# Vision Aided Dynamic Exploration of Unstructured Terrain with a Small-Scale Quadruped Robot

D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim

***Abstract*— Legged robots have been highlighted as promising mobile platforms for disaster response and rescue scenarios because of their rough terrain locomotion capability. In cluttered environments, small robots are desirable as they can maneuver through small gaps, narrow paths, or tunnels. However small robots have their own set of difficulties such as limited space for sensors, limited obstacle clearance, and scaled-down walking speed. In this paper, we extensively address these difficulties via effective sensor integration and exploitation of dynamic locomotion and jumping. We integrate two Intel RealSense sensors into the MIT Mini-Cheetah, a $0.3$ m tall, $9$ kg quadruped robot. Simple and effective filtering and evaluation algorithms are used for foothold adjustment and obstacle avoidance. We showcase the exploration of highly irregular terrain using dynamic trotting and jumping with the small-scale, fully sensorized Mini-Cheetah quadruped robot.**

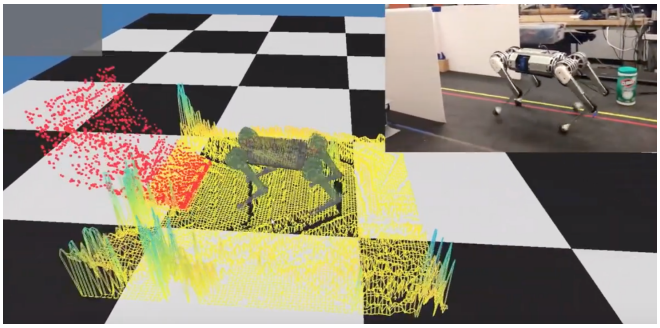


Fig. 1. **Mini-Cheetah integrated with vision sensors.** Mini-Cheetah is a small-size quadruped robot fully functioning with vision system.

## I. Introduction

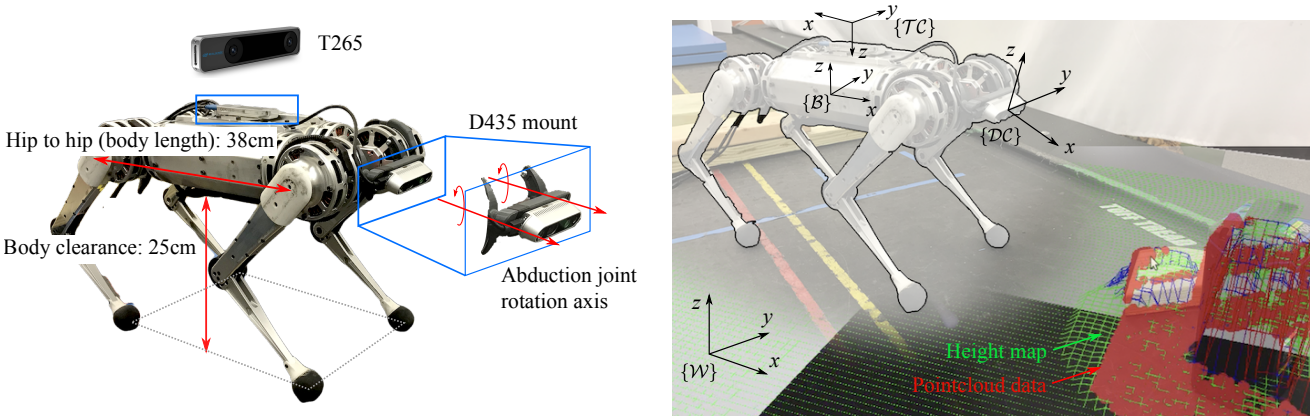
Legged robots are well suited for rough terrain locomotion due to the fact that limb utilization provides much larger coverage over irregular terrain than wheeled or tracked mobile platforms. This has motivated research on disaster response using legged systems [1]–[3]. Quadruped robots in particular have been highlighted as promising candidates for locomotion over challenging terrain [4]–[7] and robust, dynamic locomotion has been demonstrated on various systems. Existing quadrupeds share a common factor in that they are large enough to clear normal obstacles such as stairs or curbs. However, smaller robots are desirable for the exploration of spaces with narrow regions such as tunnels, damaged buildings, ships, or submarines. However, except for a few studies using the Boston Dynamics LittleDog robot [8]–[10], there is no active research on small-scale quadruped robots. Furthermore, quadruped robots with fully integrated perception capabilities are particularly rare.

