. The latter is then internally subdivided into single-wheel driving and stepping based on the elevation difference in the trajectory examined. If needed, we can also treat these two sub-actions as separate primitives and enable just one of them, based on the platform considered and map requirements. These motion primitives allow to exploit the capabilities of our hybrid robotic platform to be able to accomplish the desired tasks by employing different locomotion strategies concatenating autonomously the atomic actions in different order. We defined these primitives considering a number of different parameters that can be easily changed via a configuration file. Using these

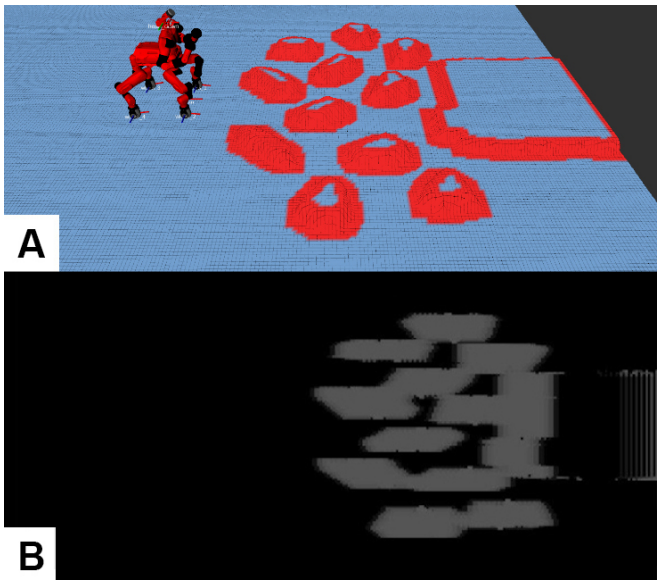

Fig. 3: (A) The binary foothold validity map where the red elements represent untraversable areas and the blue the safe parts of the environment. (B) The scaling factor map associated with the map above. Black means that there are no obstacles in the surrounding space, so no scaling is needed (preferred driving), grey instead shows areas close to obstacles in which single wheel actions may be needed.

parameters it is possible to enable/disable the actions, change the costs assigned, reshape the reachable space considered or also modify the clearance to keep around the wheels for avoiding the collision with the obstacle.
All the parameters were experimentally tuned by performing several trials with different obstacle arrangements in order to obtain a more efficient planner, both in terms of search time and optimality of the solution. Although, a change in the parameters of the primitive actions' costs does not prevent the planner to find a solution, even if less optimal. However, changing the parameters of the primitives like reachable space or safety thresholds can, of course, challenge and eventually prevent the planner to find a solution due to the more constraints imposed on the motion capabilities, preventing the robot to adapt to a complex scenario.
The following paragraphs present in details the two primitives implemented:

1) Whole Robot Driving: this primitive allows to perform the driving action with the whole robot enabling the robot to translate and/or rotate. This action is selected if all the wheels are not close to any obstacles. The above check was incorporated in the map with the introduction of the inflation radius to speed up the planner computation, avoiding repetitive loops checking the neighbors of the wheels position. The driving action is parametrized by the maximum distance and rotation that can be achieved with a single use of the action. Modification to these parameters are allowed, but note that increasing them will result in longer planning time because, for each state considered, there will be more possible neighbors to be visited. We experimentally decided to consider a maximum translation of 0.08m and rotation of $\pm 10^o$ for each application of the action.
The cost defined for this action, C_D, is the following:

$$C_D = (|i| + |j|)\ (O_d + (\ G_e|D_e| + G_d + |\alpha|) + C_i + Ca) \quad (2)$$

where i and j represent the movement (in grid cells) along x and y direction respectively, O_d = 35 is the offset for the driving corresponding to the minimum cost of the driving action, G_e = 24 is the gain for the elevation difference (D_e) of the wheels before and after the primitive motion, G_d = 2 is the cost associated to the driving action and α is associated with the rotation angle considered.
C_i = 0.75 O_d and C_a = 10 R_{diff}, which are additional costs used to, respectively, score more the nodes close to objects based on an inflation radius defined (i.e. 20 cm) and based on the alignment of the robot in order to prefer actions that allow to face the path being followed, having R_{diff} the rotation difference between the actual yaw and the desired one resulting in such alignment.
This definition allows to consider different possibilities and prefer the configurations in which the robot is driving looking forward, far enough from the obstacles and with the wheels at the same elevation level, so driving in flat areas.

