2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Onboard Perception-Based Trotting and Crawling with the Hydraulic Quadruped Robot (HyQ)

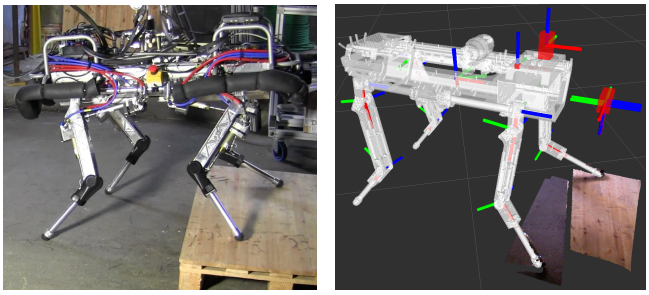
Ioannis Havoutis Jesus Ortiz Stéphane Bazeille Victor Barasuol Claudio Semini Darwin G. Caldwell

***Abstract*— This paper presents a framework developed to increase the autonomy and versatility of a large (∼75kg) hydraulically actuated quadrupedal robot. It combines onboard perception with two locomotion strategies, a dynamic trot and a static crawl gait. This way the robot can perceive its environment and arbitrate between the two behaviours according to the situation at hand. All computations are performed on-board and are carried out in two separate computers, one handles the high-level processes while the other is concerned with the low-level hard real-time control. The perception and subsequently the appropriate gait modifications are performed autonomously. We present outdoor experimental trials of the robot trotting over unknown terrain, perceiving a large obstacle, altering its behaviour to the cautious crawl gait and stepping onto the obstacle. This allows the robot to locomote quickly on relatively flat terrain and gives the robot the ability to overcome large irregular obstacles when required.**

I. Introduction

Legged robots are naturally superior at accessing a large variety of surface conditions than wheeled robots. This is partially due to the ability of legged systems (bipeds, quadrupeds, etc.) to decouple the path of the robot from the sequence of footholds and due to their inherent ability to utilize a range of locomotion strategies, tailored to the situation at hand. This way a legged robot –or animal– can chose a dynamic locomotion strategy, e.g. trot, gallop, when faced with terrain where accurate foot placement is not crucial for the success of the behavior. These can be situations where the robot locomotes on surfaces of varying conditions, e.g. grass, soil, pebbles, gravel, and varying inclination, where the support surface can be regarded as continuous. On the other hand, on structured environments or terrain with discrete footholds, e.g. steps, stairs, cluttered rooms, legged robots can employ a range of –typically non-gaited– static or quasi-static locomotion strategies that rely more on accurate foothold planning, and consequentially on the features of the terrain.

In the dynamic locomotion case, quadruped robots have been shown to successfully operate with minimal exteroceptive feedback [1], [2], [3], [4]. In most cases an on-board *inertial measurement unit* (IMU) is used to feedback the systems' attitude, while such input is adequate to overcome stochastic disturbances and deduce the surface inclination. In contrast non-gaited locomotion approaches rely more on exteroceptive feedback as higher order, cognitive, processes are critical to the success of the behavior. This, usually more computationally intensive, feedback comes from perception sensors, visual or depth, that are used to '*make sense*' of the robot's environment and are subsequently used –primarily– for foothold selection, map-building and localization, collision detection and avoidance, and navigation.



(a) Robot on step. (b) Robot model and depth data.

Fig. 1. *(a)* The HyQ quadruped robot stepping onto a wooden block that is $15cm$ tall. *(b)* The HyQ model within the ROS framework. This is used on-line in the perception layer to calculate transformations between the defined reference frames. The red rectangle in front of the quadruped's body represents the depth sensor and its coordinate frame. Note that the two snapshots do not correspond to the same time instant or trial.

