Reactive Trotting with Foot Placement Corrections through Visual Pattern Classification

Victor Barasuol [1], Marco Camurri[1], Stephane Bazeille[1], Darwin G. Caldwell[1] and Claudio Semini[1]
firstname.lastname at iit.it

***Abstract*— Agile robot locomotion on rough terrain is highly dependent on the ability to perceive the environment. In this paper, we show how the interaction between a reactive control framework and an online mapping system can significantly improve the trotting performance on irregular terrain. In particular, this new locomotion controller increases the stability of the robot and reduces frontal leg and shin collisions with obstacles by correcting in realtime the foothold locations. The mapping system uses an RGB-D sensor and a motion capture system to build a three dimensional map of the surroundings of the robot. While the robot is trotting, the control framework requests in advance a local heightmap around the next nominal foothold position. Then, an optimized foot placement location is estimated by applying visual pattern classification on the acquired heightmaps, and the leg endpoint trajectory is modified accordingly. The foothold correction is performed independently for each leg. To show the effectiveness of our approach the controller was tested both in simulation and experimentally with our 80 kg hydraulic quadruped robot, *HyQ*. The results show that visual based reaction through pattern classification is a promising approach to increase locomotion robustness over challenging terrain.**

I. Introduction

Legged robots have the potential to navigate in more challenging terrains than traditional wheeled robots. Unfortunately, their control is more difficult because in addition to the traditional mapping and path planning requirement, they have to deal with more specific issues, such as balancing or foothold planning. At this level, the perception of the environment is crucial to enable the robot to navigate while coping with irregular terrains and avoiding obstacles along its path.

At the *Istituto Italiano di Tecnologia (IIT)*, our fully torque-controlled Hydraulic Quadruped robot (HyQ) [1] has been designed to perform highly dynamic tasks on difficult terrains. In recent years, the robot has demonstrated a variety of locomotion capabilities, among these are walking, trotting and jumping. More recently, we have started to integrate some perception sensors and demonstrated: IMU based balancing [2] which allows blind trotting on rough terrain; visually assisted trotting [3], [4] where we use frame by frame stereo data to increase the step height and reduce the robot velocity to be able to overcome bigger obstacles; and a vision based gait transition [5], where we evaluate the difficulty of the terrain using a RGBD camera to switch from trotting to crawling and avoid obstacles on the way. Finally, last year we demonstrated fully planned obstacle crossing on *a priori* known terrain [6] computing and executing a series of specific footholds in order to avoid a gap or climb a step.

Contribution: in this paper, we introduce a novel reactive locomotion behavior that uses the 3D map of the environment to determine the foothold positions in order to overcome obstacles while avoid leg/object collisions with both the front and hind legs. By reactive we mean that no path planning is performed: leg trajectories are modified on the fly, according to the local heightmap extracted around each nominal foothold position and processed by a low computational cost algorithm based on machine learning.

The mapping is performed online using an RGB-D camera and a motion capture system that provides the robot position and orientation at a fast rate (250 Hz). During the experiments the robot was teleoperated by the user who was sending high level commands (forward velocity, heading) along an arbitrary path over terrain strewn with obstacles. In this paper, we present results both for simulated and real indoor experiments with the robot HyQ.

The paper is organized as follows: in the next section we will discuss the related work and emphasize the novelty of our approach. In Section 3 and 4 we will describe the robot and detail the mapping framework. In Section 5 and 6 we will explain how learning is used and how the motion control works. Finally in Section 7 we will show experimental results before concluding in Section 8.

II. Related Work

Significant progress has been achieved during the last few years in the field of robot perception abilities. But more specifically in the context of quadruped and highly dynamic robots, this problem has received less attention. The perception of the environment is, however, crucial for locomotion as soon as the terrain gets challenging. Nowadays, quadruped robots are more commonly used to develop low-level controllers, rather than high-level cognitive processes. Furthermore, legged locomotion requires precise and failsafe perception proficiency, even during dynamic motions, impacts, or complex visibility conditions. However, there has been limited research on the integration of vision sensors and the subsequent demonstration of online use of perception in a real system performing dynamic motions.

Kolter *et al.* [7] presented the most autonomous approach by performing vision based SLAM with optimal planning. In their control framework they register the point clouds acquired with a stereo camera and use a texture synthesis

*This work was supported by Istituto Italiano di Tecnologia (IIT)

[1] Department of Advanced Robotics, Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genova, Italy.

algorithm to fill occluded areas to perform motion planning with their quadruped *LittleDog* [8]. The vision processing and path planning were performed on an external computer. Using the same robot, Kalakrishnan *et al.* [9] achieved accurate foothold planning on given maps. A precise model of the environment was previously acquired while an accurate motion capture setup provided the complete state of the robot. Also with *LittleDog*, Filitchkin and Byl [10] used a monocular camera to perform terrain classification and select between predetermined gaits to traverse terrain of varying difficulty.

Stelzer *et al.* [11] developed a complete stereo vision based navigation framework for their hexapod robot. The robot pose is estimated by fusing IMU data with relative leg odometry and visual odometry measurements. A digital terrain map and a traversability map are built using the images and the computed pose. Then, paths are planned using a D* lite planner and the robot determines the appropriate path for the terrain it must traverse. In a similar way, [12], [13] achieved navigation with obstacle avoidance but without foothold planning. They also fused information from stereo vision, leg odometry, and IMU to obtain the state estimation of the quadruped robot *BigDog*. Then, they used a 2D cost map and A* based path planning to perform navigation.

