Design and performance evaluation of receding horizon controllers for quadrupedal robots: case study on stairs climbing and balancing ⋆

Artem A. Egorov * **Maxim V. Lyahovski** * **Denis A. Sokolov** *
Alexey M. Burkov * **Sergey A. Kolyubin** *

* *Biomechatronics and Energy-Efficient Robotics Lab, ITMO University, St. Petersburg, Russia*
e-mail: {aaegorov, amburkov, s.kolyubin}@itmo.ru

Abstract: This paper describes the experience of controlling the locomotion of a legged robot related to small-sized four-legged robots. A convex model predictive control was used to stabilize the robot's body and control its position as the main tool. The original solution was augmented with capabilities to perceive early contact, move on sloping surfaces, estimate height with less error, and detect the height of obstacles. All improvements were tested on a staircase and compared with the original solution.

Keywords: Mobile robots, Perception and sensing, Information and sensor fusion.

1. INTRODUCTION

This paper describes the principles and features of controlling a walking robot. A control architecture has been developed for walking four-legged robot with a three-step serial kinematics leg mechanism, and heuristics have been added to improve the quality of motion. All results were tested on the Unitree A1 robot while walking stairs.

However, before diving into the details, it is worth highlighting the advantages and disadvantages of walking robots, which will be worked further with. The main advantage is the ability to break contact with the surface in a controlled manner, thereby stepping over obstacles. However, it should be immediately clear that this depends on the ability to determine how to step over and stabilize itself. This relationship strongly affects the robot's configuration - more complex algorithm, heavier robot, more powerful motors. This requires the development of both detailed and simplified solutions. In addition to the high cross-country capability compared to wheeled robots, such robots have the ability to hold stable configurations on difficult surfaces, thus preventing falling over. A final advantage worth noting is the ability to use their multi-step kinematics and dynamics when interacting with objects, which increases the variability of the way they solve problems. For example, humans use the torque created by their body weight to open heavy doors, which mobile robots are unable to do.

However, these advantages require robots to solve the real-time stabilization problem, reduce the speed of movement, and use full hybrid dynamics, the model of which depends on the presence of legs in contact. It is described in more detail, for example, in one of the first fundamental books about robot locomotion Raibert (1986).

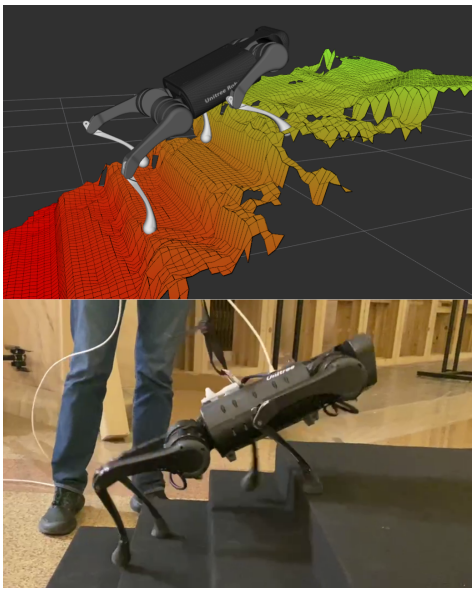

Fig. 1. Illustration of the quadruped walking stairs with visual feedback.

Such advantages are commercially viable in human-centered environments because they do not require changes to the environment and lower speeds are welcome. Examples of such tasks are condition monitoring and compliance with safety measures in factories (Afsari et al. (2021)), mapping cave systems, creating virtual twins of rooms, escorting people with disabilities, searching and rescuing people in emergency situations, etc. (Kolvenbach et al. (2020)).

⋆ Work of students A. Egorov, M. Lyahovski and D. Sokolov is supported within ITMO University MSc and PhD students' support program (project 620164).

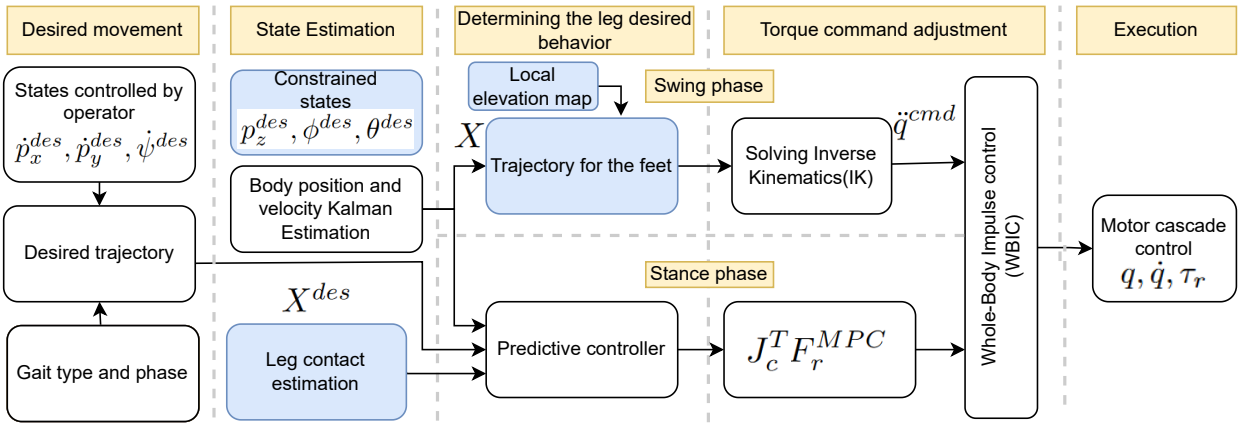

Fig. 2. Control system architecture similar to Carlo et al. (2018). The blue color indicates the stages in which the upgrade was made.