Our main contribution in this paper is a control framework that can leverage the full set of benefits of legged locomotion, by combining a dynamic trotting behavior and a planning-based crawl-gait locomotion strategy. This allows our quadruped robot –*HyQ*– to autonomously and dynamically traverse continuous terrain of varying roughness and statically locomote over discontinuous, challenging unknown terrain. Such terrain includes big or irregular obstacles and environments where careful foot placement is crucial. In addition we wish to underline that all perception is performed on-board, i.e. no external motion capture system is used to localize the robot or portions of the environment, while also no *a priory* map of the environment is used. To our knowledge this is the first time that a large-scale quadruped robot combines these two types of locomotion strategies, dynamic trotting and planning based crawl-gait, without any external localization or map information.

I. Havoutis, J. Ortiz, S. Bazeille, C. Semini and Darwin G. Caldwell are with the Department of Advanced Robotics, Istituto Italiano di Tecnologia, 16163 Genova, Italy. *email*: {ioannis.havoutis, jesus.ortiz, stephane.bazeille, claudio.semini, darwin.caldwell}@iit.it V. Barasuol is with the Department of Automation and Systems, Federal University of Santa Catarina (UFSC), Florianpolis, SC, Brazi. *email*: victor@das.ufsc.br.

978-1-4673-6357-0/13/$31.00 ©2013 IEEE

II. HyQ robot overview

All work presented in this paper was carried out using a hydraulic quadruped robot, the *HyQ*. HyQ weights approximately 75kg, is 1m long, 0.5m wide and 1m tall when its legs are fully stretched. It is designed and built *in-house* and has 12 degrees of freedom, 4 of which are electrically actuated (hip *adduction/abduction* joints) while the rest are hydraulically driven [5]. In addition it has an on-board *inertial measurement unit* (IMU) and recently an RGBD color and depth sensor (Microsoft *Kinect*) has been integrated (Fig. 1).

HyQ is equipped with two on-board computers that communicate via *Ethernet*. There is a pc-104 that runs real-time patched Linux (*Xenomai*) and is responsible for the low-level hard-*real-time* control of the robot actuators. Also real-time critical parts of the controllers run on this machine. The second is a stronger pc (Intel i7, 16gb) that runs Linux as well as the open source *Robot Operating System* (ROS) [6]. It is dedicated to high level processes as state estimation, localization and mapping and foothold computation. ROS is responsible for reading and publishing data from the perception sensors. All the generally computationally intensive perception procedures are built as ROS nodes while the joint state and attitude of the robot is published by the real-time pc.

III. Related work

The work of Raibert [4] has been influential to research on dynamic locomotion approaches. Most of this was concerned with small to medium sized systems utilizing hydraulic and pneumatic actuators, and springs. Intuitive control laws were used for trotting while considerable gain tuning was required. Recently Rutishauser et al. [7] explored passive compliance approaches, where a lightweight passive spring-loaded quadrupedal structure with minimal actuation (1 motor per leg) was used. Remy et al. in [8] explored passively stable dynamic locomotion approaches over a range of different gaits, using a medium sized spring loaded model with a simplified telescopic leg. Ugurlu et al. in [9] experimented with trotting on a stiff electrically actuated quadruped robot where an active compliance control framework was used. Boston Dynamics' *Big Dog* and *LS3* are two of the most impressive quadrupedal platforms. Their remarkable performance has been demonstrated, mostly though videos online. However, no details on the hardware design or control methods have been published to date, therefore making the comparison against other approaches difficult.

A large body of literature in deliberate planning approaches in quadrupedal locomotion has resulted from the DARPA-funded learning locomotion challenge. The teams involved in this competition produced similar planning and control solutions [10], [11], [12]. All teams built upon a global plan over sets of given maps, while footholds were selected with hand-tuned, and learnt, features of varying terrain area resolution. The difference from our case is that in all this work a precise model of the environment was given to the planning and control framework, while an accurate motion capture setup provided the complete state of the robot.