2) Single-Wheel Action: with respect to our previous work [14], here we do not have the need to provide separate actions: reshape of the support polygon, stepping on, over and down. In this implementation we have a single-wheel action that allows to move the wheel in the reachable space considering the difference in the elevation for the trajectory connecting the starting and ending position of the wheel. In this way, if the elevation difference is smaller than the radius

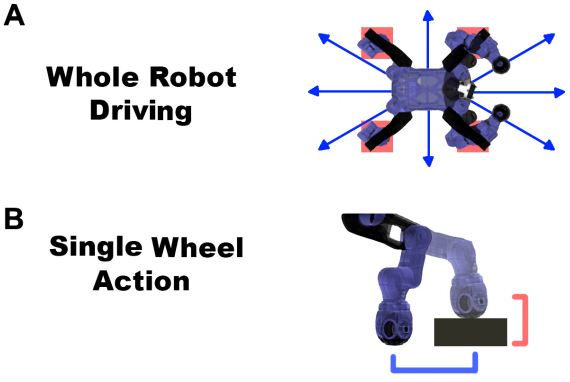

Fig. 4: Motion primitives considered: (A) whole robot driving, (B) single-wheel action. In (A) the red area around the wheels is used to show the clearance to enable the action, the blue arrows represent the omnidirectional movement that can be performed with one action. In (B) we check the maximum distance traveled and that is constrained to other wheels, together with the difference in the elevation during the trajectory.

of the wheel, we can consider driving assuming that such elevation difference can be overcome by the wheel action alone. Otherwise, we have to lift the wheel, based on the maximum elevation in the path followed and place it in the target position. This permits to deal with gaps, obstacles or flat surfaces. The single-wheel action is considered by the planner only if the whole robot driving is not possible in a neighborhood of the considered node. Moreover its cost is higher than that of the whole robot driving in order to favor longer whole robot drive, if possible, with respect to multiple single-wheel actions alternative solution which would be slower and more complex.

Since in this action we move one wheel, we added also a check on the support polygon that changes based on the wheel movement to discard the configurations in which the stability cannot be ensured. In addition, for the stepping actions we take into account the pelvis movement to ensure static stability. The reachable space of the wheels is translated based on the current pelvis position and the stepping actions are considered only if the support triangle allows to preserve the stability while lifting the wheel. To increase the stability we added in the cost definition also a measure of the area of the support polygon and the distance of the CoM with respect to the edges of the polygon to consider, if possible, more stable configurations.

The cost of the single-wheel action, C_{OW}, is therefore described by the following:

$$C_{OW} = s\ D_p(O_o + D_w + G + S_d + S_a + G_e\ |D_e|) + C_i \tag{3}$$

where s is the scaling factor based on the elevation difference for the node considered (explained in the Section III-B), O_o = 115 for the single-wheel driving and O_o = 240 for the stepping are the minimum costs of these actions. D_p and D_w represent the distance traveled by the pelvis and the wheel respectively, S_d and S_a are the costs proportional to the distance of the CoM to the support polygon and the area of the polygon and G is the gain associated with the motion selected, for driving ($G_d = 2$) and for stepping ($G_s = 5$).

The parameters of the cost functions were experimentally tuned by performing several trials in simulation and are defined in such a way to have an admissible heuristic and to score differently the actions, based on our objectives. In addition, for each primitive considered, we included a check on the elevation under the pelvis to avoid navigating if the motion would result in a collision between the pelvis of the robot and an obstacle higher than clearance height below the pelvis. A graphical representation of the two primitives can be found in Fig. 4.

