Terrain mapping with a pan and tilt stereo camera for locomotion on a quadruped robot

Stephane Bazeille[1], Marco Camurri[1], Jesus Ortiz[1], Ioannis Havoutis[1], Darwin G. Caldwell[1] and Claudio Semini[1]

Abstract— **Legged robots are expected to have superior mobility on rough terrain than wheeled robots.**

The main reason is that legged locomotion is more adaptable to a wide range of terrain types as the robot can decompose its path into a sequence of footholds and can use different locomotion strategies. In order to accomplish most of the locomotion tasks the robot requires high level control (*i.e.,* to adjust the locomotion parameters and to choose optimal footholds) which depends on real-time localization and accurate terrain mapping. In this paper, we propose a SLAM solution using a pan and tilt stereo camera mounted on an hydraulically actuated quadruped robot that builds a map and keeps track of the robot's position.

Since the computation needs to be carried out on board and the robot is subject to considerable motion during its locomotion (regular vibrations, impacts or slippages), we developed a dedicated implementation based on fast stereo depth computation, GPU based map building and mechanical motion compensation. Combined with a foothold planning framework presented in our previous work [1], this localization and mapping ability allows to perform locomotion in a fully planned manner. Successful results of foothold planning with our quadruped robot show the effectiveness of our method.

Keywords: Stereo Vision, SLAM, Legged Locomotion.

I. INTRODUCTION

Legged robots have the potential to navigate in more challenging terrains than other robots do. Unfortunately their control is more demanding because they have to deal with the traditional mapping and path planning as well as some more particular issues like balancing and foothold planning, that are specific to legged locomotion. The perception system is crucial to enable the robot to navigate, coping with terrain irregularities and avoiding obstacles.

At the *Istituto Italiano di Tecnologia*, we designed the fully torque-controlled Hydraulic Quadruped robot (HyQ) to perform highly dynamic tasks in difficult terrains; we showed crawling, walking, trotting and jumping capabilities. More recently, we demonstrated some visually assisted trotting, Inertial Measurement Unit (IMU) based balancing, step reflex, and foothold planning on known terrain [1], [2].

In this paper, we present the integration of a dedicated stereo vision-based Simultaneous Localization And Mapping (SLAM) system and terrain modeling on HyQ that allows foothold planning. To extend the robustness of these two modules during navigation, a mechanical motion compensation based on IMU information has also been added. By decoupling camera and body motion, we improved the accuracy of mapping and reduced SLAM failures. We preferred a stereo camera instead of the RGBD camera because it provides a wider field of view, a larger range, and more flexibility for indoor/outdoor applications.

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

[1] Department of Advanced Robotics, Istituto Italiano di Tecnologia (IIT), Via Morego 30, 16163 Genova, Italy. firstname.lastname at iit.it

II. RELATED WORK

Significant progress has been achieved during the last few years in the field of robot perception abilities. In the context of quadruped robots, perception is required for different subsequent tasks and methods such as state estimation, robot SLAM, terrain modeling and classification or object recognition. Only few teams demonstrated the implementation of SLAM on a real system. In most cases they only handle parts of the problem, like doing on board terrain mapping but with the support of external localization or using accurate pre-existing maps while localizing the robot on board. Vision is rarely used on quadrupedal machines. Indeed, such platforms are commonly used to develop low-level controllers, rather than high-level cognitive processes. Furthermore legged locomotion expects precise and failsafe perception capabilities, regardless of difficulties like fast motion, impact shocks, complex visibility that make its use more difficult. However, up to now, few people have worked on the integration of vision sensors on quadrupedal platforms.

Kolter *et al.* [3] presented the most autonomous approach by removing the dependence on given maps and external state input. In their control framework they use a stereo camera with a simple ICP-based technique for point cloud registration to incrementally build a map of the environment. Then, they use a texture synthesis algorithm to fill occluded areas in order to obtain a complete map for the subsequent motion planning of their quadruped, *LittleDog* [4], [5]. While the camera was on the robot, the vision processing and path planning was performed on an external computer.

Chilian and Hirschmuller [6] implemented a multi sensor fusion algorithm by merging inertial measurements, legged odometry, and visual odometry for the DLR Crawler. A semi global matching algorithm for stereo vision was implemented in order to compute an elevation map where the traversability of their hexapod robot could be estimated.