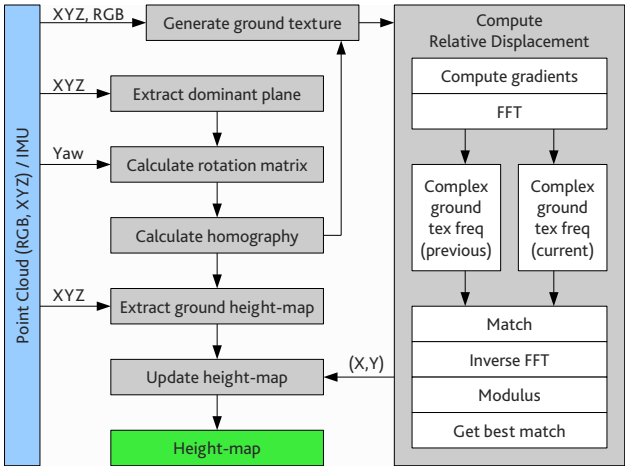


Fig. 2. Map builder pipeline. From the vision information (point cloud) and the sensory information (IMU), this algorithm extracts the pose of the camera with respect to the ground, and the relative transformation between two consecutive frames. Each item in the text corresponds to a block in the figure.

Kolter et al. in [13] took a step further in autonomy by removing the dependence on given maps and external state input. In their control framework they use a well established *point-cloud* matching technique to iteratively build a map of their environment and afterwards navigate in it.

A lot of work in robotics focuses on the problem of *simultaneous localization an mapping* (SLAM). A recent example that is popular with ROS and the Kinect *RGBD* sensor is [14]. This has been used to generate maps that can be stored offline and used to localize in an online manner [15]. Using such approaches in real-time is an open problem as such solutions are generally computationally more demanding while they often operate in a time-scale that is inappropriate for a fast-paced legged robot. In previous work [16] we used single frame information from a stereo camera to track visual targets or modulate parameters of a trotting gait.

IV. Perception and map generation

In this work, we made the assumption that walking takes place on a mostly flat and horizontal surface and we designed an algorithm that can work in these conditions in real-time. We plan to improve the current pipeline by incorporating the Iterative Closest Point (ICP) algorithm for terrains that do not have a dominant plane.

The map generation relies also in other sensory information, such as the IMU. In practical terms, we are using only the yaw information. The map building pipeline as shown in Fig. 2, and the related computations are done as follows: **1)** Acquire the sensory information including the *point cloud* with color information (*XYZ/RGB*) and the trunk *yaw* from the IMU. **2)** Extract the *dominant plane* from the point cloud. This is done using the standard RANSAC algorithm on XYZ values. The plane extraction gives us the height

of the camera (Z). **3)** Calculate the *rotation matrix*. Using the normal of the dominant plane previously calculated, and the yaw information coming from the IMU. **4)** Calculate the transformation between the ground space and the camera space (*homography*), using four pairs of points from the point cloud laying on the dominant plane. **5)** Project the*RGB* points from the point cloud into the ground surface and generate the *ground texture*. This texture includes only intensity and not color information, in single precision floating point format. Only the points laying on the dominant plane are projected onto this texture. The empty spaces are filled with -1.0. **6)** Compute the relative displacement between two consecutive frames *(X, Y)*. Instead of performing a direct comparison between the ground texture, we do a matching in the frequency domain, that offers good matching performance at a reasonable computational requirement. Our method is based on the phase correlation technique [17] to which we replace the texture by the gradients. The matching is more efficient when working with gradients because it enhances the image features [18] while empty space in the texture is disregarded as gradients at such patches correspond to zero. Also, FFT works in the complex domain, this way we use the gradient in each direction, both the real and imaginary components. **7)** At this point, we have computed the 6 DoF of the relative transformation between two consecutive frames. Next, we extract the height-map of the current frame, and by applying the transformation we update our 'global' terrain map. We use a statistical approach using the average of the last stored values.

A variation over the previous pipeline could calculate also the yaw during the ground plane matching. In our case we opted for calculating only the relative displacement of the ground, because the yaw information coming from the IMU is robust enough and doesn't drift much over time. At the same time this also saves some calculations.

The map we generate has a resolution of $1cm^2$. In our application we don't need to memorize a big surface, but just the temporal ground surface under the robot. For this reason we use a cyclic map that wraps the ground coordinates.