Small robots have several fundamental limitations which demotivate their study. First of all, small robots do not have enough space to integrate comprehensive sensor suites. As such, the Mini-Cheetah also does not provide significant space for additional sensors. Therefore, instead of a large lidar sensor, we elected to use two small Intel RealSense cameras, the T265 and D435 for localization and depth perception respectively.

Installing a depth camera presented a unique challenge because the head and tail of the robot are where the hip and knee joint actuators connect to the abduction actuator. Therefore, the Mini-Cheetah has no stationary parts at its head to which we could attach a depth camera. We resolved this issue by making an attachment with two pivots designed to align with the abduction joint axes. We connected the attachment to the hip joint actuators which move as the abduction joints rotate, however the sensor remains stationary throughout the abduction because the pivot points are aligned with the rotation axes (see Fig. 2).

The space limitation effects not only sensor placement but also computation power by limiting computer and battery size. Running the perception processes in parallel with real-time feedback control is challenging given the lowered computational capacity. Therefore, we utilize simple and efficient algorithms for obstacle avoidance and vision data processing. For obstacle avoidance, we utilize a potential field algorithm, which makes virtual potential fields which generate virtual repulsive forces when the terrain height is over a threshold. The virtual forces are summed with the commanded path vector to compute the final velocity command. As a result, the robot can smoothly avoid obstacles without need for explicit low level path planning or prior knowledge of the terrain. For heightmap construction and filtering, we use an opening morphological transformation implemented using OpenCV. Gradient-based heightmap evaluation is used to avoid stepping on edges in the terrain. By optimization of the software implementation with efficient memory handling and data down-sampling, we succeeded in running all processes including locomotion control on a small Intel UP board computer in the Mini-Cheetah. We could not use the standalone setup in live experiments because of cable clearance issues with the UP board USB 3.0 port.

In addition to the technical challenges, small robots have limited capability in locomotion speed and obstacle-clearance caused by their short limb length. However, these limitations can be overcome by exploiting faster, more dynamic gaits and by jumping over obstacles. In this paper, we demonstrate rapid exploration over highly irregular terrain




(a) System specification

(b) Coordinate system and vision data

Fig. 2. **System description and coordinate system.** (a) The Mini-Cheetah robot is fully sensorized with two Intel RealSense sensors. (b) The coordinate systems are described. Each sensor signal is eventually transformed into the world reference frame to build a global heightmap.

using a trotting gait. Unlike other locomotion methods that are limited by the speed restrictions imposed by vision integration [11]–[13], our method aims to maintain dynamic locomotion capability. In addition, we also demonstrate jumping over large terrain discontinuities to further improve terrain coverage.

In summary, the major contributions of this paper are twofold: 1) systematic integration of vision sensors, perception algorithms, obstacle avoidance, locomotion control, and jump motion optimization to accomplish dynamic exploration with a small-scale quadruped robot, and 2) experimental validation using the sensorized Mini-Cheetah robot. We accomplished 0.38 m/s (or 1 body length per second) locomotion over the highly irregular terrain.

## II. System Overview

The platform used in this study is the MIT Mini-Cheetah [14]. The Mini-Cheetah is a 9 kg, 0.3 meter tall electrically actuated small-scale quadruped robot. The upper and lower leg links are 0.21 meters and 0.21 meters respectively and there are 0.38 meters between the front and hind legs as shown in Fig. 2(a). The robot stands with 25 cm ground clearance which is comparable to the height of a stair or street curb. This low clearance limits the height of obstacles that the robot can overcome under normal walking conditions.

Locomotion is executed on an Intel UP board low-power single board computer with a quad-core Intel Atom CPU, 4 GB RAM and a 1.92 GHz, running Linux with the CONFIG-PREEMPT-RT patch for pseudo-realtime operation. The UP board is the size of a credit card and has computational power comparable to that of a tablet computer.

### A. Intel RealSense

Visual data is gathered using two Intel RealSense cameras, the D435 and T265. The D435 provides depth images used to construct a map of the area surrounding the robot while the T265 serves for localization.

The RealSense D435 is $90\ \text{mm} \times 25\ \text{mm} \times 25\ \text{mm}$, has a mass of 71.8 g, and an $87° \times 58° \times 95°$ field of view (FOV). The sensor publishes $640 \times 480$ depth images at a rate of 90 Hz in a pointcloud format. The D435 has an accuracy of less than 1% error per meter distance from the sensor with a minimum distance of 10 cm. For our application where most of the sensor returns are less than a meter away, the expected accuracy is between 2.5 mm and 5 mm.