Bajracharya *et al.* [14] recently showed terrain mapping and obstacles classification for vision-in-the-loop walking on the *LS3* robot. The vision system was used to map the environment in the vicinity of the robot and inform the gait generator about the changes in the surface over which the robot is locomoting, then the gait's step height was modified. Finally, Shao *et al.* [15] also presented some path planning with obstacle avoidance on their quadruped robot that uses a stereo vision-based terrain modeling algorithm.

The main difference between the present work and those highlighted above is that we are focusing on reactive behavior while the robot is teleoperated. The robot follows the high level commands sent by the user while adjusting the foothold according to the terrain on the fly. Before moving, the map is unknown as well as the robot trajectory. Compared to our previous work on vision assisted reactive trotting and vision based gait transition [3], [5] we added in this paper a supervised learning method to improve the foothold adjustment and we worked on the mapping robustness to be able to avoid the obstacle also with the hind legs. The object/legs collision were the main issue in our previous approaches since we had a crude terrain model. Front foot collisions might cause damage to the robot structure and make the robot get stuck or fall, and shin collisions might cause slippage and confuse the locomotion controllers since all the forces measured by the joint load cells are assumed to come from the ground/feet contact interactions.

It has to be noticed that in [16] a blind local reflex approach for terrain negotiation was proposed to face leg collisions problems. This approach is simpler but the main drawback is that it requires high retraction torques and more leg workspace as the robot velocity increases.

In this paper we take advantage of our mapping framework to increase the locomotion robustness through vision-based reactions. When traversing an uneven terrain, the visual perception is used to better exploit the leg's workspace while selecting footholds to avoid frontal foot collisions and leg shin collisions.

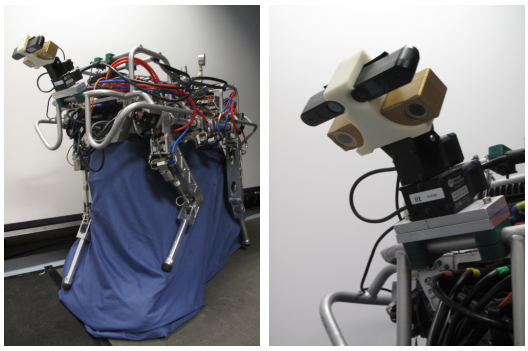

Fig. 1. The quadruped robot HyQ and its vision system: Left: the whole robot, Right: View of the robot's active head consisting of an RGB-D camera and a stereo camera (here not used) mounted on a pan and tilt unit.

III. System overview

HyQ [1] is a versatile hydraulically actuated quadruped robot that weighs $80\,\mathrm{kg}$, is $1\,\mathrm{m}$ long and $1\,\mathrm{m}$ tall (Fig. 1). The robot's legs have three degrees of freedom each, two joints in the sagittal plane (hip and knee flexion/extension) and one joint for hip abduction/adduction.

A. Sensors

The robot is equipped with both proprioceptive and exteroceptives sensors which include:

- high-resolution encoders and torque sensors for every joint;
- a LORD Microstrain® 3DM-GX3-25 IMU;
- an ASUS® Xtion PRO LIVE RGB-D sensor.

The camera is mounted on a FLIR PTU-D46-17 Pan and Tilt Unit (PTU), which has a pan range of $\pm 159°$, and a tilt range of $-47°/+31°$. This is mounted with an inclination of $32°$ with respect to the robot's horizontal frame and with a tilt value set to $45°$ for the experimental setup described in this paper. These values allow the robot to see between $20\,\mathrm{cm}$ and $80\,\mathrm{cm}$ ahead. The relative position and orientation between the PTU and robot frame estimation is discussed in the next Section.

B. Calibration of the system

For this kind of vision-based locomotion task, the camera and the robot must be properly calibrated, to compute the transformation matrix ${}_bT_c$ between the optical frame and the robot base. In our situation the PTU support frame was fixed by hand on the robot protection frame. As a consequence its position and orientation were unknown and could not be estimated manually with accuracy. To properly calibrate this system we developed an automatic procedure based on the visual tracking of the foot, inspired by the PR2 vision/arm calibration [17]. The method presented in [18] consists in

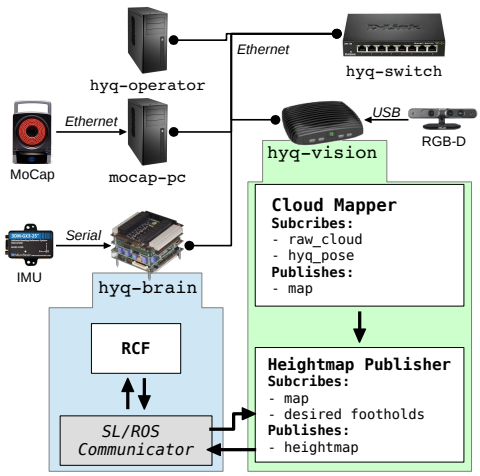

Fig. 2. Sketch of the hardware/software architecture. The robot has two onboard computers; *hyq-brain* (blue field) executes our Reactive Control Framework (RCF) [2] on the Real-Time environment SL [20] and also handles the communication between SL and ROS; *hyq-vision* (green field) executes the mapping modules, taking from the motion capture system the robot poses. A separate computer, *hyq-operator*, is used to send high level commands to the robot.

tracking with a camera a colored marker attached to a front foot and estimating the transformation between the camera frame and the base frame using a set of different positions.

As the PTU can move, the transformation matrix is computed for a default position and then updated in real-time with commodity methods provided by the Robot Operative System (ROS) and the PTU software driver. A more detailed description about the issues related to the estimation of ${}_bT_c$ and an alternative calibration method are described in [19].