V. Control Framework

The control framework can arbitrate between two locomotion behaviours. The first is a dynamically stable trot that is suitable for fast locomotion over regular terrain of varying inclination. The second behaviour is a deliberate planning and crawl-gaited locomotion style that is tightly coupled to the perceptual layer of the system and suitable for irregular and non-continuous terrain situations. The choice between the two behaviours is based on the perception layer of the system and uses a simple roughness criterion, often overly cautious, similar to [19].

A. Dynamic trotting controller

The dynamic trotting controller, also named Reactive Control Framework (RCF) in previous work [1], consists of two main blocks, named *Motion Generation* and *Motion Control* blocks (Fig. 3), that work in harmony to provide

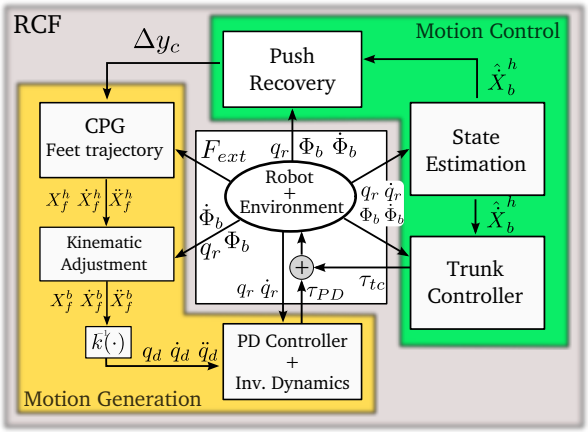


Fig. 3. Diagram of the dynamic trotting controller, also called Reactive Control Framework (RFC) in previous work. This highlights the main functional blocks and the information flow.

suitable feet trajectories and to control the robot's trunk motion and posture.

The *Motion Generation* block is built on an algorithm based on Central Pattern Generators (CPGs). The CPGs used here consist of four non-linear oscillators, synchronized according to the desired gait, that provide outputs as position references for each foot. Each oscillator has parameters directly associated to the step height, step length, forward velocity and duty factor, which we consider as *locomotion parameters* that can be modulated independently.

The *Motion Control* block is mainly concerned with the robot's balance. It uses torque, positions and IMU measurements and implements a *push recovery* and a *trunk controller* algorithm. The push recovery algorithm computes suitable footholds that drive the robot naturally to the default posture after an external disturbance. The trunk controller algorithm computes the joint torque references of the stance legs, to obtain a desired force and moment acting on the trunk.

B. Deliberate planning controller

We chose a craw-gait strategy that is not dynamic, in the sense that the CoM of the quadruped has phases where its forward and lateral velocity are zero. This is beneficial for rough irregular terrain where the interaction needs to be minimal as the terrain conditions can change unexpectedly. This problem arises as the interaction model cannot be overly accurate, for example forward momentum can make a large rock move but cautious stepping in general will limit this effect. In the next subsections we describe the locomotion pattern used, the controller to realize this gait and the procedure that selects appropriate footholds.

1) Locomotion gait: We use a cyclic crawl gait that utilizes a static stability criterion to produce the quadruped's walking pattern. The gait cycle follows the pattern; left hind (*LH*) to left front (*LF*) to right hind (*RH*) to right front leg (*RF*). An overview of this cyclic gait is sketched in Fig. 4. This pattern minimizes the trajectory length that the trunk travels within the alternating support triangles, while in our implementation there is no backward motion. The swing