Two major modern approaches used for controlling quadrupeds are optimal and receding-horizon controllers and reinforcement learning. We see that the latter demonstrates impressive results as it is reported in the most recent papers. For example, Hwangbo et al. (2019) describes RL-based controllers that were trained in a few hours in simulation which uses trust region policy optimization method Schulman et al. (2015) outperforming the best existing model-based controllers running on the same robot. A research paper Rudin et al. (2022) combines local navigation and locomotion of legged robots by training an end-to-end policy with deep reinforcement learning. Training a policy in this way opens up a larger set of possible solutions, which allows the robot to learn more complex behaviors.

However, as for now our personal experience with RL-based approaches demonstrates that there is still a simulation-to-reality gap keeping aside the problem of reward engineering and providing guaranteed performance within unstructured environment and with limited computational resources. Therefore, we focus our current work on modern nonlinear and optimal control methods.

Over the past few years, since the work of Mini Cheetah Carlo et al. (2018), various modifications and alternative frameworks for MPC-based controllers design for four-legged robots have been released. Nevertheless, it is still difficult to reproduce and adapt these works for various robots, since the algorithms have many parameters and heuristic coefficients that are hardwired into the code, and it is not explained how to configure and debug them. This is a problem since the MPC algorithms require an accurate description of the model, and the PID coefficients need to be tuned during the operation of the robot. Thus, the purpose of this work is not only to present and verify state-of-the-art receding-horizon controller within four-legged robots control scenarios, but also provide practical recommendations on how baselines should be modified for real-robot implementation.

We also address the problem of perception-enabled locomotion. To successfully overcome various kinds of obstacles, it is necessary to have a system that will identify and describe these obstacles. Such a system can be based on visual sensors information, such as cameras, 2D of 3D lidars. Based on this data a map of the robot's environment is built, containing information about the location and height of objects in some neighborhood around the robot. This local robot-centered map is further used to plan support points of the robot's paws when planning the trajectory of the limbs and the entire body of the robot.

The rest of the paper is organized in the following way. Section 2 will discuss quadrupedal locomotion control. In section 3 we focus on planning references for robot body pose and feet locations both in blind and sensor-enabled walking. Section 4 is presenting experimental results to verify performance of the presented control system for Unitree A1 quadrupedal robot walking stairs. And then we conclude the paper with discussion on the obtained results and future steps.

2. LOCOMOTION CONTROL

2.1 Body stabilization

Two schools of principle can be distinguished among the methods for controlling walking robots: model-based optimal control and neural network reinforcement learning. We have considered methods that can be run on the internal computer of a relatively small walking robot, for example, Unitree A1 from the Figure 1. Such methods allow not to overload the robot and look for new computationally efficient stabilization solutions. From a commercial point of view, the systems should be more flexible and scalable, which means that they should be able to work under limited conditions.

To begin, based on the diagram from Figure 2, let's break down what tasks the controller has. To move successfully, it is necessary to determine the desired trajectory of the robot's body, plan the sequence of steps (gait) and the positions of the foot-surface contact points. Then design the leg movement to the next point of contact, and then calculate the forces to stabilize the body for each unique set of supporting legs. In addition, a separate problem is the estimation of the state of the robot.

One of the basic model-based optimal control methods is the linear-quadratic regulator (LQR). The work (Chignoli and Wensing (2020)) describes the application of such a controller based on a variational model for balancing a

robot on two legs. The controller uses a linearized model of the robot body dynamics to solve an integral quadratic criterion for the state error and the magnitude of the support reaction forces, which are then translated into torques in the motors. Due to the fact that the linearized model and the quadratic criterion together represent a convex optimization problem, the uniqueness of the extremum can be guaranteed. Due to this approach, developers can safely use numerical solution search methods which increases robustness. However, already at this step the question arises: should the dynamics be linearized with respect to the current or desired state of the robot body.

A further extension over LQR is the model predictive controller (MP), the application of which is described in Carlo et al. (2018). The authors of the paper opted for MP because it showed better stability to unpredictable short-term perturbations. The MPC also used a linearized model near the desired state of the robot's body. The next step to improve the stabilization method was to add heuristic dependencies between the desired and current states of the system (Bledt et al. (2017)). Thus, walking characteristics were taken into account and the quality of control was improved. For example, the dependence of the desired body pitch angle on the velocities along the axes. The authors developed a software product for extracting heuristic dependencies (Bledt and Kim (2020)).

