

Lidar-based navigation-level path planning for field-capable legged robots

I. Havoutis*, D. G. Caldwell and C. Semini

*Dynamic Legged Systems Lab,
Department of Advanced Robotics, Istituto Italiano di Tecnologia,
Genova, 16163, Italy*
* E-mail: ioannis.havoutis@iit.it

In this paper we present a navigation framework for field deployment for the hydraulically actuated quadruped, HyQ.¹ Our framework uses a lidar sensor to perceive obstacles and free space around the robot and plans a path through traversable areas to given goals. A path following procedure generates velocity and angular rate commands that are sent to a locomotion controller that produces the trotting gait pattern, following the desired path to the goal.

Keywords: Legged Locomotion, Navigation, Quadruped, LIDAR.

1. Motivation

Legged robots are able to access a wide range of terrain types of varying difficulty. In unstructured environments, where only a number of discrete footholds are possible (disaster sites, forests, etc.), legged quadrupedal systems typically use walking gaits.^{2,3} In environments where smooth, continuous support is available (flats, fields, roads, etc.), where exact foot placement is not crucial for the success of the behavior, legged systems use a variety of more dynamic gaits, e.g. trotting, pacing, galloping.^{4,5}

Legged robots operating in the field are often directly controlled by a human. In many use cases, such systems can operate in an autonomous mode, provided that adequate perception and navigation capabilities are added. For example, such *locomotion autonomy* can enable a legged robot working in a construction site to reach material pick-up and drop-off checkpoints without human supervision. To this end we develop a navigation and locomotion framework that is capable of perceiving obstacles and free space around the robot using a lidar sensor, then plans and reliably follows paths to given goals using a trotting locomotion controller.

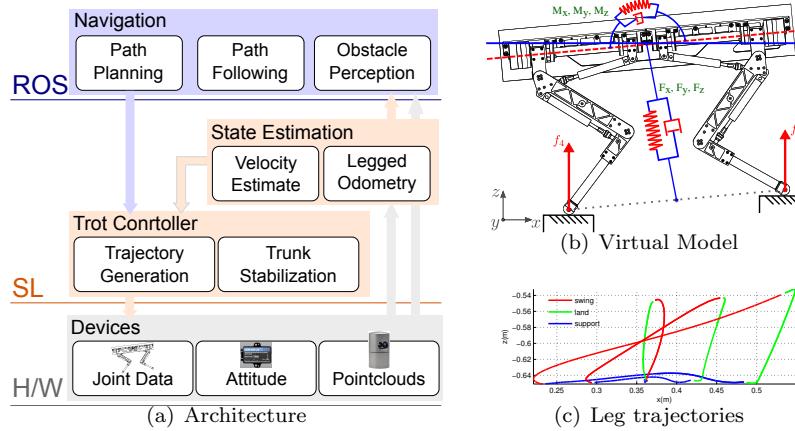


Fig. 1. (a): Overall architecture of the approach. (b): The virtual elements used to calculate forces and moments that stabilize the quadruped's trunk. (c): Trotting trajectories that correspond to the foot of the left front leg in the robot's body reference frame. The different colors denote the three gait cycle phases while the three trajectories correspond to trotting in place, trotting forward with a velocity of 0.5m/s and 1.0m/s.

1.1. Related work

A large body of literature in robotics is devoted to quadrupedal locomotion. Early examples include the work of Raibert⁶ on hydraulically and pneumatically actuated legged machines. Recent examples include Hawker and Buehler,⁷ and Remy et al.,⁸ focusing on passively compliant elements and light-weight leg structures. Close to our research stands Boston Dynamics' BigDog though little is still known about its control structure or experimental evaluation. In previous work we experimented with a feedback/feedforward control structure similar to the one presented here, coupled with a CPG-based trajectory generation procedure.⁵ We have also experimented with an active compliance control structure that emulates a passive telescopic leg behavior.⁴

Navigation-level path planning has received relatively small attention from the legged robots community as many of the applications concern close distances and often indoor usage. Such cases are often best suited with a variety of stereo and RGB-D sensors or combinations thereof. On the other hand many of the teams of the DARPA Urban Challenge made extensive use of lidar sensors and similar path planning approaches^{9,10} as such systems are tailored toward field deployment.

2. Locomotion & Navigation framework

An overview of our framework is presented in Fig.1(a). Our approach is divided into a navigation and a locomotion level, which in practice are handled with two separate onboard computers. The details of the locomotion and navigation levels follow.