The Realsense T265 is $108\ \text{mm} \times 24.5\ \text{mm} \times 12.5\ \text{mm}$, has a mass of 55 g, and has two fisheye lenses with a combined $163 \pm 5°$ field of view (FOV). The sensor has an onboard inertial measurement unit (IMU) which allows for more accurate rotation and acceleration measurements. Finally, the RealSense T265 has an integrated Intel Movidius Myriad 2.0 Visual Processing Unit (VPU) which is highly optimized to run visual-inertial simultaneous localization and mapping (SLAM). The T265 publishes a pose estimate produced by the visual-inertial SLAM algorithm at a rate of 200 Hz which was used by the robot for localizing within the world reference frame.

The RealSense's small size, wide field of view, high frame rate, and global shutter make it a great option for mobile and highly dynamic robotics applications. Lidar sensors were also considered for use in this experiment but were ruled out early on due to their size which was roughly 1/3 of the Mini-Cheetah's body size. The power consumption and localization computation power required for lidar would have been too demanding for the battery and computer currently on-board the Mini-Cheetah. Moreover, the high cost, gyroscopic forces, pointcloud sparsity, low frame rate, and high minimum range (approximately 1 m) of lidar were also significant factors in our sensor selection.

### B. Framework Overview

Our framework consists three primary components: Vision processing, locomotion control, and a motion library (Fig. 3). Vision processing finds the robot's absolute pose and builds a heightmap surrounding the robot. The vision outputs are

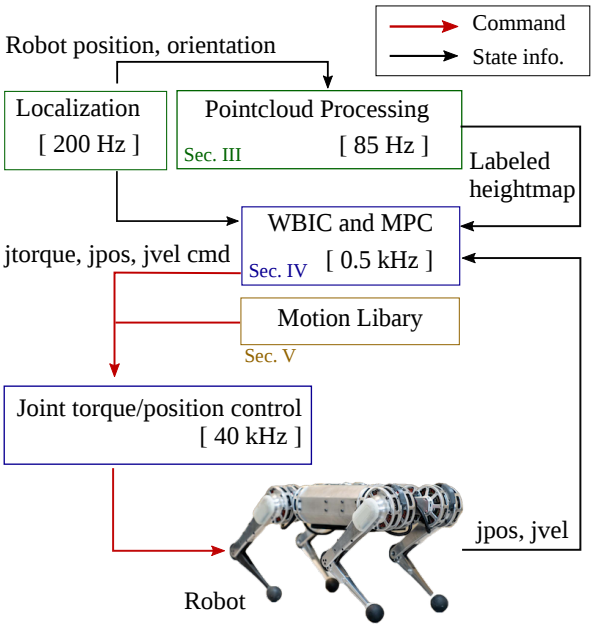



Fig. 3. **Software Framework.** There are three update loops in our software setup.

sent to the locomotion controller through Lightweight Communication and Marshaling (LCM) [15]. The locomotion controller uses the visual data to adjust the direction of walking and step locations. The motion library saves pre-planned motions such as jumping and a recovery-stand-up protocol. Once a predefined motion is called, the saved joint position and torque trajectories are sent to the actuator controllers to execute the commanded behavior.

Running the vision process in parallel with the real-time control presents a challenge due to the high computational demands of both systems. By optimizing the computations with efficient memory handling, down sampling, and by developing a compact and efficient custom dynamics engine, we succeeded to run all computations including both the vision process and locomotion control in the Mini-Cheetah's small UP board computer. Unfortunately, we could not use the standalone setup in the experiments because of difficulties accessing a USB 3.0 port. The depth camera's pointcloud data streaming rate varies significantly depending on communication protocol; 90 Hz on USB 3.0 but only 10 Hz on USB 2.0. The UP board computer has a single USB 3 port but the location of the connector is orthogonal to four USB 2.0 ports. The current Mini-Cheetah setup requires that one of the computer's sides faces a wall, and so we must forgo use of either the USB 3.0 port or the four USB 2.0 ports. In the experiment, the localization sensor, T265, is connected to the UP board computer, but the depth camera, D435, is wired to the external desktop. In the upcoming hardware revision, we will resolve this issue and demonstrate the vision-aided locomotion without tether.

## III. Vision Processing

The vision processing outputs the robot's global position/orientation and a labeled heightmap. The position/orientation process is solely done by Realsense T265 and the localization information is sent to both the heightmap construction program and the real-time locomotion controller. The robot's absolute posture is used to transform the pointcloud data into the global frame in the heightmap construction process. The details of the computation are given in the following sections.