However, there are also many solutions to nonlinear quadratic optimization problems, for example Hutter (2013). This paper uses a Nonlinear Quadratic Regulator, but then it evolved into a Nonlinear MPC (Hutter et al. (2016)). However, these are time-consuming calculations, especially when calculating on the forecast horizon, so a combination of whole-body control and MPC named as whole-body impulse control (WBIC) for a small-scaled quadroped has become more popular (Bledt et al. (2017); Carlo (2020)). We have considered these controllers because this combination allows us to quickly and qualitatively determine forward and backward control. Further results will be achieved based on the convex model predictive control formulation from the Carlo et al. (2018). The state system (4) is derived from writing Newton's second law for one rigid body representing the translational (1) and rotational (2) motion of the robot. The weight of the legs, according to the version of the paper, is 10% of the total robot mass and their dynamics can be neglected. We checked with the data to see if this is true for the Unitree A1, before using the controller.

$$\ddot{p} = \frac{\sum_{i=1}^{n} f_i}{m} - g \tag{1}$$

where p, m, f_i, g are correspondingly the position and mass of the robot body, the reaction force on the i-th leg and the acceleration of gravity.

$$\frac{d}{dt}(I\omega) = I\dot{\omega} + \omega \times (I\omega) = \sum_{i=1}^{n} r_i \times f_i \tag{2}$$

where w, I, r_i are correspondingly the angular velocity and inertia of the robot body, the vector from center of mass to contact point on the i-th leg.

$$\dot{R}(\phi, \theta, \psi) = [\omega] \cdot R(\phi, \theta, \psi) \tag{3}$$

where $R(\phi, \theta, \psi) = R_z(\psi)R_y(\theta)R_x(\phi)$ orientation of the body described through rotation matrices with respect to the axes in world frame. $[\omega]$ means take a skew-symmetry matrix from vector $a\omega$. Then the linearization was performed near the horizontal position of the robot's body and the following model was obtained by assuming $\omega \times (I\omega) \sim 0$.

$$\begin{gathered}\frac{d}{dt}\begin{bmatrix}\hat{\Theta}\\ \hat{p}\\ \hat{\omega}\\ \hat{\dot{p}}\end{bmatrix} = \begin{bmatrix}0_3 & 0_3 & R_z^T(\psi) & 0_3\\ 0_3 & 0_3 & 0_3 & 1_3\\ 0_3 & 0_3 & 0_3 & 0_3\\ 0_3 & 0_3 & 0_3 & 0_3\end{bmatrix} \cdot \begin{bmatrix}\hat{\Theta}\\ \hat{p}\\ \hat{\omega}\\ \hat{\dot{p}}\end{bmatrix} + \\ \begin{bmatrix}0_3 & \dots & 0_3\\ 0_3 & \dots & 0_3\\ \hat{I}^{-1}[r_1] & \dots & \hat{I}^{-1}[r_n]\\ 1_3/m & \dots & 1_3/m\end{bmatrix} \cdot \begin{bmatrix}f_1\\ \vdots\\ f_n\end{bmatrix} + \begin{bmatrix}0\\0\\0\\g\end{bmatrix}\end{gathered} \tag{4}$$

where $0_3, 1_3, \hat{\Theta}$ are correspondingly zero value and identity matrix 3×3 and the robot body global rotational angle. To bring the model to a linear form, we need to add one more condition to the 12 states: $\dot{x}_{13} = 0; x_{13}(0) = g$.

After defining the model, a quadratic programming problem is generated, relative to the state error and the value of the control on the forecasting horizon based on the model. As for Whole-Body Impulse Control (WBIC)t is used a full robot dynamics to optimaze for torque values with 10 times higher frequency than MPC (Carlo et al. (2018)).

$$\begin{gathered}\min_{\delta f_r, \delta a} \sum_{i=0}^{k-1} \|\delta a\|_{Q_1} + \|\delta f_r\|_{Q_2} \\ S_f(M\begin{bmatrix}\ddot{q}_f\\ \ddot{q}_i\end{bmatrix} + C + G) = S_f J_c^T f_r \\ \ddot{q} = \ddot{q}^{cmd} + \begin{bmatrix}\delta a\\ 0\end{bmatrix} \\ f_r = F_r^{MPC} + \delta f_r\end{gathered} \tag{5}$$

where $\|\delta a\|_{Q_1}$ is the quadratic weighted norms with diagonal positive matrix Q_1 and Q_2, M, C, G is the mass matrix, Coriolis and gravity vector, respectively, for the dynamics of the full robot. S_f is the matrix leaving only the equations of floating base dynamics. $\ddot{q}$ and f_r are determining the desired position and controlled reaction forces. J_c^T is a contact leg Jacobi matrix. $\ddot{q}^{cmd}$ and F_r^{MPC} are the body acceleration calculated from inverse kinematics, and the reaction forces obtained from the MPC at the previous step. Finally, the whole point is that the solution of the optimal control is only a small adapting of the previously obtained acceleration δa and force δf_r results , which speeds up the calculations (Carlo (2020)).