C. Software Architecture

The experimental setup involves the use of two onboard computers: a PC-104 ("hyq-brain") computes the foot placement and controls the motions of the robot, while an IntensePC ("hyq-vision") is dedicated to mapping. Two external computers are used to send user commands and to localize the robot through a motion capture system. All the computers share the same local network. The hardware/software architecture is shown in Fig. 2.

Two software frameworks are used: the PC-104, which runs real time linux, controls of the robot through SL [20], a simulation and real-time control software package. All other computers use ROS as the communication interface. We developed a dedicated interface between SL and ROS based on shared memory to share information (Fig. 2, grey box). Message synchronization is guaranteed by a NTP server (Network Time Protocol), together with the internal synchronization features provided by ROS.

IV. Mapping

The mapping system (Fig. 2, green box) generates two distinct representations of the environment: the first one is a dynamically built map that describes the most recently seen local terrain; it is obtained by merging the point cloud while the robot moves. When merging, all the points in the map that overlap with the latest scan are discarded, to be consistent with changes in the environment.

The second representation is composed of a collection of four 2D heightmaps — one for each foothold — which are computed by projecting the points of the point cloud onto the horizontal frame (see Section IV-B).

Immediately after the touchdown of a foot, the Reactive Controller Framework (RCF), presented in [2], collects the heightmap around the next desired position of that foot and applies the trajectory adjustment, according to the displacement returned by the logistic regressor (see Section V).

In the following sections we will describe the two nodes that compute the representations and we will define the horizontal frame.

A. Cloud Mapper

The update of the map is described in Alg. 1: when a new cloud in the camera frame C_c is available, it is transformed into base link frame b, together with the actual map M_w (where ${}_bT_w$ is provided by the motion capture system (MoCap)). Then, every point $m \in M_w$ that lies outside the border of $C_b = {}_bT_c\, C_c$ is added to it. The resulting sum is cut around the robot using pre-defined margins and passed through a voxel filter, to make it more compact. The result is finally re-transformed into the fixed-world frame w.

As at each step the map is expressed inside the fixed-world frame, the fusion of clouds is obtained by simple point addition. In a context where no exteroceptive sensors are availble, using such a frame would be impossible because of accumulated drift. In fact, a locally defined frame as described in [19] is suitable for the task described in this paper. Future work will include the integration of proprioceptive localization and local mapping of [19] with the work described herein.

The extent of the map and the voxel size are crucial parameters for the system: small voxels (*e.g.,* $< 1\,\mathrm{cm}$) would make the cloud too dense to be processed in time (*i.e.,* in less than the sampling period), while a sparse cloud would produce gaps in the heightmap. Similarly, a big map would reduce the output frequency of the mapper. For our experimental setup we used a voxel size of $2\,\mathrm{cm}$, and a map $2\,\mathrm{m}$ wide and $3.5\,\mathrm{m}$ long, with the robot center placed $0.25\,\mathrm{m}$ in front of the map

Algorithm 1 Point Cloud Merging

$C_b \leftarrow {}_bT_c\, C_c$ ▷ Cloud into base link
$M_b \leftarrow {}_bT_w\, M_w$ ▷ Map into base link
for each $m \in M_b$ **do**
  if $m_x < \min_x C_b$ **or** $m_y < \min_y C_b$ **or** $m_y > \max_y C_b$ **then** ▷ m is outside the xy border of C_b
    $C_b \leftarrow C_b + m$
  end if
end for
$F_b \leftarrow \mathrm{filter}(C_b)$
$M_w \leftarrow {}_wT_b\, F_b$

center. These parameters met the desired update frequency of 30 Hz, using the equipment described in Section III-A.

B. Horizontal Frame

The trajectories of the end effector through our framework, are expressed in the so called Horizontal Frame (HF), defined in [2]. The HF has the same origin as the base link, but its Roll and Pitch values are always aligned with gravity. All the desired foothold positions and the corresponding heightmaps are referred to this frame, since using the base link would result in wrong corrections (*e.g.,* if the base link is inclined with respect to the gravity, all the trajectories would be skewed accordingly). The HF is computed by a separate node, which is not shown on the architecture sketch of Fig. 2 for sake of conciseness.

C. Heightmap Publisher

Given a desired foothold position f, the corresponding heightmap is computed by projecting the Point Cloud Map onto the horizontal frame and cutting a square of $d^2 \cdot r^2$ around the xy-plane of f, where d and r are the number of pixels per line and the width of each pixel in meters, respectively. Each pixel value is the average of the z (height) of all the points of the cloud within the area covered by that pixel. When no points are available, the default z of the desired position is sent. For our setup we used $d = 15$ and r = 2 cm. It has to be noted that the system can deal with dynamically changing values, as the desired foothold message that is sent by the RCF contains also d and r. To reduce the effect of the noise, a Gaussian Filter of size 3×3 pixels is also applied to the heightmap before being sent. Note that since the heightmap is generated from an uncolored point cloud, changes in light conditions or issues related to RGB artifacts are not affecting the classification.

V. Visual Pattern Classification

For each desired foothold position, finding the corresponding trajectory adjustment to avoid the obstacle is treated as a classification problem: the feature vector $\mathbf{x} = [1 x_1 \cdots x_n]^{\mathrm{T}} \in \mathbb{R}^{(n+1)\times 1}$ is given by the n pixel values of the heightmap and the extra 1 coefficient to include the bias term, while the class $y \in \mathbb{N}^m$ is an instance taken from a pool of m possible corrections applicable to that foothold position (see Fig. 3).