To sum up, we defined the two primitives providing a set of parameters that can be easily changed via configuration files. These parameters contain: offsets and gains to define a preliminary priority, reachable space of the end effectors, minimum and maximum distance among the wheels, size of the clearances for safety reasons, maximum elevation that can be reached for stepping actions, minimum values for the support polygon area and distance of the CoM from the support polygon edges. By modulating them we can define how important are those factors, for example, we can increase the offset of the single wheel actions to prefer much longer driving rather than a sequence of stepping actions to cross obstacles.

B. Primitive-Based Motion Planner

Once the map has been computed, the operator needs only to send the target position that the robot has to reach. At this point the planner starts computing the h score for all the nodes considered, based on the Euclidean distance (maximum between the x and y) with respect to the goal position. In addition, to speed up the computation, we compute also the scaling factor for each possible node in the map comparing its elevation with the one of its neighbors. This scaling factor has the objective of reducing the cost of the single-wheel actions as these primitives are computationally more expensive. We therefore considered, at the beginning, an over-estimated offset score for single-wheel actions that, thanks to the scaling factor, is reduced for those nodes surrounded by objects and so more inclined to need single-wheel actions. The information of the scaling factor, in more details, is estimated by looking at the neighbors of the nodes. If the elevation difference is smaller than the wheel radius, no scaling factor will be applied since we will not need to step. On the other hand, if the elevation difference is higher than the wheel radius, the scaling factor will change to facilitate the use of the single-wheel actions, speeding up the searching. Fig. 3B presents the scaling factor map associated with the map above. In particular, we can notice that the black areas are the ones where the scaling factor is 1, so no scaling is applied corresponding to an area in which there are no objects around and driving can be performed safely. The grey color instead is relevant to areas with obstacles, meaning that it may be necessary to use single-wheel actions to navigate in that area.

At this point the actual search can start. Here the planner employs the primitives enabled with a priority and costs assigned to explore the nodes in the map, which are characterized by the pelvis position and orientation together with other information such as: wheels position, primitive selected and wheel moved. The ARA* iteration exits when the smallest f score in the list of nodes that we have to expand is higher than the f score of the goal. At this point, the planned solution is extracted and becomes ready to be executed. The ϵ value is decreased, the f scores of the nodes that were not visited are updated based on the new ϵ and a new iteration of ARA* can start. The procedure is iterated until ϵ converges to 1, which corresponds to the optimal solution, or when a new goal is assigned.

IV. Validation Study and Results

To demonstrate the correctness and the performance of our framework we run experiments in the Gazebo simulation environment and then on the real CENTAURO robot.

Thanks to the feature that allows us to enable/disable the primitives, we started considering only the driving with the

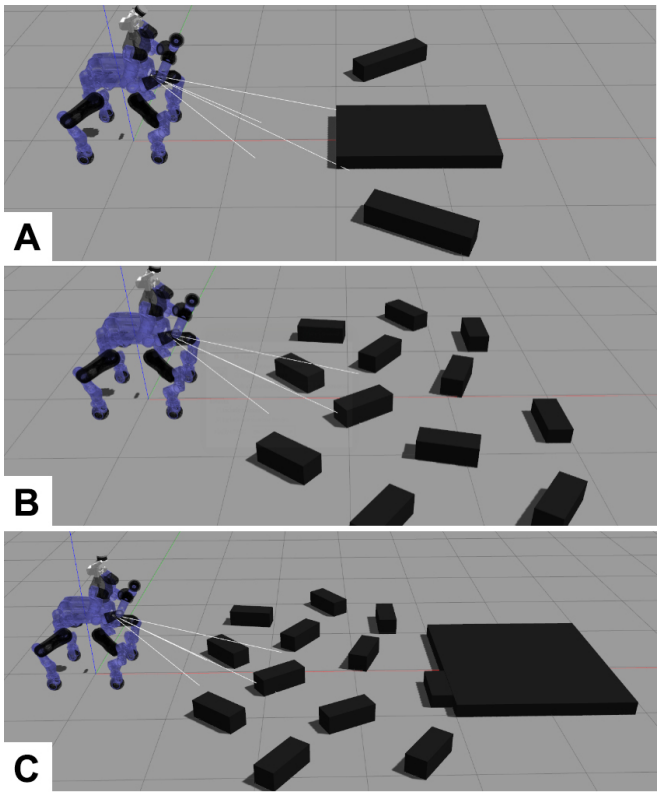

Fig. 5: A set of considered scenarios to evaluate the framework in simulation. (A) Simple environment in which only whole-robot driving is required. (B) Cluttered scene that needs reshape of the support polygon. (C) Introduction of a platform, requiring the use of stepping actions.