2.1. Locomotion

We chose a trotting gait as we are interested in locomotion speeds ranging from trotting on the spot to fast-paced trot (1m/s). The trot is a symmetrical gait in which diagonally opposite legs swing in unison. This allows a desirable division of the total force that the leg actuators should be able to produce when supporting the weight of the robot, when thrusting, and when receiving impact forces at touchdown. Our locomotion approach divides the control procedure into two subsystems. One generates the trajectories that the two leg pairs execute given external user input, while the other calculates a feed-forward trunk stabilizing input according to the current state of the system. In the interest of space we briefly describe these subsystems below.

2.1.1. Trajectory generation

While trotting, the four legs are naturally divided into two pairs. Each pair follows an identical foot trajectory. This trajectory depends on the commanded forward velocity, turning rate, step and swing height, and the gait cycle period. The step height dictates the body height in the body reference frame during the support phase, and the swing height defines the leg swing apex. The gait cycle period is used to calculate the timing of the leg trajectories as this is intuitively further subdivided in three distinct phases; swing, land and support (Fig. 1(c)). We set the step height to -0.65m and the swing height to -0.55m, according to the dimensions of HyQ.¹ In our experiments we use a gait cycle period of 0.5s (2Hz), that is suitable for velocities up to 1m/s.

2.1.2. Trunk stabilization

To stabilize the trunk of the robot we follow a virtual model control approach.² We calculate virtual forces and moments according to a reference state and the current state of the system (Fig. 1(b)) as computed by an onboard state-estimation algorithm. The virtual forces and moments are then transformed to forces f_i that the feet in contact need to apply. We use

a least-squares optimization that provides the least norm solution, i.e. the minimum force solution. The feet forces are subsequently mapped to feed-forward torques (τ_{ff}) for the joint actuators of the legs that are in stance. This is done by: $\tau_{ff} = J^T \mathbf{f}$, where \mathbf{f} is the vector of (linear) forces that each foot in stance needs to apply to emulate the virtual model behavior, and J is the Jacobian of the legs that are in contact.

2.2. Navigation

The navigation framework deals with all the procedures that are required for the quadruped to navigate autonomously in a large-scale unstructured environment. In brief we separate this into four procedures that are needed to achieve the navigation task. Namely these are: *obstacle perception*, *path planning* and *path following*, and are explained in detail in the following sections.

2.2.1. Obstacle perception

The lidar (Velodyne, HDL-32E) produces about 700.000 points per second, with an 10Hz frame-rate. This produces a pointcloud $\mathbf{P} = [p_i, \dots, p_n]^T$ by ‘scanning’ the geometry of the environment. We then need to classify the points to two sets, the *obstacles set*, $\mathcal{O} = \{p_j\}$, and the *free-space set*, $\mathcal{F} = \{p_k\}$. We are most interested in a robust detection of the points belonging to the obstacle set as this is important for our navigation needs, i.e. these signify the areas that we need to avoid in the path planning step that follows.

We start by discretizing the area around the robot into a square grid where the edge of the square is $L = 10m$, while the resolution is set to $\ell = 0.1m$, i.e. each grid-cell is $0.1m^2$. We then consider all the points that belong to the point-cloud, \mathbf{P} , and accordingly assign them to their respective grid cells, $g_{i,j}$, according to their position. During this we keep track of the extrema points, in our case the minimum and maximum height points, that belong to each volume that a grid-cell defines along the vertical.

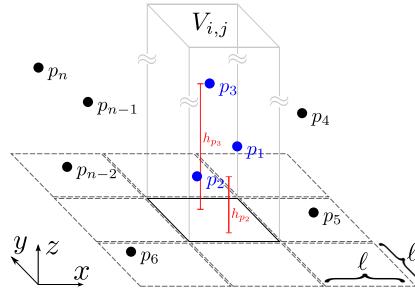


Fig. 2. Obstacle perception procedure. See Sec. 2.2.1 for details.

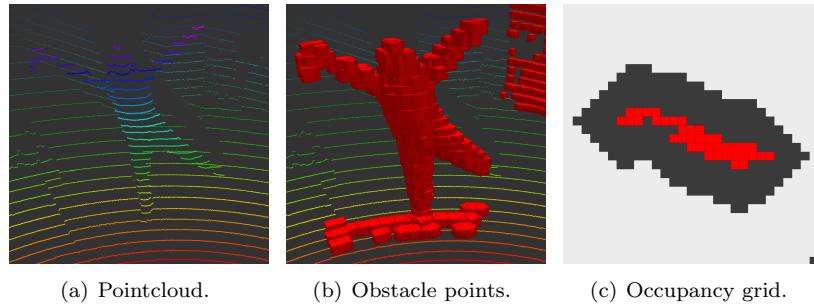


Fig. 3. The obstacle segmentation pipeline. (a) The pointcloud scan as published by the lidar. (b) The points that belong to the obstacle set \mathcal{O} . Note that the obstacle points are visualised as red cubes. (c) The (inflated) occupancy grid. The red cells are the direct projection of the obstacle points to the horizontal grid. The obstacle cells are then inflated according to the geometry of the robot, here represented by the grey grid-cells.