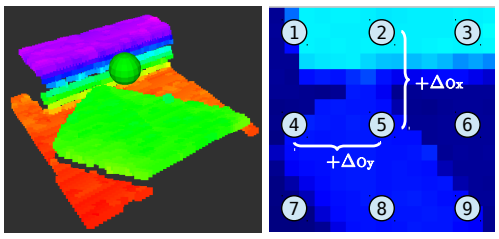

Fig. 3. *Left*: example of point cloud in false colors. The cloud covers an area of 30×30 cm. The green ball indicates a desired foothold position. *Right*: the corresponding heightmap in false colors (different from *left*) with the 9 possible footholds adjustments and the displacements according to the robot axis convention.

In the following sections we will describe how the training set was generated and how the set of weights for the regressor has been computed.

A. Training set generation

In this paper, we opted for a heightmap composed of 15×15 pixels, and for an output set of 9 possible displacements, one for each cardinal direction plus the "null" displacement. Hence, $\mathbf{x} \in \mathbb{R}^{226\times 1}$ and $y \in \mathbb{N}^9$.

For each leg, the corresponding training set was generated as follows: first we defined 33 different patterns (see Fig. 4) that fit a variety of possible obstacles, namely: stairs, bars, logs or stones. For each pattern we acquired 100 samples from the depth sensor, yielding a set of 3300 input examples.

Since the reaction to an input depends also on which leg is involved, the set of examples was replicated and independently labeled for each leg, according to the pattern it belongs to.

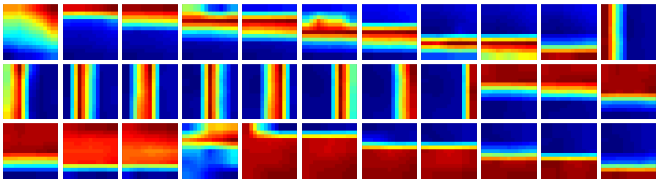

Fig. 4. Examples of pattern, taken from the training set. The images are in false colors and rescaled between minimum (dark blue) and maximum (dark red) values of the pattern. Each pixel represents the average height of a 2 cm^2 area of terrain, referred to the horizontal frame.

B. Logistic Regression for Foothold Decision

As a classifier we opted for a logistic regressor [21], which had been demonstrated to be sufficiently robust (90% success rate) and fast (as it involves only four 15×15 matrix multiplications) during our preliminary tests with the patterns shown in Fig. 4.

Since we have more than one class (in our case $m = 9$) we opted for a One-vs-All multi-class classification by training m binary classifiers h_{θ_i}, $i \in \mathbb{N}^m$ where $\theta_i \in \mathbb{R}^{(n+1)\times 1}$ is the weight vector for the i-th class and:

$$h_{\theta_i} = h_{\theta_i}(\mathbf{x}) = \frac{1}{1 + \exp\left(-\theta_i^{\mathrm{T}}\mathbf{x}\right)} \tag{1}$$

Given a specific leg, a class $c \in \mathbb{N}^m$ and a training set of k examples $(\mathbf{x}^{(i)}, \tilde{y}^{(i)})$, with $\mathbf{x}^{(i)} \in \mathbb{R}^{(n+1)\times 1}$, $i \in \mathbb{N}^k$, and $\tilde{y} \in \{0, 1\}$ indicates whether $y = c$ or not, we compute the weight vector θ_c by minimizing the cost function:

$$J(\theta_c) = +\frac{1}{k}\sum_{i=1}^{k} C(h_{\theta_c}(\mathbf{x}^{(i)}), \tilde{y}^{(i)}) + \lambda \sum_{j=1}^{n} (\theta_c^j)^2 \tag{2}$$

where: $C(h_{\theta_c}(\mathbf{x}^{(i)}), \tilde{y}^{(i)})$

$$= \log\left(h_{\theta_c}(\mathbf{x}^{(i)}) + (1 - \tilde{y}^{(i)}) \log\left(1 - h_{\theta_c}(\mathbf{x}^{(i)})\right)\right) \tag{3}$$

and $\lambda = 0.001$.

From the weight vectors we define the weight matrix of a leg as $\Theta_{\text{leg}} = [\theta_1\ \theta_2\ \theta_c\ \ldots\ \theta_m]^{\mathrm{T}} \in \mathbb{R}^{(n+1)\times m}$ and $\mathbf{h}_{\Theta_{\text{leg}}}(\mathbf{x}) = [h_{\theta_1}\ h_{\theta_2}\ \cdots\ h_{\theta_m}]^{\mathrm{T}}$.

The predicted class y for a test example $\mathbf{x}_t$ is then computed as the index of:

$$\max\left(\mathbf{h}_{\Theta_{\text{leg}}}(\mathbf{x}_t)\right) \tag{4}$$

We assigned to each class y a corresponding offset $(\Delta^{\text{leg}}_{Ox}, \Delta^{\text{leg}}_{Oy})$, as depicted in Fig. 3.

VI. Reactive Motion Control

A. Reactive Controller Framework and Step Adjustment

As the basis for locomotion control and generation, we use our Reactive Controller Framework presented in [2]. In this framework the kinematic references are generated by four non-linear oscillators that trace an elliptical trajectory that can be modulated according to the terrain.

The idea of a vision based reactive trot is to modulate the step by moving the center of the foot elliptical trajectory (origin) according to the terrain surface around its expected foothold, trying to reach the touch-down position that avoids a collision. We define these relative displacements of the origins as $\Delta^{\text{leg}}_{O} = [\Delta^{\text{leg}}_{Ox}\ \Delta^{\text{leg}}_{Oy}\ \Delta^{\text{leg}}_{Oz}]^{\text{T}}$, where leg = LF (left-front), RF (right-front), LH (left-hind) and RH (right-hind).