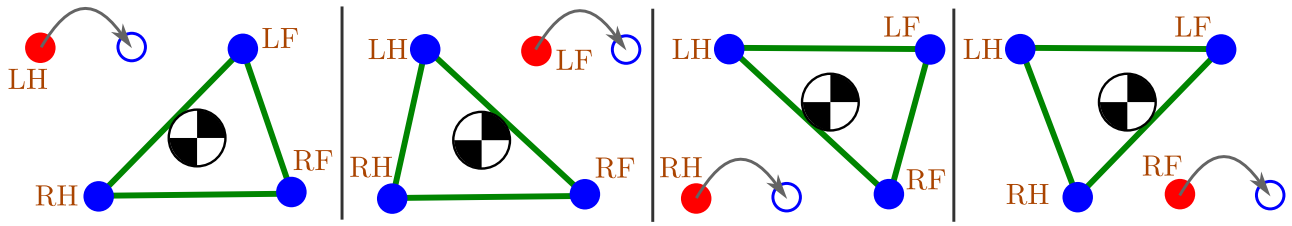


Fig. 4. The craw gait locomotion strategy. From left to right, the succession of swing legs and support triangles. The CoM is positioned over the current support triangle before a leg swing motion is initiated. Note that the CoM of the quadruped moves only forward.

targets are provided by the foothold selection procedure, using feedback from the perception layer, as described in the following section. Furthermore the attitude of the robot's trunk is modified accordingly, resulting in pitching up when the robot is stepping on an obstacle and pitching down when the robot is stepping off an obstacle. This is beneficial as it increases the overlap between the fore or hind legs' workspace with the environment geometry. This way the legs seldom reach an overstretched configuration.

2) Foothold selection: The success of the crawl gait locomotion controller depends highly on the selection of appropriate footholds. This problem has been the focus of much research work resulting in a variety of generally cost based approaches [10], [12]. Most of such approaches assume accurate and global knowledge of the locomotion surface and often millimeter-accurate localization from an external source, typically a motion capture system. In our approach we assume that the terrain is unknown, thus cost functions that depend on multiple terrain resolutions are not usable. We focused our effort to a local cost evaluation on the vicinity of a nominal swing target reference. This way the controller queries the perception layer with a Cartesian point in the robot's frame of reference. This in turn serves as a midpoint for the surface evaluation of a predefined grid of an area of $0.2m^2$ around the given point. A height-map centred around this point is extracted and the surface derivatives are calculated (*local gradient*). The surface derivatives with respect to the plane are summed together and the resulting matrix is convolved with a 3×3 matrix of ones to produce a filtered estimate of the local surface change. This can be regarded as a local costmap of the given surface based on the rate of change of the surface. A mask with 6 predefined 4×4 target 'windows' is used to calculate the cost at these 6 patches. The patch with the lowest score is the one where the underlying surface exhibits the least change and the lowest score patch is chosen as the subsequent swing leg target. Fig. 5 presents an example of the local foothold selection procedure. The size of the local patch and the size of the cost windows was selected based on the size of the robot's foot ($2.5 \times 2.5cm$).

3) Locomotion controller: The control framework uses an inverse dynamics calculation procedure and a low-gain PD controller to execute the planned motions (Fig 6). The benefits from this control setup are twofold and can be regarded also as coupled. First the interaction with the environment, especially when the contact estimation is not accurate, is smoother as the feedback control (PD) is highly compliant. Second the accuracy of tracked motions is greatly increased, resulting to more accurate contact estimation and interaction with the environment.

We use the floating base inverse dynamics algorithm presented in [20] that produces a numerically robust and analytically correct solution for the desired actuator torques, τ, given a reference trajectory that the gait controller generates, i.e. $(\mathbf{q}_r, \dot{\mathbf{q}}_r, \ddot{\mathbf{q}}_r)$. In the interest of space we refer the interested reader to the inverse dynamics work in [20], nonetheless a brief overview of the approach is given below.

In the usual formulation the system configuration is represented as:

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_r^T & \mathbf{x}_b^T \end{bmatrix}^T, \tag{1}$$

where $\mathbf{q}_r \in \mathbb{R}^n$ is the joint configuration of the rigid-body robot model, where $n = 12$ for our case, and $\mathbf{x}_b \in SE(3)$ is the position and orientation of the coordinate system attached to the robot base, typically the robot's trunk, and measured with respect to the global inertial frame.