whole robot, then we added the single-wheel driving to perform the reshape of the support polygon and finally we introduced the stepping action, in which the wheels can be lifted to cross obstacles. To acquire information from the environment we used the Velodyne Puck 16 [1], a 3D LiDAR with a resolution of 0.03 meters. This resolution was slightly decreased for the map definition, more precisely to 0.04 meters, avoiding to loose too many details by selecting a higher value but introducing at the same time a smoothing filter for reducing the noise in the map.

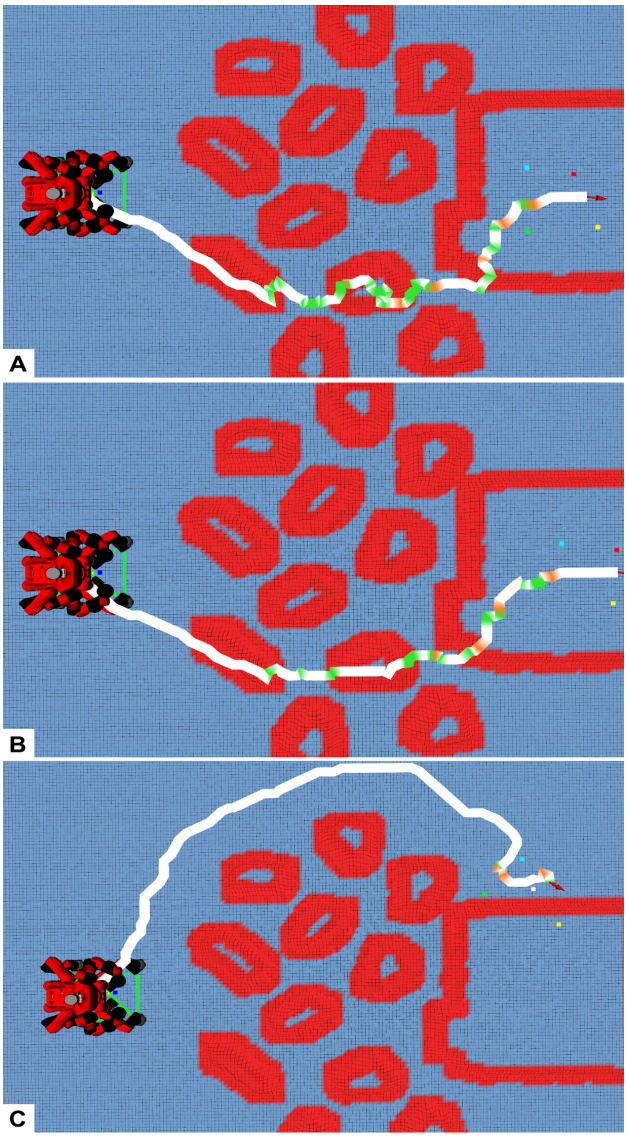

Fig. 6: Examples of foothold validity map and corresponding plan to reach a target in different positions, starting from the same pose. The colored line shows the planned CoM trajectory and the color identifies different actions: whole robot driving in white, single-wheel driving in green and single-wheel stepping in orange.