Each Δ^{leg}_{O} takes discrete values defined according to a decision grid, composed by cells inside the local height map. The value at the center of each decision grid cell represents the average surface height inside its area, as illustrated in Fig. 3 for a decision grid composed of nine cells. The Δ^{leg}_{Ox} and the Δ^{leg}_{Oy} components take discrete values that are determined by the distance between the center cell (the nominal foothold position) and the cell chosen to avoid undesired collisions. The Δ^{leg}_{Oz} component is the difference between the surface height in the chosen cell and the expected height at touch-down.

Important stages in the proposed approach are the prediction and reaction phases. For each leg, the prediction and reaction phases happen simultaneously during part of the swing phase period. More precisely, both phases start at the foot lift-off and end when the desired foot trajectory reaches the maximum foot clearance.

B. Input grid sizing

In our first studies we found out that the grid sizing is mainly determined by the step length, the body pose, the workspace of each leg and the size of the obstacle to be avoided (*i.e.,* to not step on). The next paragraphs explain the roles each of these factors plays on the grid sizing.

1) Avoiding blind spots: The first task when sizing the grid is to identify the occurrence of blind spots, *i.e.,* regions that are not visually sampled between two consecutive steps and hence where the robot will not react to the obstacles. The blind spots appear from the relationship between grid size and step length, as illustrated in Fig. 5 for the case of a gait duty factor $D_f = 0.5$ (D_f = stance phase duration/step cycle duration).

According to Fig. 5, the grid size G_s must be equal or greater than $2L_s$ to avoid a blind spot between two consecutive footholds. Since the robot is supposed to perform omnidirectional locomotion this relationship is extended to both grid length and width. At the critical limit, *i.e.,* $G_s = 2L_s$, any trunk disturbance might change the foot swing trajectory creating a blind spot. This is one of the reasons why we prefer overlapping ($G_s > 2L_s$) between to consecutive grids.

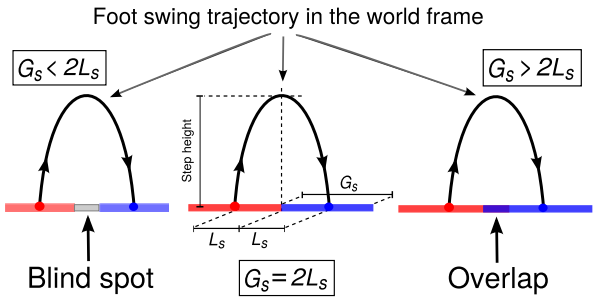

Fig. 5. Perspective view of the foot swing trajectory and the occurrence of blind spots according to the relationship between the grid size G_s and step length L_s.

2) Shin collisions and body pose: the body pose (*i.e.,* roll, pitch and trunk height) directly affects the chances of having shin collision because it defines the angle between the lower limb of the leg and the ground surface. Higher trunk heights allow the robot to avoid higher obstacles, but reduce the maximum horizontal workplane, thus reducing the maximum grid size, while lower trunk heights cannot clear some obstacles but have a larger workplane grid size (Fig. 6).

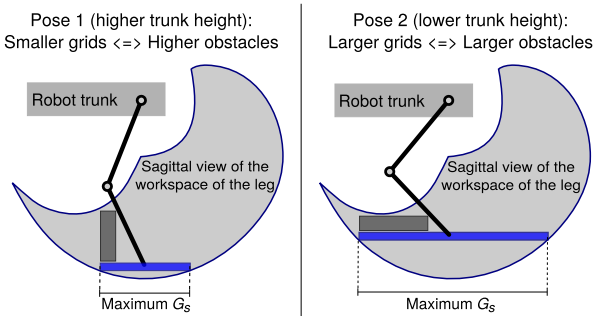

Fig. 6. Trunk pose dictates the maximum grid size and consequently the range of objects the robot is able to detect and overcome.

3) Grid size and object avoidance: When a new heightmap around the predicted foothold is created, the robot must decide between keeping the nominal foothold or stepping around an obstacle. However, how can one know that such an obstacle is not a bar, a rock, or a stair? It is the size of the grid that defines the objects size one is able to identify. A small grid, even if defined with some overlap to avoid blind spots (Section VI-B.1) might be too small to identify if the height map variation inside the grid comes from the shape of a bar/rock or from the beginning of a stair. In this case, the user must define G_s so that the objects the robot must be able to identify, and possibly deviate from, fit inside the grid.

C. Mapping, prediction and tracking: the three main issues

In this section we highlight three main issues that limit the performance of our approach: the map provided by the perception system, the prediction of the footholds and

the execution of the movements to match the desired foot placement (position tracking errors).

1) Mapping: The perception system sends to the RCF a height map, around a desired foothold. Such a mapping is commonly prone to errors and might provide maps that have drifted and contain noise because the robot localization in the world is not accurate and the point is noisy. Ignoring the noise is part of the learning process but drift on the map might lead to a reaction that is not needed.

The second issue is the update rate of the mapping pipeline. With our RGB-D camera and the localization with a motion capture system we achieved mapping at approximately $25\,\mathrm{Hz}$. Indeed computational cost is relatively low and we are only limited by the camera frame rate, but with a stereo camera and a registration algorithm this update rate could be smaller. The map rate dictates how fast the robot is able to visually react. For example, a $25\,\mathrm{Hz}$ rate means a new map each $40\,\mathrm{ms}$. If the robot is running at $1\,\mathrm{m/s}$ then it will be visually insensitive to the last $4\,\mathrm{cm}$ walked. Mapping inaccuracies and the update rate are the bottleneck of this approach drastically reducing the accuracy of any subsequent robot action. These problems will be addressed in Section VI-D.