The equations of motion when the quadruped is in contact with the support surface are:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^T\tau + \mathbf{J}_C^T(\mathbf{q})\lambda, \tag{2}$$

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{18\times18}$ is the floating base inertia matrix, $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{18}$ is the floating base centripetal, Coriolis and gravity forces, $\mathbf{S}$ is the actuated joint selection matrix (in essence separating base and robot joints) $\tau \in (R)^{12}$ is the vector of joint torques, $\mathbf{J}_C \in \mathbb{R}^{k\times18}$ is the constraint Jacobian of k linearly independent constraints, and $\lambda \in \mathbb{R}^k$ is the vector of k linearly independent constraint forces. Constraints are typically foot contact locations where external forces

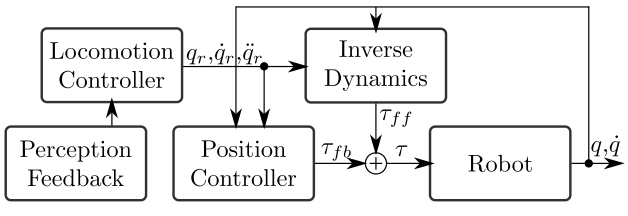


Fig. 6. The control structure used to preform the planned motions resulting from the non-gaited locomotion behaviour. The locomotion controller receives perceptual feedback in the form of appropriate stepping targets (Fig.5). The locomotion controller produces a reference plan for the inverse dynamics and the low-gain PD controller. The resulting torques of both the *feedback* and *feedforward* block are combined and sent to the robot.

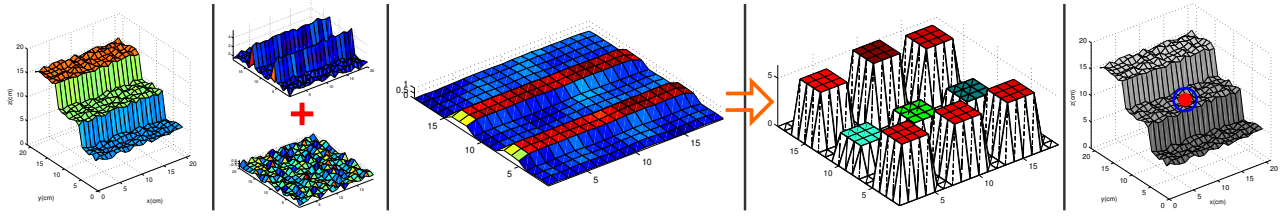


Fig. 5. An overview of the local foothold selection procedure. *From left to right*; a 0.02×0.02 surface around the nominal swing target is converted to a height map. The local gradients are then computed, note that the surface estimate and surface gradients are typically noisy. The combined local gradient is filtered with a 3×3(cm) matrix and a mask with 6 equidistant windows is used to calculate the local cost of each of the 6 patches. The midpoint of the patch with the lowest cost is chosen as the swing target. This way the quadruped avoids stepping near sudden surface height changes (steps) and promotes stepping on the most flat path that the local surface can provide. The size of the local patch and the size of the cost windows was selected based on the size of the robot's foot.

are applied to the floating base system, i.e. *ground reaction forces* (GRFs).

The solution is of the form,

$$\tau = (\mathbf{S}_u\mathbf{Q}^T\mathbf{S}^T)^+\mathbf{S}_u\mathbf{Q}^T[\mathbf{M}\ddot{\mathbf{q}}_d + \mathbf{h}], \tag{3}$$

where $\mathbf{S}_u$ is a selection matrix and $\mathbf{Q}$ is the *dynamics projection* matrix that results from a *QR* decomposition of the control space.

One problem that the above formulation poses occurs when contact conditions change. In that instant, the inverse dynamics solution changes from one support triangle to the next, something that can result in abrupt torque changes and can potentially result to sudden jerks that can render the system off-balanced. As the contact conditions change in a planned fashion we overcome this limitation by interpolating between the two contact model solutions in a linear manner.

VI. Experimental Results

We have carried out a set of experimental trials both inside our lab, locomoting on a treadmill, and outside, on relatively flat concrete ground. In the latter case we have used wooden boards to simulate irregular terrain as well as a flat wooden construction of approximately $1m^2$ area and $15cm$ height. In this setting the quadruped is asked to move forward while no overall -navigation level- planning was utilized.