3. GAIT PLANNING AND ADAPTATION

3.1 Behaviour planning

After determining how to stabilize the robot, it is worth thinking about what options there are for setting the desired task. High-level planning can be either direct acceptance of speed commands from the operator (Carlo et al. (2018); Bledt et al. (2017)) or processing them

through the motion planner and surface contact positions (Bledt et al. (2018); Mastalli et al. (2020); Winkler et al. (2018)). The movement of the body is related to the legs and vice versa, which requires the definition of constrained motion. The most common way for finding the next contact point is called the Raibert heuristic (Raibert (1986)), which was later refined by adding Capture point and a simplified zero-moment point heuristics (Carlo et al. (2018)). The essence of this heuristic is to step in the direction of motion over half of the expected distance traveled by the robot body to prevent falling. A general view can be found in 10. For gait planning, one can use a predetermined sequence of steps, switching them by timer (Carlo et al. (2018)) or event (Bledt et al. (2018)), or optimizing them according to possible points of contact and time duration per step (Winkler et al. (2018)). After analyzing the available open-source solutions, our team decided to take the developments of the MIT team, in particular Carlo et al. (2018); Bledt et al. (2017), and launch the Unitree A1 robot on complex surfaces. In the final solution, the operator can specify the linear and angular velocity with the joystick with respect to the displacement plane XY. However, the remaining two angles and the height of the body must be determined independently.

3.2 Calculating body vertical coordinate reference

We set desired body height with respect to the walking terrain in a body-fixed local coordinate frame located at its geometric center Ψ_r. As feet contact points change instantly, desired body height will be a step-constant function of time.

At first, by solving forward kinematics task using leg encoders measurements, we define feet locations for stance legs in Ψ_r $X_i = [x_i \ y_i \ z_i]^T$.

Local walking plane equation parameters are defined by best-fit approximation from the last recorded four feet contact points' coordinates. Leg sequence depends on a gait type. To define which legs are at the stance phase and time instance, when X_i should be calculated we can employ data from contact detection sensors located in each foot (more explanation later).Past traces must be also updated according to the displacement of the body-fixed frame.

Then, we can calculate normal to local walking plane

$$\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = (\phi^T \phi)^{-1} \phi^T \cdot b \tag{6}$$

where n_x, n_y, and n_z are coordinates of a normal to walking surface expressed in Ψ_r, $\phi = [X_1^T X_2^T X_3^T X_4^T]^T$, and $b = [1\ 1\ 1\ 1]^T$.

Finally, we can set reference for the body vertical coordinate from the desired body height h^* and normal coordinates expressed in Ψ_r. Since this reference should be defined in a world frame, we need to have an accurate localization of the robot body position, i.e. transformation from Ψ_r to the world frame.

To do so, we can use observer based on the Kalman filter Bloesch et al. (2013).

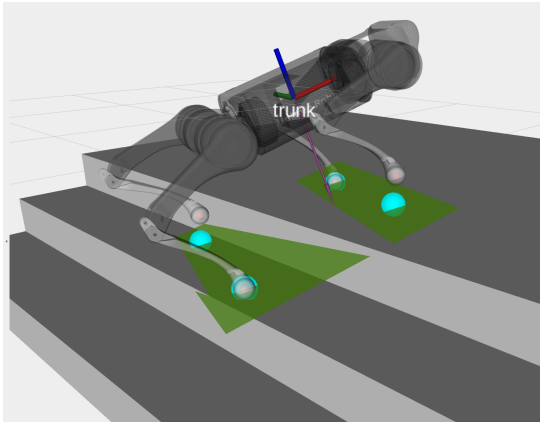

Fig. 3. Testing blind walking on stairs and visualizing in RVIZ. Visualization of the last four footprints (turquoise), the walking plane (green) and the height of the body relative to this plane (purple).

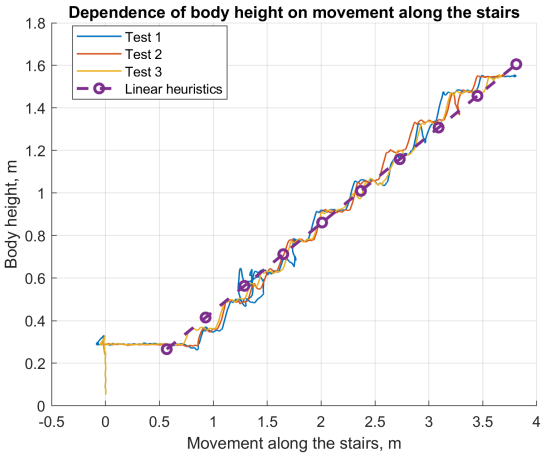

Fig. 4. Dependence of the body height relative to position on the stairs

Te verify our approach, we conducted several experiments in the Raisim simulation environment Hwangbo et al. (2018). Data on the absolute position of the robot during climbing the stairs were recorded. The graph of the dependence of the height of the robot's body on the robot's movement along the "X" axis is shown in the Fig. 4. The graph combines 3 test samples and the resulting heuristics, which is an averaged data approximated by linear function.