2) Foothold prediction: The foothold prediction depends on the swing phase trajectory of the foot and the trunk velocity. The swing phase trajectory and period are considered well known, since they are pre-defined and, by now, independent from parameters related to motion control. On the other hand, the trunk velocity estimation is challenging. Apart from the systematic errors, the estimated velocity signal presents peaks that come from touch-down impacts and an oscillatory component due to some inherent trunk swinging. These signal features affect directly and proportionally the prediction of the next foothold location. Thus, feeding back the estimated velocity directly into the predictor is risky. For example, if at the end of the prediction phase the predictor receives the estimated velocity resulting from a touch-down impact or from the instant the trunk reaches its maximum swing velocity, the robot will react according to a map that is centered far from its true footfall. A low-pass filter applied to the estimated velocity drastically reduces the miss-predictions due to peaks, however, it does not cancel the trunk oscillatory component, while decreasing the cut-off frequency would introduce too much delay in the visual feedback loop.

To solve this problem a moving average filter was used, tuned to the step frequency of the robot, making the filter time window $t_w = 1/2f_s$. Figure 7 shows the estimated trunk velocity and the filtering for the two different techniques.

From Fig. 7, we can see that the moving average filters the signal peaks and the inherent oscillations. Thus, the filtered velocity feed-back into the predictor becomes the average velocity for the swing phase period, leading to a much more accurate foothold prediction.

3) Position tracking errors: The position tracking error is an issue related to the execution of the reactive motion. Since the motion is reactive, tracking the desired joint positions might be hard. Achieving small tracking errors is very challenging if one seeks compliance and safety. Hence, the robot may react correctly visually but still not execute properly. With our robot we found out that the position tracking error might vary from $0.5\,\mathrm{cm}$ to $4\,\mathrm{cm}$, depending on the external disturbance forces.

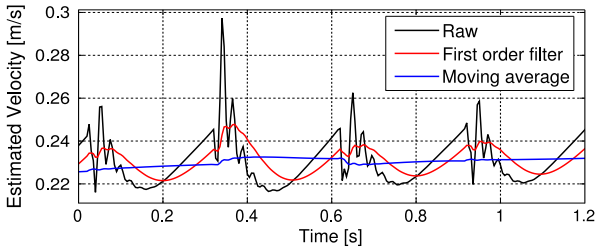

Fig. 7. Estimated trunk forward velocity for three different methods. The black line refers to the velocity given by the state estimation (raw velocity). The red and blue lines are the results of applying a first order low-pass filter and a moving average filter on the raw velocity, respectively.

D. Decision Maker

The decision maker is the function block that incorporates the learned inference and, according to the decision grid internal distances, provides the discrete Δ^{leg}_{Ox}, Δ^{leg}_{Oy} and Δ^{leg}_{Oz} to the RCF described in [2].

In this paper we consider a grid size $G_s = 0.3\,\mathrm{m}$ that is enough to detect objects of about $10\,\mathrm{cm}$ length while having no blind spots for velocities of 0 to $0.5\,\mathrm{m/s}$.

Because our approach is implemented on a real robot, the decision maker must take into account the three main issues explained in Section VI-C. Regarding the mapping issue, only the noise and the map update rate are considered. The drift is supposed to be zero, as we use a VICON system to build the map. The other two main issues remain as previously detailed.

Due to the three main issues, an *uncertainty region* is created around the ideal foothold location. Inside this uncertainty region we can not ensure that the reaction will overcome the obstacle or avoid a frontal or shin collision. To address this problems we opted for a decision maker that takes *conservative action*, as depicted in Fig. 8.

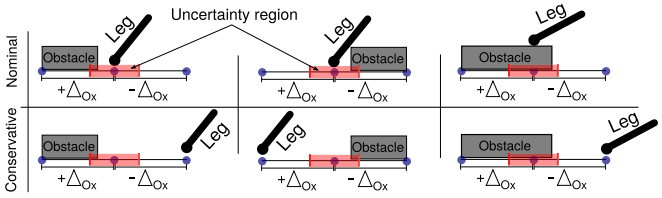

Fig. 8. Examples of conservative actions. The blue dots are foothold locations and the translucent red box represents the uncertainty region around the nominal foothold. The conservative actions take footholds far from the uncertainty region to avoid an unexpected collision with the obstacle.

To avoid creating excessive joint torques, the discrete decision values are filtered by a first order filter before being sent to the RCF. The time constant of the filter is adjusted on the fly according to the leg swing time. The response time of the filter is chosen to be eight times faster then the swing time.

VII. Simulated and Experimental Results

A. Vision based trotting in simulation

In this simulation, we show how locomotion robustness can be substantially improved by using vision information to execute foot placement in a reactive approach. To demonstrate this, we created two distinct scenarios: in the first one the robot had to cross a single pallet that is $10\,\mathrm{cm}$ high and $70\,\mathrm{cm}$ long; in the second instance, the robot had to cross four bars, each $10\,\mathrm{cm}$ high, differently spaced from each other ($0.3\,\mathrm{m}$, $0.35\,\mathrm{m}$ and $0.4\,\mathrm{m}$). Beside its simplicity, this task clearly shows the effects of foot frontal collisions and leg shin collisions. The simulated scenario is illustrated in Fig. 9.

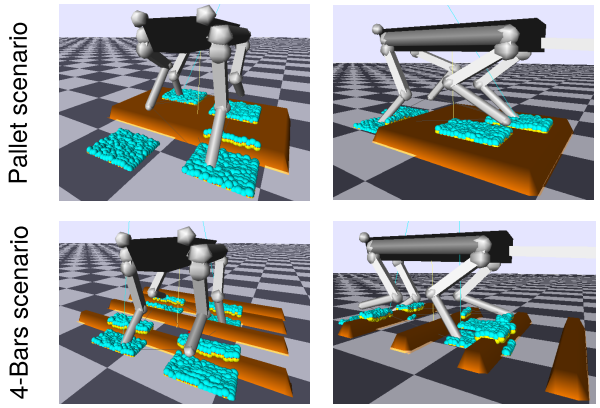

Fig. 9. Simulated scenarios: pallet crossing task (top) and multi-bars crossing task (bottom).