### A. Heightmap Construction

The pointclouds published by the RealSense camera are in the reference frame of the camera. In order to be able to plan a path or footsteps with this data, they need to be transformed into the world reference frame. This is done by first calculating a transformation from the world frame to the camera frame by combining transformations from world to localization camera, localization camera to robot's body center, and robot's body center to pointcloud camera then using the resulting transformation matrix to transform the pointcloud data into the world reference frame. This calculation is shown below in equation (1)

$$\mathbf{T}_{DC}^{\mathcal{W}} = \mathbf{T}_{TC}^{\mathcal{W}}\,\mathbf{T}_{\mathcal{B}}^{TC}\,\mathbf{T}_{DC}^{\mathcal{B}} \tag{1}$$

where $\mathcal{W}$, $\mathcal{DC}$, $\mathcal{TC}$, and $\mathcal{B}$ represent the world, depth camera, tracking camera, and robot's body center respectively, the superscript represents the reference frame and the subscript represents the point. For example, $\mathbf{T}_{\mathcal{DC}}^{\mathcal{W}}$ represents the transformation from the origin of the world reference frame to the depth camera. These coordinate systems can be seen in Fig. 2(b).

$\mathbf{T}$ represents a transformation matrix which is a member of the Special Euclidean group, SE(3), which is the set of all transformations that can be applied to a rigid body. Each matrix $\mathbf{T}$ is composed of a rotation matrix, $\mathbf{R}$ and a translation vector $\mathbf{t}$ as follows in equation (2):

$$\mathbf{T}_{\mathcal{DC}}^{\mathcal{W}} = \begin{bmatrix} \mathbf{R}_{\mathcal{DC}}^{\mathcal{W}} & \mathbf{t}_{\mathcal{DC}}^{\mathcal{W}} \\ \mathbf{0}_3^\top & \mathbf{1} \end{bmatrix} \tag{2}$$

We use $\mathbf{0}_3$ to represent a three dimensional zero vector and $\mathbf{0}_n$ to represent a $n$-dimensional zero vector in this paper.

The complete transformation is then

$$\mathbf{P}^{\mathcal{W}} = \mathbf{T}_{\mathcal{DC}}^{\mathcal{W}}\mathbf{P}^{\mathcal{DC}} \tag{3}$$

The world-frame pointcloud is then used to update a 2.5 dimension heightmap where the half-dimension refers to the fact that the third dimension is encoded as the value in an matrix of the other two dimensions. More specifically, the heightmap is a representation of the world in which the X and Y dimensions are discretized into $1.5\ \mathrm{cm} \times 1.5\ \mathrm{cm}$ cells. For each pointcloud received, the value of a given cell in the heightmap, representing the height at that location, is calculated by taking the Z component of the most recent pointcloud point in the corresponding X-Y cell range.

For the purposes of this experiment, the full world map was only $15\ \mathrm{m} \times 15\ \mathrm{m}$ discretized into a 1000 cell $\times$ 1000 cell grid. In order to increase the publishing rate, only a 100 cell $\times$ 100 cell (or $1.5\ \mathrm{m} \times 1.5\ \mathrm{m}$) region centered about the robot's current location is published at a given point in time.

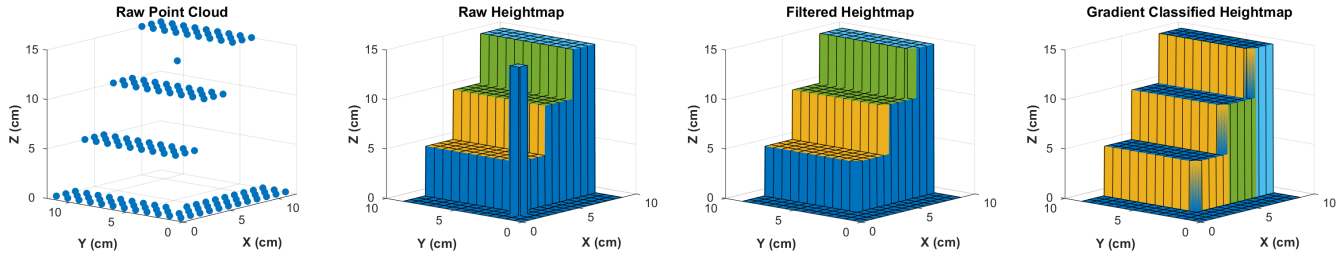



Fig. 4. **An overview of the filtering process.** *From left to right;* an example 3D pointcloud representing a set of stairs with a single noisy return. The returns are binned in X and Y to form a 2.5D heightmap. The heightmap is then filtered with an opening morphological transformation. Finally, the gradient is computed of the filtered heightmap and the value of the gradient is used to segment the terrain by traversability. In the above rightmost image, dark blue represents STEPPABLE, yellow represents UNSTEPPABLE, green represents JUMP ONLY, and light blue represents IMPASSABLE. Note that these colors have no meaning in the center two images and are only used for ease of visualization.