3.3 Pitch and roll references calculation

While the robot is climbing, we require its body to be in parallel to the stairs slope, and desired body pitch and roll angles should be calculated from this constraint. Because of vision sensors limitations, we decided to estimate it from leg encoders and the kinematic model similarly to local height estimation.

To maintain robot body in parallel to walking surface its desired pitch and roll angles should be calculated as

$$\psi^* = arccos\left(\frac{n_x}{\sqrt{n_x^2 + n_y^2 + n_z^2}}\right) - \frac{\pi}{2} \tag{7}$$

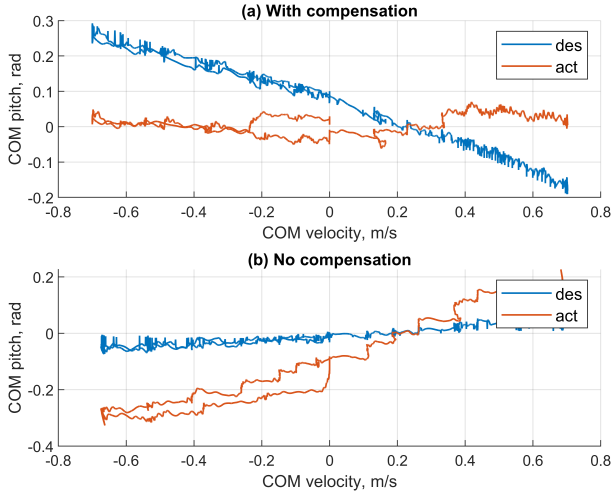

Fig. 5. Pitch current and desired angle (a) with and (b) without heuristics for the horizontal world x-direction movement.

and

$$\phi^* = -arccos\left(\frac{n_y}{\sqrt{n_x^2 + n_y^2 + n_z^2}}\right) + \frac{\pi}{2} \tag{8}$$

The proposed locomotion control approach gives a fairly accurate tracking at low speeds, but with relatively fast movement even on a horizontal surface, an undesirable deviation from reference pitch and roll angles appears Carlo et al. (2018). Therefore, we introduce correction relations:

$$\bar{\psi}^* = \psi^* + \tilde{\psi},$$

where $\tilde{\psi} = k_{vx} \cdot \dot{p}_{x,d} + k_{ox}$ and

$$\bar{\phi}^* = \phi^* + \tilde{\phi},$$

where $\tilde{\phi} = k_{vy} \cdot \dot{p}_{y,d} + k_{oy}$.

Coefficients k_{vx}, k_{ox}, k_{vy}, and k_{oy} can be estimated empirically similar to Bledt and Kim (2020). To find its values for our setup, we carried out experiments with linear movements on a flat horizontal surface at various constant speeds. In the first experiment, the desired velocity changed with a fixed time step along the X axis, while for the second one we changed velocity along the Y axis.

Both experiments show a linear dependence between angle of inclination and the desired speed, which makes it possible to calculate two coefficients with least squares method. We have got $k_{vx} = 0.072$, $k_{ox} = 0.047$, $k_{vy} = -0.33$, and $k_{oy} = 0.00027$.

The results of applying heuristic corrections can be seen in Fig. 5. For Fig. 5(a), when the correction is applied, the actual robot body pitch angle holds around zero, as expected, while Fig. 5(b) shows how the actual pitch angle drifts from the reference if there is no correction.

3.4 Contact detection

There are different approaches for contact detection. Work Camurri et al. (2017) presents an algorithm for estimating contact by analyzing internal torques in kinematic pairs. The method of probabilistic detection of contacts using a disturbance observer based on generalized torques and forces can be found in Bledt et al. (2018). Another way is to use direct measurements from the feet sensors. Our test-bed platform Unitree A1 robot is equipped with foot pressure sensors at the ends of the paws.

At first, this approach was tested within Raisim simulation environment Hwangbo et al. (2018). The software interface of the simulator allows to determine the contact of the desired object with the environment, and, if necessary, to obtain the magnitude of the force of a normal reaction upon contact. Here a constant threshold was used to binarize contact sensor feedback to mimic real feet pressure sensor behaviour.

We used this information to adapt robot gaits if early contact has been detected, i.e. stop further foot movement.

3.5 Gait scheduling

The robot's movement control is determined by a simple gait scheduler based on Carlo et al. (2018) that switches the controller between two phases – swing and stance. The current phase of movement is determined by the binary variable $s \in \{0 = swing, 1 = stance\}$. Having no contact feedback, the gait planner predetermined state s depending on the gait type and time. The type of gait is determined based on three values: the duration of one step cycle, the offset of the contact phase, and the duration of the contact phase. The offset indicates the time stamp from which the contact phase will begin. The duration determines how much of the step period the contact phase will occupy.