If the height difference between these two points is greater than a set obstacle threshold, $t_{\mathcal{O}}$, then all the points that belong to this volume (column) are added to the obstacle set.

For example consider the pointcloud in Fig.2. The points p_1, p_2 and p_3 belong to the grid-cell $g_{i,j}$ and accordingly to the volume, $V_{i,j}$, that this defines. The difference between the highest (p_3) and the lowest (p_1) points of the volume $V_{i,j}$ is then considered and accordingly all the points that belong to the volume in question are added to the obstacle set \mathcal{O} while the rest are added to the free-space set \mathcal{F} . The obstacle threshold, $t_{\mathcal{O}}$, is typically set at $0.25m$. A real-world example of the obstacle segmentation procedure is available in Fig. 3(a) and 3(b). The former presents the raw pointcloud as scanned, while the latter is the result of the obstacle perception procedure, where obstacle points are visualised as red cubes. Note that with this procedure there can be grid-cells that are not assigned any points. This grows with the distance from the sensor as the distance between two consecutive ‘scan-rings’ increases and is by definition true for areas that are ‘shadowed’ (occluded) by any other geometry present.

According to \mathcal{O} we populate an occupancy grid $G_{\mathcal{O}}$ that is communicated to the path planning step. Alternatively one can choose a voxel grid representation if 3D path planning is required. We selected an occupancy grid representation as it is adequate for navigation-level planning and drastically reduces the communication overhead.

2.2.2. Path planning

The obstacle perception step produces an occupancy grid that contains all the perceived obstacle areas within the defined grid. To simplify the path

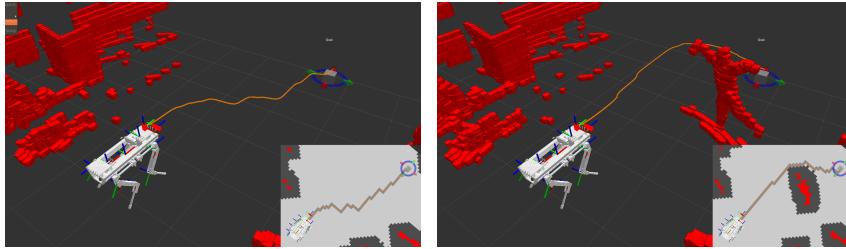


Fig. 4. Examples of the A* path planning results with the procedure described in Sec.2.2.2. The planned paths in 3D with the obstacle set points $p_j \in \mathcal{O}$ visualized as red cubes. *Inset plots:* Top-down view of the A* planned path to the current goal. The light-grey path on the grid is the path of cell-centers that is returned by the A* search algorithm. The orange line is the smoothed path that the quadruped follows.

planning problem we use the obstacle inflation approach. Accordingly all obstacle cells are ‘inflated’ according to an inflation factor that is dependent on the robot’s geometry. In our trials we use an inflation factor of $0.5m$, according to the robot’s width, which in practice works well since the quadruped has a preferred direction of motion. An omnidirectional locomotion approach would require a more elaborate obstacle inflation step that would take the possible orientation of the robot into account.

Once the inflated grid has been computed we use the A* search algorithm to find the shortest path from the current position of the robot to the goal position. The goal position is provided as user input and is expressed in the odometry (global) frame. For our current A* implementation we use a four-way connectivity and the Manhattan distance metric, while we use the Euclidean distance as a heuristic function. The produced path, $g_{i,j}^{1\dots n}$, is a sequence of cells in the inflated occupancy grid that is then transformed to the sequence of cell center-points.

The choice of four-way connectivity and Manhattan distance for the shortest path computation serves to keep the computational load to a minimum but in general produces non-smooth paths. We use a gradient decent approach to smooth the resulting path. This works iteratively by computing a smoothed version, $g_{sm}^{1\dots n}$, of the original path according to a smoothing coefficient. This is run until convergence, which in our experimental trials was on average below 10 iterations.