### B. Filtering And Traversability Evaluation

A spatial filter was used to fill in sparse regions of the pointcloud and to filter out extraneous sensor returns. The filter consists of an erosion operation followed immediately by a dilation operation. Both erosion and dilation are computed by convolution of the heightmap with a 5 cell $\times$ 5 cell elliptical kernel shown below:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4}$$

The center point of the kernel is referred to as the anchor point. In the convolution, the kernel is scanned over the image. In the case of erosion, the value of the output cell corresponding to the location of the anchor point becomes the minimum value of the original image overlapped by the kernel. In the case of dilation, the value of the output cell corresponding to the location of the anchor point becomes the maximum value of the original image overlapped by the kernel. The dilation operation serves primarily to fill in sparsely sampled regions while the erosion operation serves to eliminate extraneous returns [16].

The filtered heightmap is then used to compute a traversability map. The traversability map has the same grid cell structure as the heightmap but only four possible cell values. The set of possible cell values are STEPPABLE, UNSTEPPABLE, JUMP-ONLY, and IMPASSABLE.

A determination of STEPPABLE means that a foot can be placed in that grid location; UNSTEPPABLE means that a foot cannot be placed in that grid location but it can be stepped over; JUMP-ONLY means that it can only be passed by jumping; and IMPASSABLE means that there is no way to pass that particular grid location.

The class of each cell in the grid is determined by a two step process. The first step is to generate a gradient map with the Sobel gradient operator [16]. The Sobel gradient operator convolves two $3 \times 3$ kernels with the original image to approximate the discrete derivative in X and Y at each location in the image. For an image $\mathbf{I}$, the gradient in X, $\mathbf{G_X}$, and in Y, $\mathbf{G_Y}$, are computed as follows in equations (5) and (6):

$$\mathbf{G_X} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{I} \tag{5}$$

$$\mathbf{G_Y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{I} \tag{6}$$

where $*$ represents two-dimensional convolution.

Once the discrete derivatives in X and Y are computed, the gradient map takes maximum value in each cell. The gradient map is then thresholded into the four different categories.

## IV. Locomotion Control

For the walking motion control, our group's hybrid control scheme presented in [17] is used. The control scheme consists of two controllers, model predictive control (MPC) and whole-body control (WBC). The combined controller provides outstanding stability and robustness, which allow us to accomplish 3.7 $\mathrm{m/s}$ forward running with the Mini-Cheetah. An almost identical controller is used in this paper with two additional features; obstacle avoidance and foothold adjustment based on visual information.

### A. Obstacle Avoidance

To avoid obstacles, we utilize a potential field algorithm. The presented algorithm is useful to let the robot find an obstacle free path subject to the given walking direction command. Therefore, we assume that there is high-level direction command which may be informed by a global map while allowing the robot to overcome smaller scale irregularities autonomously.

At every iteration of the control loop, if we identify an object with an impassable height, we add a radial basis function (RBF) at the location of the object. To avoid multiple potential fields overlapping on the same object, we do not allow additional RBFs to be generated near existing ones. In the final velocity command computation, the function's gradient is added to the desired path command. This simple method works effectively in a real system without any additional path finding or prior knowledge about the environment.

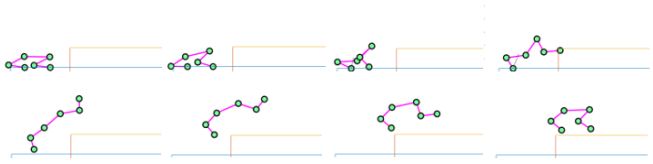


Fig. 5. **Jumping motion.** The joint trajectories and torques are computed by offline optimization, and the robot perform the jump by executing the joint commands.

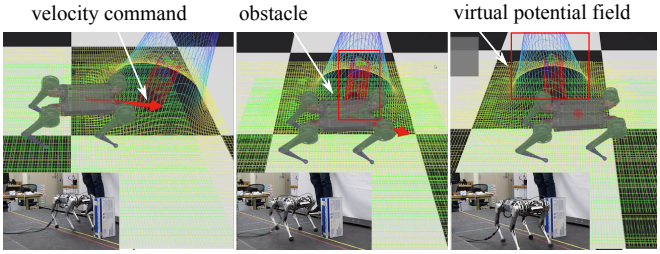



Fig. 6. **Obstacle avoidance.** Mini-Cheetah navigates around an obstacle because of the repulsive force generated by the virtual potential field.