In case of early contact, it is suggested to move away from the planned gait pattern and immediately enter the stance phase by a signal from the contact sensor. The control actions for swing and stance are different. In the swing phase, impedance control is used to perform the trajectory tracking task. In the stance phase, the force obtained from the predictive controller is applied. If it is planned that the robot's leg is in swing, and indeed it has made contact, the controller will continue to make efforts to achieve the goal until the swing phase is completed. And in the opposite case, when it is planned that the leg is in stance, but it is actually in a swing, the controller will create the necessary control to create a reaction force, as a result of which the leg will come into contact without completing the desired trajectory. Both situations cause instability when moving through unstructured terrain with obstacles.

If an early contact occurs and if surface contact data is available, it is possible to switch from a time-interval-based scheduler to a contact-based scheduler. To do this, we introduce the value $\Delta p = \sigma_{gait} - \sigma_{feedback}$, where σ_{gait} is the scheduled stance offset, $\sigma_{feedback}$ is sensor-corrected offset. Thus, the modified stance offset σ' and stance duration ϕ' will be equal:

$$\begin{cases} \sigma' = \sigma_{gait} - \Delta p \\ \phi' = \phi_{gait} + \Delta p \end{cases} \tag{9}$$

Where ϕ_{gait} is the scheduled stance duration.
After modifying the parameters, the controller will switch to stance mode in accordance with the sensor on the leg.

To test the upgraded scheduler, the situation of an unexpected collision of the robot's foot with a solid surface was

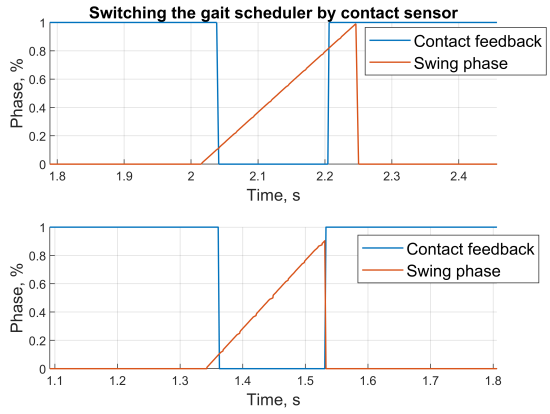

Fig. 6. Comparison of the basic gait planner and the modified one in case of an unexpected collision with an obstacle. Red line is the phase time in leg swing and blue is a sensor measurement.

modeled. Figure 6 shows the results of two experiments: with the original scheduler and our modification. The swing phase varies from 0 to 1.

The state that can be observed in Figure 6 a) is an early contact. The controller performs the swing phase, but the foot is in contact with the surface. On the graph, this is expressed in the intersection of the curves. Figure 6 b) shows an identical graph for the modified gait planner.

3.6 Building elevation map

This section will describe the methodology for constructing an elevation map and the algorithm for finding the most advantageous foothold in the constructed terrain map based on the orientation of the map segments.

Mobile robots are based on accurate mapping in real time using built-in sensors to provide autonomous navigation over rough terrain. Existing approaches often rely on absolute localization based on tracking external geometric or visual features Taketomi et al. (2017). To circumvent the reliability problems of these approaches, a method of terrain mapping based only on "blind" localization based on kinematic and inertial measurements is used Fankhauser et al. (2018). The proposed method considers drift, uncertainty of the condition assessment and the noise model of the distance sensor. It gives a probabilistic estimate of the terrain in the form of a grid-based elevation map. In terms of data collection, the map update method based on range measurements processes the range sensor data and updates all map cells with which new measurements are associated. As the robot moves, the uncertainty of the robot's movement extends to the map data to match the formulation of centering the map on the robot. This is done when updating by robot movement for all cells on the map. These two processing steps are used to create a robot-oriented elevation map in real time. Since computational costs increase with increasing map size, the size is limited by the local area around the robot. The map can be represented by several layers of information. The main layer contains the height of terrain sections, other layers may contain additional information, such as surface normals, variance, timestamp, and others.

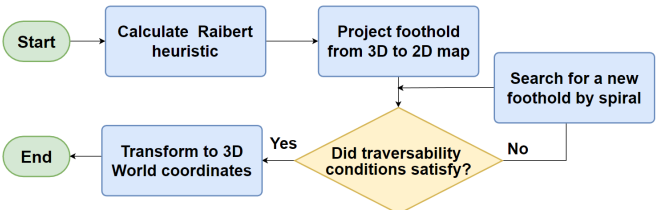

Fig. 7. Algorithm for finding the desired foothold based on elevation map

3.7 Adapting desired foothold

The desired trajectory of the robot's leg movement in the swing phase is constructed from three points using Bez'ier curves in the world coordinate system Carlo et al. (2018). The root points of the trajectory are the starting point p_0; the highest point of the trajectory h_{swing}; the final point of the trajectory or the desired foothold p_f.

The foothold, according to the article Carlo et al. (2018), is determined by the Raibert heuristic:

$$p_f = p_0 + \frac{\Delta t}{2} \cdot U + k(U - U_{des}) \tag{10}$$

where Δt is the stance duration;U – current body speed;U_{des} – desired body speed;k – feedback gain.