A. *Simulation*

To evaluate the correctness of the planner implemented we started with a simple scenario in which there are few obstacles on the ground, far enough from each other, permitting to move the robot to the goal position via whole robot driving only. To make the experiment more interesting, we placed a big obstacle in front of the robot and two thinner ones on the sides so that the resulting plan guides the robot over the smaller obstacles, avoiding a simple forward navigation. The scenario can be seen in Fig. 5A. With this scenario the robot was able to complete the task in all the 10 tests performed, reaching the target without colliding with objects. The plan was found in about $5\,\mathrm{sec}$ and the total distance travelled was approximately $5.15\,\mathrm{m}$. Following this, we changed the setup and considered a series of brick obstacles placed in a random configuration in front of the robot as in Fig. 5B. To deal with this scenario, we had to enable the single-wheel driving in order to permit the robot to change its support polygon and move the wheels among the obstacles. Also in this case the robot was able to complete all the 10 tests without failures. Here the time required to find a solution was $5.8\,\mathrm{sec}$. for an overall movement of $5.1\,\mathrm{m}$.

[1] https://velodynelidar.com/products/puck/

Finally we increased the complexity of the scenario by adding a platform in which the robot has to get on as shown in Fig. 5C. In this case we considered also the stepping action, requiring the robot to extricate from random objects on the ground and finally step on the platform. Here the robot completed successfully the task 13 times out of 15 tests performed. The plan was found in all the tests, requiring on average $9.8\,\mathrm{sec}$ and the target position specified was reached travelling for $5.1\,\mathrm{m}$. The two failed trials were due to a wrong estimate of the obstacles in the map that brought the robot to fall, moving the wheels on the edges of the obstacles.
In Fig. 6 you can see the resulting plans associated with three runs of the last experiment. To show the capabilities of the planner, we considered different target locations in a neighborhood of the center of the platform. Based on the target specified, the planner reacts providing a different sequence of actions. For example, in Fig. 6C we set a target on the left edge of the platform and the robot completed the task by driving around the bricks and then stepping on the platform. The colored line shows the planned CoM movements along the map in which the colors are used to highlight the different actions selected: white for whole-robot driving, green for single wheel driving and orange for the stepping action. As you can see, in all the three plans shown the most used action is the whole robot driving since we imposed this as the one with the highest priority, limiting the use of the other primitives only when necessary. In addition, it can be observed that, close to the stepping actions there are single wheel movements that are used to reshape the support polygon and permit to lift a wheel without losing balance.
In the plan found also the poses of the wheels are obtained but in the Figure we omitted this information for each state to keep the image readable.

B. Real Robot

Similar experiments were carried out also on the real robot CENTAURO. As for the simulations, we considered different scenarios of increasing complexity, requiring to add a primitive in order to reach the target specified.
In Fig. 7 you can see the setup considered for the final experiment on the real robot, together with the foothold validity map obtained and the corresponding plan found by the planner. In this case the planner was able to find a the first feasible solution in approximately $8.1\,\mathrm{sec}$ requiring the robot to pass over the two obstacles, adjusting its support polygon and then stepping on the platform while keeping the equilibrium during the execution. The only input from the operator is the target location to reach by the robot, then both planning and execution are done in an autonomous fashion.
During these tests the planner was always able to provide a solution if there was enough valid space in the map between the objects. The only problem that we encountered was related to the localization errors that sometimes made the robot slightly collide with the obstacles during the execution. In particular for the localization of the robot we used the Intel Realsense T265[2] camera and we experienced an error in the localization proportional to the distance traveled, so we handled this by adding a correction to the planned sequence in order to compensate the localization error. The extended video of the experiments in simulation and with the real robot can be accessed in the linked video [3].