In order to evaluate improvements, we considered two cases: a) our standard reactive trotting, in which the kinematic references can be adjusted to the terrain surface and b) our standard reactive trotting with vision based reaction for foot placement. In both cases, the robot trots without heading control, with step heights of $0.12\,\mathrm{cm}$, a duty factor equal to 0.5, a step frequency equal to $1.7\,\mathrm{Hz}$ and desired forward velocity of $0.5\,\mathrm{m/s}$ (pallet case) and $0.25\,\mathrm{m/s}$ (4-bars case). We consider a 15×15 grid size with $2 \times 2\,\mathrm{cm}$ cells and $2\,\mathrm{cm}$ of perception noise when simulating the vision system. Many trials were performed for both cases to have the robot crossing the obstacles at different moments in the step cycle. The robot velocity was considered for analysis since it is easily affected by leg collisions. The results are shown in Fig. 10.

These results show that the use of perception improve the locomotion robustness. Indeed, the trunk velocity is more stable and the heading disturbances are reduced when the perception is used. In contrast, when the perception is not used, the velocity drops to negative values, due to frontal and shin collisions, and sometimes the robot gets stuck. Over the whole simulations performed using perception, frontal collisions were completely eliminated and the shin collisions significantly reduced. It has to be mentioned that the study on the grid sizing and cell dimensions are out-of-scope in this paper and will be explored in a future work.

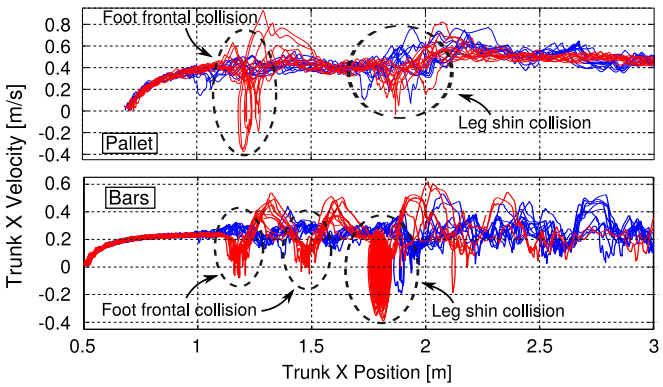

Fig. 10. Trunk velocity according to the distance walked for the two crossing tasks: pallet (top) and 4-bars (bottom). The desired velocity is $0.5\,\mathrm{m/s}$ for the pallet scenario and $0.25\,\mathrm{m/s}$ for the 4-bars one.

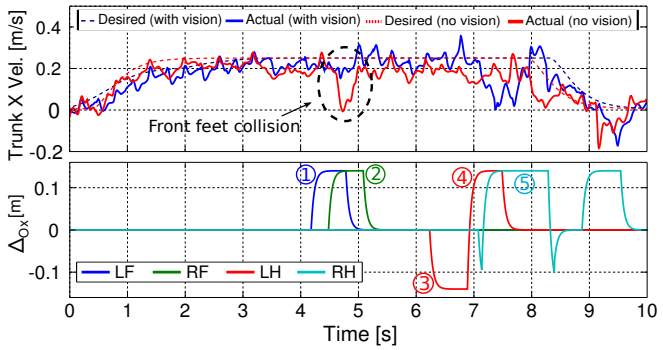

Fig. 11. Experimental results for a single bar crossing task with and without visual reaction. The top plot shows the velocity performed by the robot for both cases. The bottom plot show the robot reactions when the vision is activated. Obs: the velocity droppings between 7-9 seconds are due to safety pulls to protect the robot from a collision with the robot crane structure

B. Experimental results

The experimental scenario is similar to the multi-bar crossing task described in Section VII-A. The robot had to traverse a single bar with desired velocity of $0.25\,\mathrm{m/s}$. The decision maker was setup to take conservative actions, as described in Section VI-D, for a 15×15 pixels heightmap and a 9×9 decision grid.

For the experimental results we demonstrate the robot reactions by showing the Δ_{Ox} of each leg. Since the bar is orthogonal to the robot longitudinal axes, there will not be lateral reactions (*i.e.*, $\Delta_{Oy} = 0$). The velocity tracking and the robot reactions are shown in Fig. 11, where each reaction is enumerated and detailed in Fig. 12 by means of frames extracted from the video footage of the experiment.

From the top plot of Fig. 11 one can see that, when the vision is activated, the robot reacts with its four legs and steps over the obstacle, maintaining the forward velocity. Without vision, the robot is not able to avoid the obstacles. In this case there is a collision with both front legs that makes the velocity drop to $0\,\mathrm{m/s}$ (time plot between $4\,\mathrm{s}$ and $5\,\mathrm{s}$).