Using this dependence, it is possible to find the position of the foothold only in the plane of motion, X-Y, the height of the point cannot be determined. To calculate the component of the height proposed, an elevation map is used.

Before finding a foothold , it is necessary to classify the cells of the map according to the type of traversability. The following method is used as a criterion for assigning a traversability level to a map cell. The normal vector is calculated for each cell. After, the projection of the normal vector onto the vertical Z axis is computed. If the projection is less than threshold value ($threshold = 0.98$), then the cell is marked unsteppable.

A flowchart describing the foothold correction algorithm is shown in Figure 7. First, the reference value of the foothold is calculated according to the Raibert heuristic. Then it is determined to which cell of the elevation map this point belongs. Having determined the cell, the cell is checked for traversability . If the condition is satisfied, then a recalculation is performed from the coordinates of the map cell to the world Cartesian coordinate system, otherwise a search is launched among the nearest cells along a spiral of a limited radius. Finally, the found cell on the map is converted to the world coordinate system, and the z coordinate is set equal to the value of the cell.

4. CONTROLLER PERFORMANCE EXPERIMENTAL VERIFICATION

The designed controller was tested and debugged on the Unitree A1 robot. After testing in the simulator, the transfer to hardware did not create difficulties. It was necessary to change the joints gains in control loop. There was also an attempt to launch the controller on the Unitree Go1 robot, which is similar in kinematic structure, but has different engine parameters and mass-

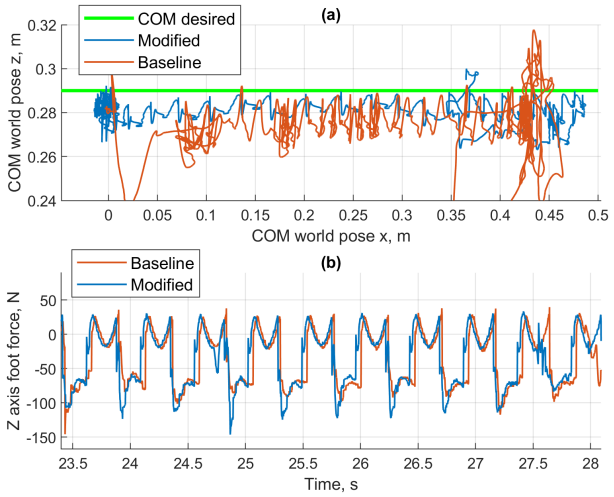

Fig. 8. Comparison of two controllers in (a) COM height relative to the world frame x position and (b) z-projection foot force from real robot.

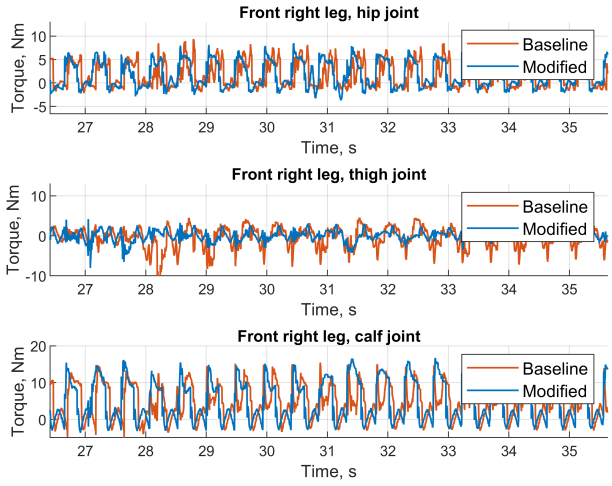

Fig. 9. Generalized torques of the joints of the right front leg

dimensional characteristics. Despite some difference in the characteristics of the motors, it showed that MPC-based algorithms can compensate for the difference in equipment.

4.1 Blind-walking experiment

Our proposed approach with early contact detection, local body height evaluation and pitch angle corrections helps quadruped robot to overcome challenging terrains with no perception. Experiments [1] were conducted to overcome stairs with no perception and a step height of up to 15 cm both in the RaiSim simulator Hwangbo et al. (2018) and on a real Unitree A1 robot.

4.2 Comparison with baseline

To compare our solution with baseline, a test scenario was developed. The robot in the trot gait started moving forward, then stood on the step with its right front paw, stopped in this position and stomped in place for about 5 seconds. This scenario allows us to clearly see the improvements in our approach. Processing of early contact and heuristic correction of the orientation of the body of the robot allowed to reduce the error by state.

Figure 8 shows how the controller handles the task of holding the COM position when the feet are at different heights.

Figure 9 shows the generalized torques for one leg that collided with an obstacle.Our modification shows a decrease in the value of the generalized torques, especially in the hip joint.

We will also attach the recorded log files [2] during test runs so that anyone can look at them, and in the future they will be useful for comparison and analysis. The files are recorded in rosbag format, which is easy to use inside the ROS ecosystem.