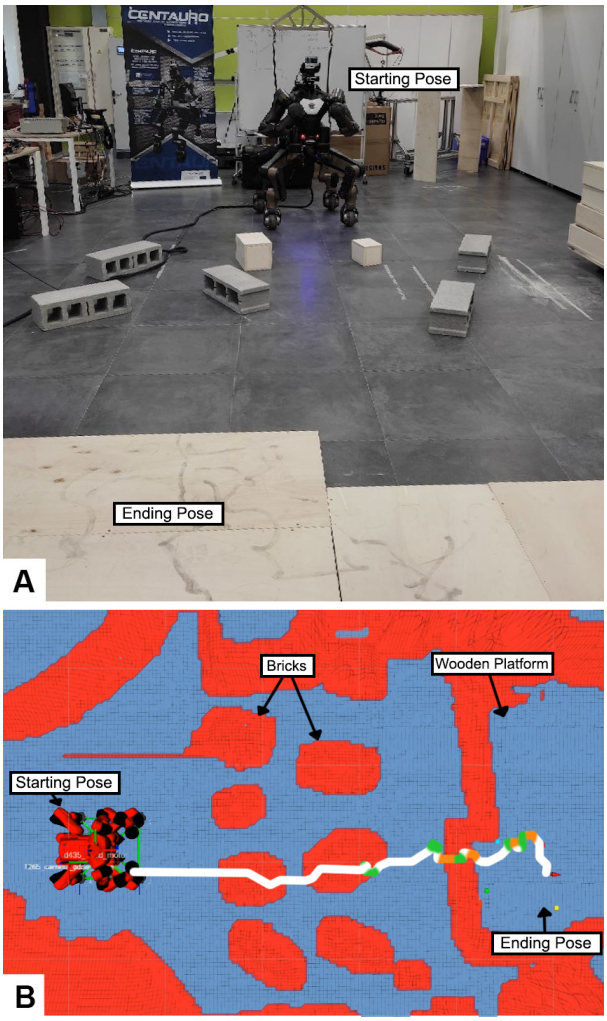

Fig. 7: (A) Real scenario considered to test the implemented framework. (B) Corresponding foothold validity map and plan found by the primitive-based planner.

The current implementation has still some limitations that will be addressed in future works. First of all, map updates cannot be performed while the robot is moving. This will be tackled with an online version that we are currently working on and which permits to update continuously the plan in the proximity of the robot to deal with changes in the environment and localization errors. In addition, occlusions of ground areas hidden by the obstacles prevents the perception of the elevation in those cells, which in this

[2] https://www.intelrealsense.com/tracking-camera-t265/
[3] https://www.youtube.com/watch?v=5w6est-syVM

work is indirectly obtained through the application of filters that averages the known elevation in the neighbourhood of the hidden areas, provoking an overestimate of the obstacles. This can prevent a solution to be found or bring the wheels too close to the edges when stepping on and down resulting in a possible failure of the execution. Currently we dealt with this issue by increasing the required clearance distances from the obstacle. The online updates will allow us to deal more effectively and with high robustness also with these occluded terrain areas due to obstacles.

V. Conclusions

In this work we presented a new hybrid search-based planner for the wheeled-legged robot CENTAURO. It employs a set of motion primitives that are easily extendable and can be customized based on the platform physical capabilities and navigation needs, searching on the foothold validity map applying the primitives based on their feasibility and priority. The planner demonstrated good efficacy in addressing terrains with different complexity and obstacle composition and permitted to execute autonomously navigation tasks successfully negotiating these terrain environments. Future work plans to include the introduction of wheeled and legged motion simultaneously together with manipulation primitives to push objects that block the way. An online execution implementation of the developed planner forms another direction of our future work plans: this will allow the robot to deal with dynamic obstacles and new entities that may intervene while the robot is navigating in the environment as well as to be more robust against small changes in the scenario and localization errors.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101016007 CONCERT, the Italian Fondo per la Crescita Sostenibile - Sportello "Fabbrica intelligente", PON I&C 2014 - 2020, project number F/190042/01-03/X44 RELAX and Leonardo Centauro project (code ETCM053501).

References

[1] N. Kashiri et al., "CENTAURO: A Hybrid Locomotion and High Power Resilient Manipulation Platform," in *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1595-1602, April 2019, doi: 10.1109/LRA.2019.2896758.

[2] A. Stentz et al., "Chimp, the CMU highly intelligent mobile platform", in *Journal of Field Robotics*, vol. 32, no. 2, pp. 209–228, March 2015, doi: 10.1002/rob.21569.

[3] D. Belter, P. Łabcki and P. Skrzypczyński, "Estimating terrain elevation maps from sparse and uncertain multi-sensor data," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2012, pp. 715-722, doi: 10.1109/ROBIO.2012.6491052.

[4] P. Fankhauser et al., "Collaborative navigation for flying and walking robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2859-2866, doi: 10.1109/IROS.2016.7759443.