In a similar way, [7] fused the information from stereo vision, legged odometry, and IMU in order to obtain accurate state estimation of their quadruped robot BigDog [8]. On

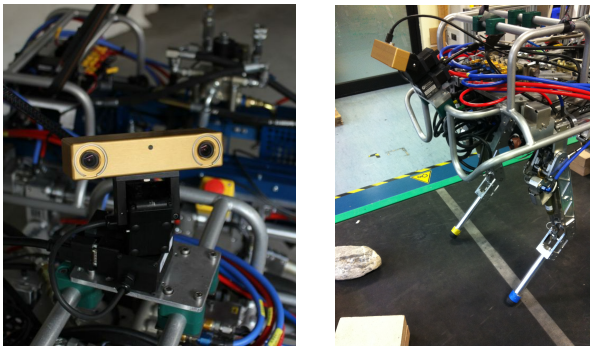

Fig. 1. Pictures of our experimental setup. (a) The HyQ vision setup (b) The stereo camera in the mapping position while the robot is trotting.

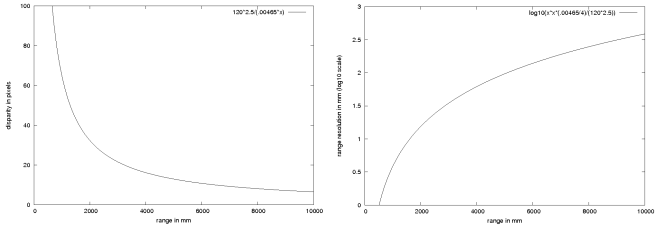

Fig. 2. Illustration of the depth capabilities and the depth resolution of our bumblebee camera. (a) Relationship between depth and disparity (cf Eq. 1), (b) Relationship between depth resolution and depth.

the above mentioned system they also developed obstacle detection algorithms using stereo vision and LIDAR, a registration pipeline and 2D cost map computation which was used eventually for A* based path planning.

Bajracharya *et al.* [9] recently showed terrain mapping for vision-in-the-loop walking on the LS3 robot from Boston Dynamics. The vision system was used to map the environment in the vicinity of the robot and inform the gait generation process about possible changes in the surface where the robot is locomoting. Their main contribution focused on the robustness of the mapping in difficult terrain (vegetation, slopes) and difficult light condition (day or night).

Filitchkin and Byl [10] used a monocular camera to perform terrain classification. The classification was then used to influence the locomotion behavior of their *LittleDog* quadruped. Finally Shao *et al.* [11] presented an obstacle avoidance approach for their quadruped robot that uses a stereo vision-based terrain modeling algorithm.

III. The HyQ robot, and its stereo set up

A. The HyQ robot

HyQ [12] is a versatile hydraulically actuated machine that weighs $80\,\text{kg}$, is $1\,\text{m}$ long and $1\,\text{m}$ tall (Fig. 1.a) and has upper and lower leg segments of $0.35\,\text{m}$ in length. 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. It is equipped with a PC104 for actuation control at $1\,\text{kHz}$.

B. Choice of the vision setup

On HyQ, vision is mainly needed to build a 3D model of the surroundings that is used to compute suitable footholds that allow the robot to overcome obstacles. The main features we have taken into account are:

- the camera height from the ground $1\,\text{m}$
- the resolution greater than the robot foot size ($3\,\text{cm}$)
- the minimum size of the desired 3D model $2\,\text{m}^2$

To obtain a map of the surroundings large enough at this close distance, we had to choose wide angle lenses. Also, we decided to mount the camera on a Pan and Tilt Unit (PTU) to enable some active motion and fixed the camera looking downwards with an approximate tilt angle of $55°$. We also selected the stereo camera characteristics (baseline, image sensor, resolution, frame rate), trying to find the best compromise between accuracy and computation load.

The HyQ vision set up is a *Bumblebee2* firewire colour camera (*Point Grey*) mounted on a Pan and Tilt Unit (*Flir* PTU-D46-17). It is shown in Fig. 1.b.

- The camera has a focal length of $2.5\,\text{mm}$, a wide field of view of $97°$, 2 CCD $1024\,\text{px} \times 768\,\text{px}$ at $20\,\text{FPS}$, a sensor size of $4.80\,\text{mm} \times 3.60\,\text{mm}$, and a $12\,\text{cm}$ baseline. It is accurately pre-calibrated with an accuracy of 0.11 pixel.
- The PTU has a pan range of $\pm 159°$, a tilt range of $-47°/+31°$, a maximum speed of $145\,°/\text{s}$, and a maximum control rate of $60\,\text{Hz}$.

The whole SLAM algorithm runs on a dedicated computer equipped with a quad-core Intel processor at $2.50\,\text{GHz}$ and an NVIDIA GPU GeForce GTX 640.