Figure 12 shows snapshots of the following reactions: 1) LF foot moved forward, performing a longer step to overcome the bar; 2) RF foot moved forward to overcome the bar; 3) LH

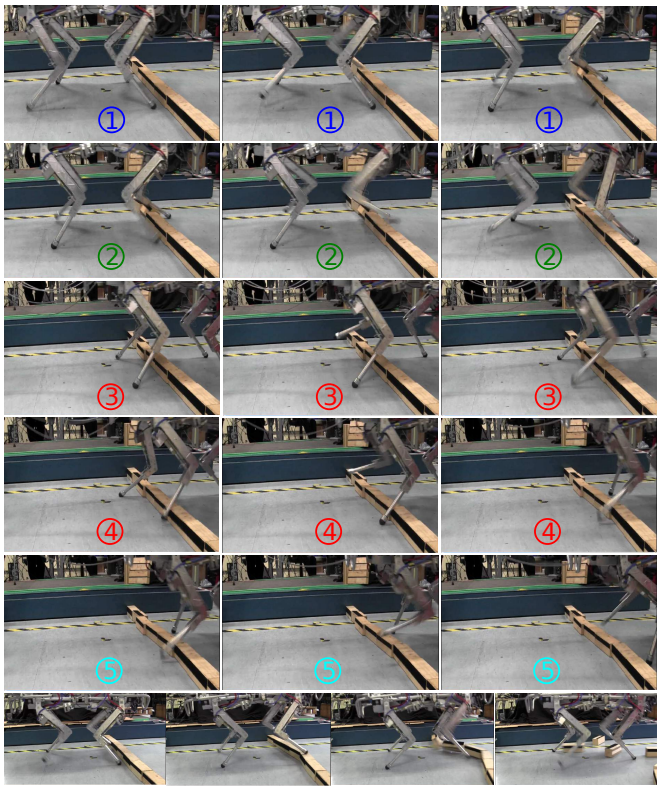

Fig. 12. Snapshots of the robot reactions when overcoming the bar (the robot moves from left to right). The snapshots are identified according to the enumeration from Fig. 11 (bottom plot). Snapshots from the experiment without visual reaction are shown in the bottom row.

foot moved backward to avoid a collision; 4) LH foot moved forward to overcome the bar and 5) RH foot moved forward to overcome the bar. Both simulated and experimental results showed that the proposed approach can substantially reduce the motion disturbances due to foot frontal collisions and legs shin collision.

VIII. Conclusion

In this paper we presented a novel reactive locomotion behavior that uses the 3D map of the environment to estimate the best foot placement inside the leg's workspace to limit undesired variations of the robot's pace on uneven terrain. The mapping framework uses an RGB-D camera and a motion capture system for localization. A heightmap around each future foothold is used in the controller to improve the foothold by means of supervised learning. Results show that the visual pattern classification significantly reduces both front foot and shin collisions. As a consequence, the robot behavior is smoother and more stable, while the control remains simple, robust, and fully reactive (no path planning). Our future work will focus on developing an accurate onboard SLAM with our stereo camera and perform experiments outdoor on more difficult terrain. Also, we planned to extend our gait adaptation approach by increasing the resolution and the complexity of the learned patterns.

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," *J. of Systems and Control Engineering*, 2011.

[2] 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 ICRA*, 2013.

[3] 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 TePRA*, 2013.

[4] 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*, pp. 1–11, 2014.

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

[6] A. Winkler, I. Havoutis, S. Bazeille, J. Ortiz, M. Focchi, R. Dillmann, D. G. Caldwell, and C. Semini, "Path planning with force-based foothold adaptation and virtual model control for torque controlled quadruped robots," in *IEEE ICRA*, 2014.

[7] J. Z. Kolter, K. Youngjun, and A. Y. Ng, "Stereo vision and terrain modeling for quadruped robots," in *IEEE ICRA*, 2009.

[8] J. Pippine, D. Hackett, and A. Watson, "An overview of the Defense Advanced Research Projects Agency's Learning Locomotion program," *Int. J. of Robotics Research*, 2011.

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

[10] P. Filitchkin and K. Byl, "Feature-based terrain classification for littledog," in *IEEE/RSJ IROS*, 2012.

[11] A. Stelzer, H. Hirschmüller, and M. Görner, "Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain," *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 381–402, 2012.

[12] J. Ma, S. Susca, M. Bajracharya, L. Matthies, M. Malchano, and D. Wooden, "Robust multi-sensor, day/night 6-dof pose estimation for a dynamic legged vehicle in gps-denied environments," in *IEEE ICRA*, May 2012, pp. 619–626.

[13] D. Wooden, M. Malchano, K. Blankespoor, A. Howardy, A. A. Rizzi, and M. Raibert, "Autonomous navigation for bigdog," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4736–4741.

[14] M. Bajracharya, J. Ma, M. Malchano, A. Perkins, A. Rizzi, and L. Matthies, "High fidelity day/night stereo mapping with vegetation and negative obstacle detection for vision-in-the-loop walking," in *IEEE/RSJ IROS*, 2013.

[15] X. Shao, Y. Yang, and W. Wang, "Obstacle crossing with stereo vision for a quadruped robot," in *ICMA*, 2012.

[16] M. Focchi, V. Barasuol, I. Havoutis, J. Buchli, C. Semini, and G. D. Caldwell, "Local reflex generation for obstacle negotiation in quadrupedal locomotion," in *Int. Conf. on Climbing and Walking Robots (CLAWAR)*, 2013.

[17] V. Pradeep, K. Konolige, and E. Berger, "Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach," in *International Symposium on Experimental Robotics (ISER)*, New Delhi, India, 12/2010 2010.

[18] S. Bazeille, M. Camurri, J. Ortiz, I. Havoutis, D. G. Caldwell, and C. Semini, "Terrain mapping with a pan and tilt stereo camera for locomotion on a quadruped robot," in *ICRA14 Workshop on Modelling, Estimation, Perception and Control of All Terrain Mobile Robots (WMEPC14)*, 2014.

[19] M. Camurri, S. Bazeille, C. Semini, and D. G. Caldwell, in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2015.

[20] S. Schaal, "The sl simulation and real-time control software package," University of Southern California, Tech. Rep., 2009.

[21] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2009, ch. 4.