### B. Footstep Planning

In the footstep planning, we first compute the default step location with the following equation:

$$\mathbf{p}_{\text{foot},i} = \mathbf{p}_{\text{shoulder},i}, \tag{7}$$

where,

$$\mathbf{p}_{\text{shoulder},i} = \mathbf{p}_{\text{body}} + \boldsymbol{R}_z\left(\psi\right)\mathbf{l}_i, \tag{8}$$

$$\mathbf{p}_{\text{symmetry}} = \frac{t_{\text{stance}}}{2}\mathbf{v} + k\left(\mathbf{v} - \mathbf{v}^{\text{cmd}}\right). \tag{9}$$

In Eq. (8), $\mathbf{p}_{\text{body}}$ is the body position and $\mathbf{l}_i$ is $i$-th leg shoulder location with respect to the body's local frame. Since $\boldsymbol{R}_z\left(\psi_k\right)$ is rotation around yaw direction with the current body yaw angle, $\mathbf{p}_{\text{shoulder},i}$ is the $i$-th shoulder location with respect to the global frame. $\mathbf{p}_{\text{symmetry}}$ is a Raibert heuristic [18] that makes the leg's landing angle and leaving angle be identical if the robot's velocity is the same as the commanded velocity. In our setup, we use 0.03 for the feedback gain, $k$.

Once the step location is calculated, we then check the traversability of the selected location. If our first selection is not STEPPABLE, we then search the neighboring grid locations for the nearest location that is STEPPABLE. The search is simply done by iteratively checking the next cell over, spiraling out from the initial location. Once we find a proper place to step as determined by the traversability map, we read the height of that point and adjust the swing and footstep landing height based on the height of that step location.

## V. Jump Motion

In addition to the walking controller, we prepared jumping motion for scenarios in which the robot encounters an obstacle that is too high to step over. The jump controller was obtained in the same way the backflip controller [14] was created. The motion for the jump was generated using 2-D non-linear optimization of a 5-link robot. CassADi [19], an open-source optimization library was used to execute the non-linear optimization. The optimization generates a set of joint states and torques for each time-step of the trajectory.

The contact states of the feet of the robot (4 legs in contact with the ground, 2 legs in contact with the ground, 0 legs in contact with the ground - flight) were pre-defined and used as a scheduler. The jump was then generated by minimising a cost function and constraining the joint states and torques to certain conditions during trajectory. The cost function was defined to be the error between the desired final joint state of the robot and the joint state at each timestep while still being subject to trajectory constraints. The selection of this cost function minimizes excessive leg swinging during flight which removes undesirable behaviours such as legs colliding with each other at any time.

The explicit constraints set were the initial and final positions of the robot (CoM), maximum joint torques, minimal normal force for feet in contact (to prevent slippage) and zero forces on feet that are not in contact with the ground. Other constrains and limits, which were implicit, arise from actuator dynamics and robot dynamics. The purpose of these constraints was to ensure that the robot achieved the desired goal location. Fig. 5 shows a sequence of frames representing an optimal jump trajectory found using this method.

When run on hardware, the controller was switched to a landing controller towards the end of the flight. This was achieved using joint PD control on a wider stance for stability.

## VI. Results

In order to test our proposed method, we designed the experimental environment to be a gauntlet of challenges including cluttered terrain, obstacles, and stacked panels as shown in Fig. 6 and Fig. 7. We first accomplished each task repeatably in isolation before finally overcoming all obstacles in a single run. The Mini-Cheetah was able to successfully track a specified path, see and build a map of its environment, plan and execute safe footsteps on and over obstacles up $10\ \text{cm}$ tall, recover independently if it had fallen, identify obstacles which it could not traverse and modify its path to avoid them, and finally jump up onto a $13\ \text{cm}$ platform.

### A. Obstacle Avoidance

In the test presented in Fig. 6, the robot moves back and forth by following the given trajectory, and once it detects an object, Mini-Cheetah goes around the obstacle autonomously. When we move the obstacle along the given path, the robot is able to update its model of the environment and navigate accordingly (see Fig. 6(b)).