Trotting: The trotting controller as mentioned earlier is able to trot with a speed up to $0.5m/s$ according to the constraints imposed by the perception layer. During trotting the robot can overcome obstacles up to $5cm$ without difficulty. This is achieved due to the controlled compliance of the legs combined with the trunk attitude corrective term as further described in [1]. When the robot perceives obstacles that are taller than $5cm$ the trotting controller is brought to a stop and the behaviour switches to the crawl gait.

Crawl-gait: The crawl gait is a slow locomotion behaviour with emphasis on stable foot placement. The velocity of the trunk in this mode never exceeds $0.05m/s$ in the forward direction as the motions are executed with strict stability bounds in mind. Nevertheless with this locomotion mode we have been able to overcome obstacles up to $0.15m$, over 20% of the HyQ's leg in a fully extended configuration. Occasionally foot-slipping was observed, but this disturbance was never large enough to impact the stability of the robot.

Perception: Overall the perception layer has been consistent while it operates with an average bandwidth of $15hz$. When the crawl-gait controller queries the perception layer for the next locally optimal foothold, it takes on average $0.04s$ to complete this computation and to both-ways communicate. In more detail, this is the time taken for the query to be sent from the crawl-gait controller to the ROS layer, to transform the point to the appropriate reference frame, retrieve the local patch from the built map, calculate the locally optimal point and reply back to the controller.

Data from a complete trial is presented in Fig.8. In this example the robot trots forward until it perceives an irregular obstacle. This results to a transition from trot to crawl where the robot needs to halt and reposition its legs accordingly. The crawl gait locomotion controller is then used and the two fore legs step on the obstacle. The data in Fig.8 is the z-axis position of the robot's feet in the trunk (local) reference frame. Snapshots from an experimental trial with a $15cm$ high step are presented in Fig. 7.

Limitations: We experimentally evaluated the trot controller to be utilized with speeds up to $0.5m/s$. Nevertheless the trot controller can locomote with a velocity up to $2.0m/s$. When using a fast-paced trot ($v > 0.5m/s$), or sometimes

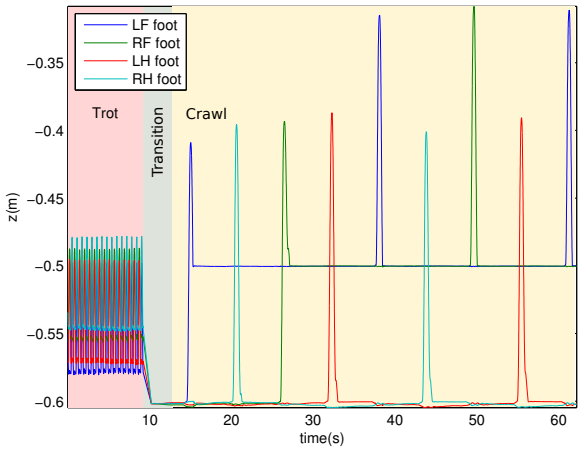


Fig. 8. The transition from trotting to crawling gait as seen from the feet positions in the robot's trunk (local) coordinate frame. The robot perceives an obstacle while trotting and its behaviour switches to the crawl gait. The left front foot then steps on the obstacle and the crawl gait cycle continues.

Fig. 7. An experimental trial on concrete terrain with a $15cm$ high step. *From left to right*: the robot trots up to the obstacle/step and the perception layer stops the trotting behaviour. The behaviour switches to the crawl-gait controller that uses perceptual feedback to step on the flat obstacle. The complete video sequence is available as accompanying material.

when locomoting over irregular terrain, the map building procedure has difficulty registering new input data. This can be improved by using a state estimation procedure, that can feedback to the internal position estimate that the perception layer keeps. Another difficulty was that the depth/vision sensor has a limited field of view in this configuration. This restricts map building to an area directly underneath the front part of the robot. In the future we aim to address this limitation using a stereo camera that can provide better/faster depth estimates and a larger field of vision/depth perception. This way we will be able to utilize faster trotting by combing planning in a larger scale (navigation) map.