[5] P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki and M. Hutter, "Robust Rough-Terrain Locomotion with a Quadrupedal Robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5761-5768, doi: 10.1109/ICRA.2018.8460731.

[6] A. Settimi, D. Caporale, P. Kryczka, M. Ferrati and L. Pallottino, "Motion primitive based random planning for loco-manipulation tasks," in *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 1059-1066, doi: 10.1109/HUMANOIDS.2016.7803402.

[7] Matt Zucker, Sungmoon Joo, Michael X. Grey, Christopher Rasmussen, Eric Huang, Michael Stilman, and Aaron Bobick, "A General-purpose System for Teleoperation of the DRC-HUBO Humanoid Robot", in *J. Field Robot* 32, 3 (May 2015), 336–351. DOI:https://doi.org/10.1002/rob.21570

[8] M. Schwarz, T. Rodehutskors, M. Schreiber and S. Behnke, "Hybrid driving-stepping locomotion with the wheeled-legged robot Momaro," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5589-5595, doi: 10.1109/ICRA.2016.7487776.

[9] T. Klamt and S. Behnke, "Anytime hybrid driving-stepping locomotion planning", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4444-4451, doi: 10.1109/IROS.2017.8206310.

[10] J. Sun, Y. You, X. Zhao, A. H. Adiwahono and C. M. Chew, "Towards More Possibilities: Motion Planning and Control for Hybrid Locomotion of Wheeled-Legged Robots," in *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3723-3730, April 2020, doi: 10.1109/LRA.2020.2979626.

[11] M. Bjelonic, C. D. Bellicoso, Y. de Viragh, D. Sako,F. D. Tresoldi, F. Jenelten and M. Hutter, "Keep Rollin' – Whole-Body Motion Control and Planning for Wheeled Quadrupedal Robots", in *IEEE Robotics and Automation Letters*, vol 4, pp. 2116-2123, April 2019, doi:10.1109/LRA.2019.2899750.

[12] M. Bjelonic, P. K. Sankar, C. D. Bellicoso, H. Vallery, and M. Hutter, "Rolling in the Deep – Hybrid Locomotion for Wheeled-Legged Robots Using Online Trajectory Optimization", in *IEEE Robotics and Automation Letters*, vol. 5, pp. 99, 2020, doi: 10.1109/LRA.2020.2979661.

[13] E. Jelavic, F. Farshidian and M. Hutter, "Combined Sampling and Optimization Based Planning for Legged-Wheeled Robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8366-8372, doi: 10.1109/ICRA48506.2021.9560731.

[14] A. De Luca, L. Muratore, D. Antonucci, N. G. Tsagarakis, "Autonomous Obstacle Crossing Strategies for the Hybrid Wheeled-Legged Robot Centauro", in *Frontiers in Robotics and AI*, November 2021, doi:10.3389/frobt.2021.721001.

[15] Maxim Likhachev, Geoff Gordon, and Sebastian Thrun, "ARA*: anytime A* with provable bounds on sub-optimality", in *Proceedings of the 16th International Conference on Neural Information Processing Systems (NIPS'03)*, 2003, MIT Press, Cambridge, MA, USA, pp. 767–774, doi: 10.5555/2981345.2981441.

[16] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, "Robot-Centric Elevation Mapping with Uncertainty Estimates", in *International Conference on Climbing and Walking Robots (CLAWAR)*, 2014.

[17] A. Laurenzi, E. M. Hoffman, L. Muratore and N. G. Tsagarakis, "CartesI/O: A ROS Based Real-Time Capable Cartesian Control Framework," in *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 591-596, doi: 10.1109/ICRA.2019.8794464.

[18] L. Muratore, A. Laurenzi, E. Mingo Hoffman and N. G. Tsagarakis, "The XBot Real-Time Software Framework for Robotics: From the Developer to the User Perspective," in *IEEE Robotics & Automation Magazine*, vol. 27, no. 3, pp. 133-143, Sept. 2020, doi: 10.1109/MRA.2020.2979954.