C. Validation of the camera characteristics

In this section we will present some stereo vision basics to clarify our study of the camera characteristics. For more details on getting depth from stereo vision, the reader can refer [13]. Two images with slightly different viewpoints show the same object in different positions; the distance between the two is called disparity. Given a disparity value D, the 3D coordinates of the matched pixels in the two stereo images in m are computed using the projective camera equations: Eq. 1

$$Z = b\frac{f}{D}, \qquad X = u\frac{Z}{f}, \qquad Y = v\frac{Z}{f} \tag{1}$$

where b is the baseline, f is the focal length, and D is the disparity (in m), X, Y, Z are the coordinate of the object, and u and v are the location in the depth image (see Fig. 3, camera frame, image frame). All u, v, f and D are in pixels in Eq. 1.

Given a change in the disparity, the smallest change in depth that is detectable by the stereo geometry is the depth resolution r:

$$r = D\frac{Z^2}{b\,f} \tag{2}$$

Fig. 2.a shows that the minimum depth for this stereo camera is $0.65\,\text{m}$: at this point, the disparity is over $100\,\text{px}$.

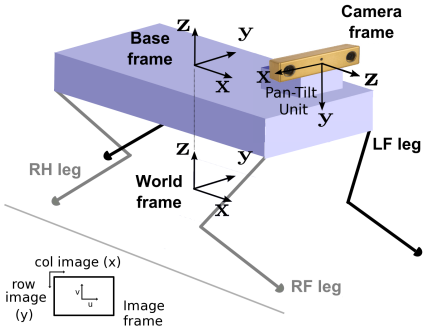

Fig. 3. Definition of HyQ reference frames: image frame, camera frame, base frame and world frame.

The maximum depth is above $40\,\text{m}$, but most of the change in disparity takes place in the first few meters. In our case, because we focus on building an accurate map in front of the robot's forward feet and we also want to reduce the computational load as much as possible, we set the stereo matching search window of disparities from $10\,\text{px}$ to $74\,\text{px}$, which corresponds to a depth restricted to the interval $0.8\,\text{m}$ to $4\,\text{m}$.

Fig 2.b shows the depth resolution r plotted on a log10 scale. At $1\,\text{m}$, the depth resolution is about $4\,\text{mm}$ (assuming a system without video noise or matching errors). At $2\,\text{m}$, it has grown to $15\,\text{mm}$ and at $3\,\text{m}$ is $35\,\text{mm}$. This resolution justifies our decision of building a height map with minimum $5\,\text{mm}^2$ cells. It has to be noted that depth resolution is not the same as depth accuracy, which measures the difference between the depth computed and the actual depth. Depth accuracy is sensitive to errors in camera calibration, which is certified to be good on this fixed pre-calibrated stereo camera. Finally, the small focal length gives us a large field of view (about $97°$ and allows to cover $2\,\text{m}^2$ surface at $1\,\text{m}$ of distance that ensures a good view on obstacles in front of the robot for foothold planning.

IV. On board SLAM with a stereo camera

SLAM is the process of incrementally building up a map within an unknown environment (without *a priori* knowledge), while at the same time keeping track of the current location of the robot. To implement this process we used a stereo camera and a code based on the Kinect Fusion algorithm for large scales [14]. The developed SLAM system, the terrain modeling and the path planning modules are described in Fig. 6.

A. Depth map computation

1) Disparity computation: The rectified images are extracted with the Point Grey proprietary library *Triclops* and the computation of the disparity for each pixel is performed using the Sum of Absolute Differences (SAD) method [15] on 4 threads. We used the SAD correlation based method because of its small computational load, that is suitable for real-time implementations. The disparity is computed on edges images, to allow matching on the changes in brightness

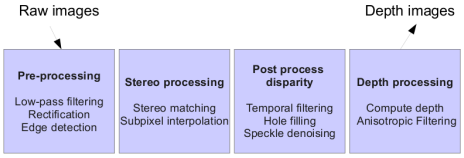

Fig. 4. Depth map computation flowchart.

rather than on the absolute pixel intensities. This approach is more robust in environment with variables light conditions, like outdoor settings, where our robot is intended to operate. In order to obtain a more accurate disparity we use a sub-pixel interpolation filter [16] on the disparity map. The filter attenuates the quantization effect and makes surfaces smoother (the disparities are interpolated up to 1/4 sub-pixel of accuracy).