VII. Conclusion

We presented a framework that utilizes on-board perception to adapt a large quadruped's behaviour according to the situation at hand. This framework can switch between a dynamic trot and a stable crawl gait. The former is chosen when locomoting over regular and inclined terrain while the latter is used for traversing irregular and often structured environments. All computations, processing and control are performed on-board and no external information or a-priory knowledge of the environment was used.

Future work: We are currently working on a dynamically balanced deliberate planning controller that uses the *ZMP* stability criterion. This will serve to increase the robot's locomotory speed over irregular terrain and minimize unwanted forward/backward robot trunk oscillation. In addition we are working on a more elaborate vision based mapping system that combines height map data with surface classification, and on a more robust terrain/foothold cost estimation technique that allows better foothold selection and more effective use of the features of the environment.

Acknowledgement

This research is funded by the Fondazione Istituto Italiano di Tecnologia. The authors would like to thank the colleagues that collaborated for the success of this project: Hamidreza Saghir, Bilhal Rehman, Hamza Khan, Jake Goldsmith, Marco Frigerio, Thiago Boaventura, Michele Focchi, and our team of technicians.

References

[1] 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.

[2] I. Havoutis, C. Semini, and D. G. Caldwell, "Virtual model control for quadrupedal trunk stabilization," *Dynamic Walking*, 2013.

[3] I. Havoutis, C. Semini, J. Buchli, and D. G. Caldwell, "Quadrupedal trotting with active compliance," *IEEE International Conference on Mechatronics (ICM)*, 2013.

[4] M. H. Raibert, *Legged robots that balance*. Cambridge, MA, USA: The MIT Press, 1986.

[5] 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," *Journal of Systems and Control Engineering*, 2011.

[6] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009.

[7] S. Rutishauser, A. Sproewitz, L. Righetti, and A. J. Ijspeert, "Passive compliant quadruped robot using central pattern generators for locomotion control," in *IEEE International Conference on Biomedical Robotics and Biomechatronics*, 2008.

[8] C. Remy, M. Hutter, and R. Siegwart, "Passive dynamic walking with quadrupeds - extensions towards 3d," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[9] B. Ugurlu, K. Kotaka, and T. Narikiyo, "Actively-compliant locomotion control on rough terrain: Cyclic jumping and trotting experiments on a stiff-by-nature quadruped," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[10] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," *The International Journal of Robotics Research*, 2011.

[11] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[12] P. Vernaza, M. Likhachev, S. Bhattacharya, A. Kushleyev, and D. D. Lee, "Search-based planning for a legged robot over rough terrain," in *IEEE International Conference on Robotics and Automation*, 2009.

[13] J. Z. Kolter, Y. Kim, and A. Y. Ng, "Stereo vision and terrain modeling for quadruped robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[14] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3d visual slam with a hand-held rgb-d camera," in *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, 2011.

[15] A. Hornung, K. M. Wurm, and M. Bennewitz, "Humanoid robot localization in complex indoor environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010.

[16] S. Bazeille, V. Barasuol, M. Focchi, I. Havoutis, M. Frigerio, J. Buchli, C. Semini, and D. G. Caldwell, "Vision enhanced reactive locomotion control for trotting on rough terrain," in *IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, 2013.

[17] B. Reddy and B. N. Chatterji, "An fft-based technique for translation, rotation, and scale-invariant image registration," *IEEE Transactions on Image Processing*, 1996.

[18] G. Tzimiropoulos, V. Argyriou, S. Zafeiriou, and T. Stathaki, "Robust fft-based scale-invariant image registration with image gradients," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[19] R. Hoffman and E. Krotkov, "Terrain roughness measurement from elevation maps," in *SPIE Vol 1195 Mobile Robots IV*, 1989.

[20] M. Mistry, J. Buchli, and S. Schaal, "Inverse dynamics control of floating base systems using orthogonal decomposition," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.