[1] https://www.youtube.com/watch?v=9zfd1r-q8ds
[2] http://bit.ly/3X0LppY

5. CONCLUSION AND FUTURE WORK

In our work[1], we verified the approach to controlling a walking robot using convex MPC controller with combination of WBIC. We tested and described the main features of transferring the solution to real hardware with limited computing power. The basic solution was not enough to overcome the stairs, so a number of modifications were carried out, such as taking into account the early contact of the legs, assessing the local plane under the robot by contact points, adjusting the angle of inclination of the body when moving on an uneven surface, heuristic estimation of the global height when overcoming an obstacle of known geometry, the use of vision to adjust the desired foothold location.

Although this solution can work successfully on the stairs, there are many unresolved critical issues. The robot does not consider its own state when planning the desired body and leg trajectories, nor does it check whether it can complete the task, whether there will be body crossings, or how well it has completed it. In terms of setting regulator coefficients and heuristics, there is no general algorithm, only empirical observations. The robot's camera doesn't quite look under the legs, which inhibits it from mapping the stairs. Our work will focus on fixing these shortcomings first to make the use of walking robots widespread.

REFERENCES

Afsari, K., Halder, S., Ensafi, M., DeVito, S., and Serdakowski, J. (2021). Fundamentals and prospects of four-legged robot application in construction progress monitoring. *EPiC Series in Built Environment*, 2, 274–283.

Bledt, G. and Kim, S. (2020). Extracting legged locomotion heuristics with regularized predictive control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 406–412. doi: 10.1109/ICRA40945.2020.9197488.

Bledt, G., Wensing, P.M., Ingersoll, S., and Kim, S. (2018). Contact model fusion for event-based locomotion in unstructured terrains. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 4399–4406. doi:10.1109/ICRA.2018.8460904.

Bledt, G., Wensing, P.M., and Kim, S. (2017). Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the mit cheetah. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4102–4109. doi: 10.1109/IROS.2017.8206268.

Bloesch, M., Hutter, M., Hoepflinger, M.A., Leutenegger, S., Gehring, C., Remy, C.D., and Siegwart, R. (2013). State estimation for legged robots-consistent fusion of leg kinematics and imu. *Robotics*, 17, 17–24.

Camurri, M., Fallon, M., Bazeille, S., Radulescu, A., Barasuol, V., Caldwell, D.G., and Semini, C. (2017). Probabilistic contact estimation and impact detection for state estimation of quadruped robots. *IEEE Robotics and Automation Letters*, 2(2), 1023–1030. doi: 10.1109/LRA.2017.2652491.

Carlo, J., Wensing, P., Katz, B., Bledt, G., and Kim, S. (2018). Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. 1–9. doi: 10.1109/IROS.2018.8594448.

Carlo, J. (2020). *Software and control design for the MIT Cheetah quadruped robots*. Ph.D. thesis.

Chignoli, M. and Wensing, P.M. (2020). Variational-based optimal control of underactuated balancing for dynamic quadrupeds. *IEEE Access*, 8, 49785–49797. doi: 10.1109/ACCESS.2020.2980446.

Fankhauser, P., Bloesch, M., and Hutter, M. (2018). Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robotics and Automation Letters*, 3(4), 3019–3026. doi:10.1109/LRA.2018.2849506.

Hutter, M. (2013). *StarlETH & Co.: Design and control of legged robots with compliant actuation*. Ph.D. thesis, ETH Zurich.

Hutter, M., Gehring, C., Jud, D., Lauber, A., Bellicoso, C.D., Tsounis, V., Hwangbo, J., Bodie, K., Fankhauser, P., Bloesch, M., et al. (2016). Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 38–44. IEEE.

Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., and Hutter, M. (2019). Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), eaau5872.

Hwangbo, J., Lee, J., and Hutter, M. (2018). Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2), 895–902. doi: 10.1109/LRA.2018.2792536. URL `www.raisim.com`.

Kolvenbach, H., Wisth, D., Buchanan, R., Valsecchi, G., Grandia, R., Fallon, M., and Hutter, M. (2020). Towards autonomous inspection of concrete deterioration in sewers with legged robots. *Journal of field robotics*, 37(8), 1314–1327.

Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D.G., and Semini, C. (2020). Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping, and whole-body control. *IEEE Transactions on Robotics*, 36(6), 1635–1648. doi:10.1109/TRO.2020.3003464.

Raibert, M.H. (1986). *Legged robots that balance*. MIT press.

Rudin, N., Hoeller, D., Bjelonic, M., and Hutter, M. (2022). Advanced skills by learning locomotion and local navigation end-to-end. *arXiv preprint arXiv:2209.12827*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, 1889–1897. PMLR.

Taketomi, T., Uchiyama, H., and Ikeda, S. (2017). Visual slam algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1), 16. doi:10.1186/s41074-017-0027-2. URL `https://doi.org/10.1186/s41074-017-0027-2`.

Winkler, A.W., Bellicoso, C.D., Hutter, M., and Buchli, J. (2018). Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3), 1560–1567.