### B. Rough Terrain Locomotion and Jump

In the test presented in Fig. 7, we made cluttered terrain with wood blocks to verify the locomotion capability of Mini-Cheetah over irregular terrain. The experiment consists of two parts, rough terrain locomotion and jumping onto

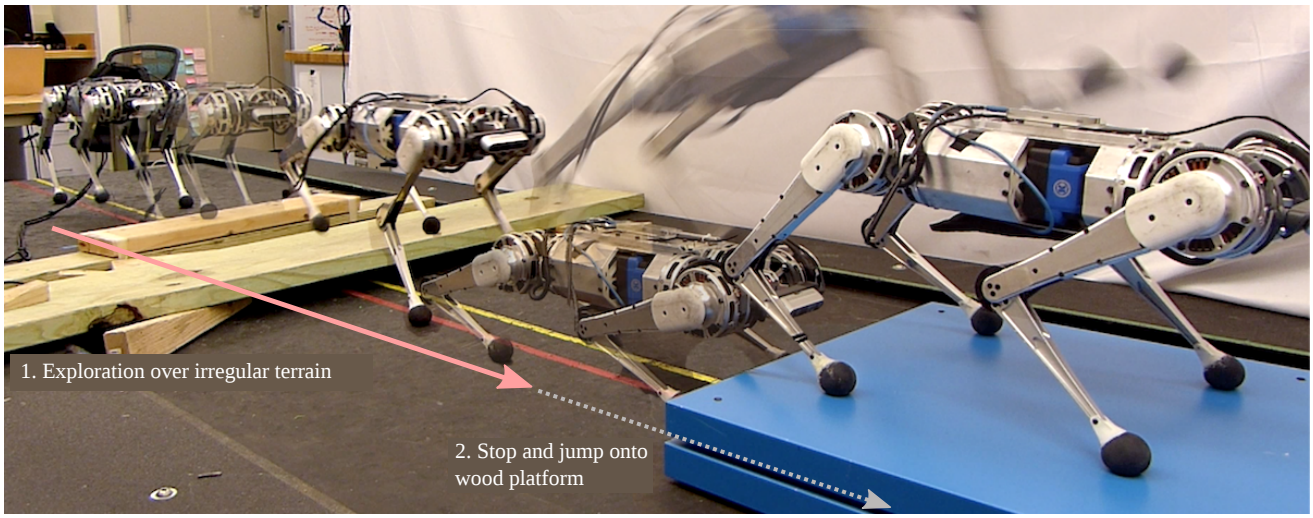



Fig. 7. **Experiment result overview.** Our experimental setup is designed to show the exploration capability of the Mini-Cheetah robot through rough terrain locomotion requiring a jump.

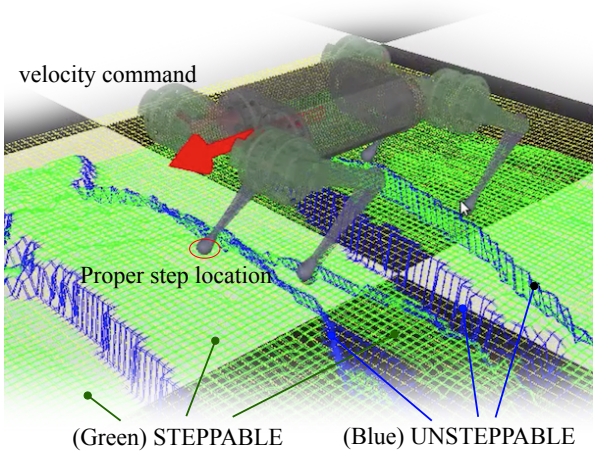



Fig. 8. **Walking over rough terrain.** The heightmap colors indicate the traversability determination of the terrain. The step planner adjust footholds based on the labelled heightmap.

a platform. The Mini-Cheetah followed a straight path at $0.33\ \mathrm{m/s}$ over obstacles up to $10\ \mathrm{cm}$ (nearly half of the robot's ground clearance). Fig. 8 shows how the controller perceived the terrain. After traveling $2.3\ \mathrm{m}$ along this path, the jump maneuver was manually engaged. The robot landed more than a body length forward and with all four feet on top of the $13\ \mathrm{cm}$ platform.

## VII. Conclusion and Future Work

In this paper, we propose a system of modules and a corresponding integration framework which allow for dynamic, rapid exploration of unknown and unstructured environments with a compact, lightweight, and robust robotic platform. These low level autonomy modules and perception integration will bridge the gap between the hardware and control progress to date and the higher level autonomy work to come. As we distribute the Mini-Cheetah platform to collaborators, we hope that this progress will catalyze further developments in controls, perception, and autonomy for highly dynamic legged systems. In particular, hope to see the Mini-Cheetah manipulate its environment, navigate rough terrain upwards of $3\ \mathrm{m/s}$, and develop a more nuanced understanding of the surrounding objects in its environment.