2.2.3. Path following

Next the planned path is converted to a meaningful command for the quadruped to follow. For this we use a ‘carrot’ path follower which operates as follows. The given path points, ranging from the quadrupeds current lo-

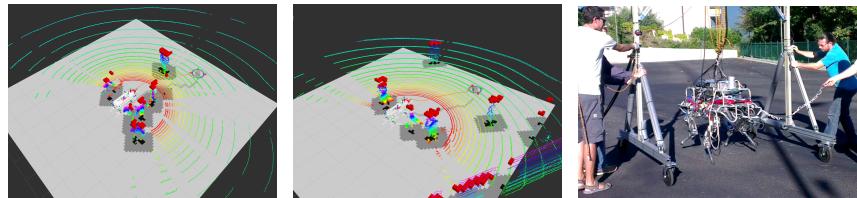


Fig. 5. *Left and middle:* Views of the world model that the robot builds-up. Occupancy grid and planned path to goal. *Right:* Picture of the HyQ trotting at the parking lot.

cation to the goal position, are sequentially examined. The path point that is d_c (length of the carrot stick) away from the robot is chosen to compute a twist command, i.e. a linear and angular velocity $t_c \in R^6$, towards this intermediate goal. The command is computed in a *PD* fashion and is constrained to forward linear velocity and angular yaw rate. A control logic that produces turning on-the-spot is also implemented for cases where the goal is beyond a reachable turning angle. Once no ‘carrot’ goal exists on the planned path, i.e. $\forall g_{sm} \nexists g_{sm}^i \geq d_c$, the goal has been reached.

3. Experimental trials

We ran a number of experimental trials with the robot navigating outdoors (Fig.5). The site of trials was a parking lot where adequate space was available to evaluate the quadruped’s behavior. On average we maintained a relatively cautious upper boundary for the velocity and the angular rate of the locomotion control which limited the speed at which goals were achieved. The quadruped never collided with obstacles or people moving around it. Nonetheless, a cautious inflation factor can limit the ability of the robot to navigate through narrow passages, for example single doors. On the other hand we do not expect to encounter such geometries in the field.

4. Conclusion

In this paper we presented a navigation framework, for legged robots operating in the field, that can provide locomotion autonomy. We showed how a lidar sensor can be used to create adequate perceptual input to enable navigation-level planning. We showed how lidar data can be used to classify obstacle and free space and subsequently use this world model to compute paths that avoid static and moving obstacles and reach goals that are tens of meters away in an outdoor environment. This approach makes use of a robust trotting controller to handle the robot’s locomotion behavior. This

way reliable path following was achieved by communicating desired velocity and angular rate commands.

In future work we aim to enable long and short-range map building and use semantic information to reason about path suitability. In addition we are developing a supervision system that will be able to choose between different locomotion modes, e.g. walking, trotting, galloping, based on the characteristics of the environment at hand.

Acknowledgments

This research is funded by the Fondazione Istituto Italiano di Tecnologia.

References

1. 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, in *Journal of Systems and Control Engineering*, 2011.
2. A. W. Winkler, I. Havoutis, S. Bazeille, J. Ortiz, M. Focchi, D. G. Caldwell and C. Semini, Path planning with force-based foothold adaptation and virtual model control for torque controlled quadruped robots, in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
3. A. W. Winkler, C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell and C. Semini, Planning and Execution of Dynamic Whole-Body Locomotion for a Hydraulic Quadruped on Challenging Terrain, in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.
4. I. Havoutis, C. Semini, J. Buchli and D. G. Caldwell, Quadrupedal trotting with active compliance, in *IEEE International Conference on Mechatronics (ICM)*, 2013.
5. V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri and D. G. Caldwell, A reactive controller framework for quadrupedal locomotion on challenging terrain, in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.
6. M. H. Raibert, *Legged robots that balance* (The MIT Press, Cambridge, MA, USA, 1986).
7. G. Hawker and M. Buehler, Quadruped trotting with passive knees: design, control, and experiments, in *2000 IEEE International Conference on Robotics and Automation (ICRA'00)*, 2000.
8. C. Remy, M. Hutter and R. Siegwart, Passive dynamic walking with quadrupeds - extensions towards 3d, in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010.
9. D. Dolgov, S. Thrun, M. Montemerlo and J. Diebel, Practical search techniques in path planning for autonomous driving, in *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08)*, (AAAI, Chicago, USA, June 2008).
10. D. Ferguson, T. M. Howard and M. Likhachev, Motion planning in urban environments, in *Journal of Field Robotics*, (11-12) (Wiley Subscription Services, Inc., A Wiley Company, 2008).