2) Filtering: The SAD correlation method computes the similarity between pixels by comparing windows around pixels of interests. This method is fast but yields to several artefacts: in some regions disparities are uncertain and are left as gaps in the disparity map; this is mainly due to insufficient texturization of the surfaces or noise. To overcome this, we used different fast filtering methods to remove outliers and fill in holes. It has to be noted that only these filters are applied, no uniqueness, surface or texture check are applied. The filter we apply is the following:

```
for every pixel i in disp_t[]
    if disp_t[i] = 0xFF00 (outlier)
    then
        if disp_t-1[i] != 0xFF00
        then disp_t[i] = disp_t-1[i]
        else disp_t[i] = median of the
             3x3 neighborhood of pixel i
        end
    end
end
```

It uses the spatial and temporal information to fill in the disparity. Every pixel without disparity values is checked and small holes are filled. Then, outliers are removed by applying a speckle filter (see [17]). It removes spikes characteristic of mismatches in correlation.

As a reference, we compute on our dedicated computer the $640\,\text{px} \times 480\,\text{px}$ disparity at $15\,\text{Hz}$ and then we extract the depth. The depth map is the pixel image that holds depth values, *i.e.,* distances from the camera to the 3D scene points.

B. Simultaneous Localization and Mapping

For the SLAM, we use a modified version of the Kinect Fusion (KinFu) Large Scale algorithm from PCL [14]. KinFu Large Scale is an extension of KinFu for building up larger maps. It performs high-quality reconstruction of geometrically accurate 3D models in real-time. KinFu was natively built for Kinect, it directly works with the depth map and performs surface reconstruction, which approximates more accurately the geometry of real world than point-based representations. It is also optimized for GPUs, which allows

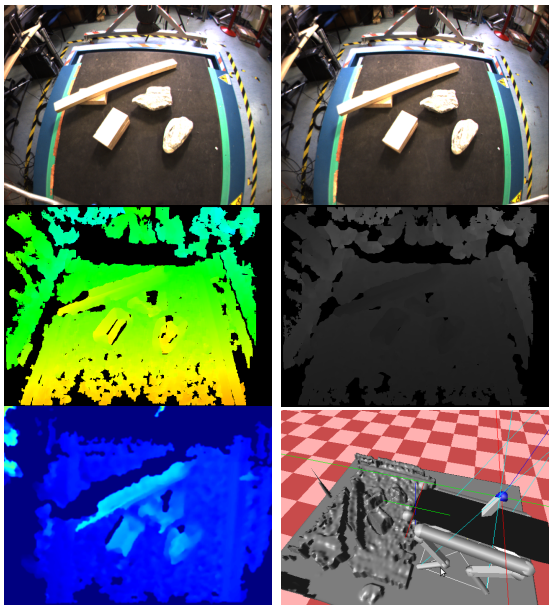

Fig. 5. Different steps of the height map building framework. (a) right image, (b) left image, (c) post processed disparity, (d) depth image, (e) height map built from one single point cloud, (f) terrain model and the robot in the simulation environment.

all the computation to be done in real time with a sufficient accuracy.

As the stereo vision module provides a depth map, we used it to feed the KinFu algorithm. The following sections describe the algorithm in more details.

1) Depth Map conversion: A depth map is converted into a 3D point cloud by using the calibration matrix (see Eq. 1). The result of this step is a point cloud with vertex and normal data for each point at three different levels of detail.

2) Pose Estimation: A modified ICP algorithm performs the alignment of two surfaces making the assumption that they are sufficiently close to each other. The ICP iterations are performed with three different resolutions and generate a six degrees of freedom transformation matrix that aligns the current point cloud with the previous ones.

3) Surface reconstruction and meshing: Ray casting is used for this surface reconstruction step. A prediction of the current global surface is obtained, with vertex data and estimated normal data. The refined ray-casted model is the same used in the next ICP step for alignment. By doing this, instead of using just the last frame point cloud as the source for alignment, a less noisy model is obtained.

C. Pan and tilt motion compensation

As this robot is subject to considerable motion during locomotion and since the ICP assumes that point clouds are close together, we added a motorized motion compensation system with a Pan and Tilt Unit based on the IMU information provided on the robot. This approach prevents that this assumption is broken, improves the SLAM results and reduces failures. To control our PTU motors we choose to implement a PD controller, that has been used to control