For the time being, we have assumed that all items detected by the depth camera are rigid objects. However, this assumption is not valid when the robot encounters grass, mud, or other soft materials. In the future, we plan to utilize visual recognition algorithms to determine an obstacle's rigidity as well as its dimensions and location.

Another important update is the hardware platform embedding of the vision system. The next generation Mini-Cheetah platform will include four Intel RealSense cameras in its body as well as a dedicated on-board vision computer. The entire system will be powered by an on-board battery, so the new Mini-Cheetah will be a fully sensorized and untethered system.

## Acknowledgments

This work was supported by the Toyota Research Institute, Centers for ME Research and Education at MIT and SUSTech, Naver Labs, and the Air Force Office of Scientific Research.

## References

[1] G. Pratt and J. Manzo, "The darpa robotics challenge [competitions]," *IEEE Robotics & Automation Magazine*, vol. 20, no. 2, pp. 10–12, 2013.

[2] E. Krotkov, D. Hackett, L. Jackel, M. Perschbacher, J. Pippine, J. Strauss, G. Pratt, and C. Orlowski, "The darpa robotics challenge finals: Results and perspectives," *Journal of Field Robotics*, vol. 34, no. 2, pp. 229–240, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21683

[3] C. D. Bellicoso, M. Bjelonic, L. Wellhausen, K. Holtmann, F. Gnther, M. Tranzatto, P. Fankhauser, and M. Hutter, "Advances in real-world applications for legged robots," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1311–1326, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21839

[4] M. Bajracharya, J. Ma, M. Malchano, A. Perkins, A. A. Rizzi, and L. Matthies, "High fidelity day/night stereo mapping with vegetation and negative obstacle detection for vision-in-the-loop walking," *Intelligent Robots and Systems (IROS)*, pp. 3663–3670, Nov. 2013.

[5] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10 822 – 10 825, 2008, 17th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016407020

[6] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of hyq a hydraulically and electrically actuated quadruped robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011. [Online]. Available: https://doi.org/10.1177/0959651811402275

[7] M. Bajracharya, J. Ma, M. Malchano, A. Perkins, A. A. Rizzi, and L. Matthies, "High fidelity day/night stereo mapping with vegetation and negative obstacle detection for vision-in-the-loop walking," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 3663–3670.

[8] P. Vernaza, M. Likhachev, S. Bhattacharya, S. Chitta, A. Kushleyev, and D. D. Lee, "Search-based planning for a legged robot over rough terrain." *ICRA*, pp. 2380–2387, 2009.

[9] J. Zico Kolter and A. Y. Ng, "The Stanford LittleDog: A learning and rapid replanning approach to quadruped locomotion," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 150–174, Jan. 2011.

[10] M. Zucker, N. Ratliff, M. Stolle, J. Chestnutt, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "Optimization and learning for rough terrain legged locomotion," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 175–191, Jan. 2011.

[11] S. Bazeille, V. Barasuol, M. Focchi, I. Havoutis, M. Frigerio, J. Buchli, D. G. Caldwell, and C. Semini, "Quadruped robot trotting over irregular terrain assisted by stereo-vision," *Intelligent Service Robotics*, vol. 7, no. 2, pp. 67–77, Apr 2014. [Online]. Available: https://doi.org/10.1007/s11370-014-0147-9

[12] I. Havoutis, J. Ortiz, S. Bazeille, V. Barasuol, C. Semini, and D. G. Caldwell, "Onboard perception-based trotting and crawling with the hydraulic quadruped robot (hyq)," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6052–6057, 2013.

[13] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krsi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 1184–1189.

[14] B. Katz, J. D. Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 6295–6301.

[15] A. S. Huang, E. Olson, and D. C. Moore, "Lcm: Lightweight communications and marshalling," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 4057–4062.

[16] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*, 2nd ed. O'Reilly Media, Inc., 2013.

[17] D. Kim, J. D. Carlo, B. Katz, G. Bledt, and S. Kim, "Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control," *arXiv.org*, Sep. 2019.

[18] M. H. Raibert, H. B. Brown, and M. Chepponis, "Experiments in Balance with a 3D One-Legged Hopping Machine," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 75–92, 1984.

[19] J. Andersson, J. Gillis, G. Horn, J. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.