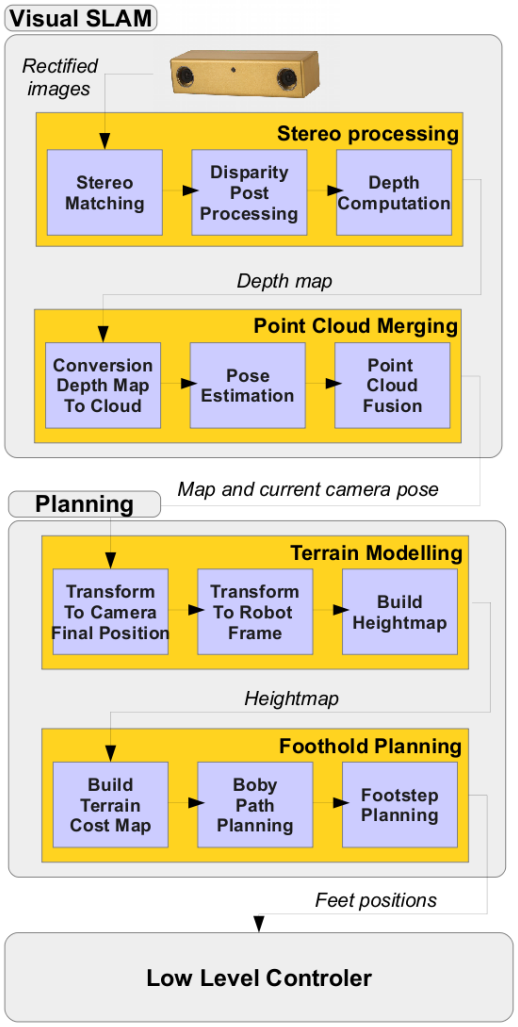

Fig. 6. Flowchart of the whole software developed.

most active head systems. For each axes, pan and tilt, we close the feedback loop in position. More details on the motion compensation controller and on the improvement of the SLAM can be found in [18].

V. From the map to the robot path planning

Since we transformed all the point clouds into the same reference frame (*i.e.,* camera frame in Fig. 3). The goal is to build a full map of the terrain for foothold planning. Consequently the merged point cloud needs to be transformed in the robot frame, and the pose estimation needs to be used to position the robot in the map.

A. Calibration

To transform the map into the robot frame we need the full transformation between the camera frame and the robot frame. As the robot setup often changes — due to repairs, installation of new parts and sensors, or simply because of a change of application — we developed a module to compute automatically this transformation with high accuracy.

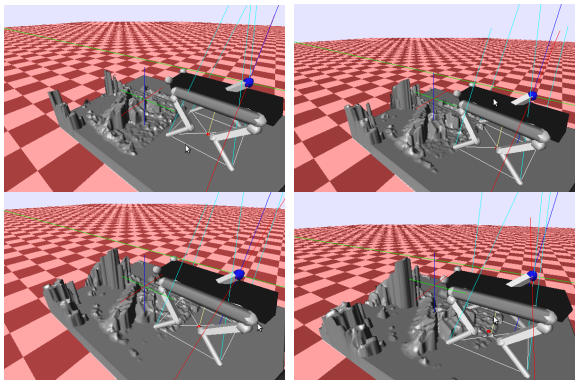

Fig. 7. The robots at different time stamp localized in its map while trotting. The robot has roughly moved 10 cm forward between each image.

The method consists in tracking a marker on the right front foot (see Fig. 1). As we know the position of the marker in the robot frame (thanks to the leg encoders and forward kinematics) and their positions in the camera frame (thanks to the visual tracking), we can estimate the transformation between the two.

1) Tracking of markers: For the tracking we use the color information with the Mean Shift algorithm. The method is easy-to-use, since there is no need of learning stage or parametrization. We implemented a modified version of the CAMShift algorithm [19]. This method sends at 20 Hz the position of the tracked marker inside the two stereo images.

For each tracker position computed in the camera frame, we associate the position of the tracker in the robot frame. A sample used for the calibration is defined as: $u_r v_r u_l v_l x y z$ where u_r and u_l are the row position (in pixels) of the barycenter of the marker in the right and left image respectively, and v_r and v_l are the column position (in pixels). Then x, y, z is the 3D position (in meters) of the marker in the robot frame (Fig. 3).

Note that the calibrated transformation of the camera is also expressed in robot coordinates.

2) Calculation of calibration matrices: For the camera calibration, we developed a flexible method that allows us to calibrate any kind of camera. We calibrate the position and rotation of the camera, as well as the intrinsic parameters. We chose a pinhole camera with Brown's distortion model [20]. The parameters that define our camera are:

- position and rotation: x, y, z, roll, pitch, yaw
- focal length and skew factor: f_x, f_y, s
- principal point and lens center: x_0, y_0, l_x, l_y
- 10 distortion parameters

Given a 3D point in space and using the above mentioned camera model, we can calculate the position of this point in the camera frame. Since we are tracking the point we know what is its actual position of this point in the image. The calibration error is defined as the sum of the squared distance of the predicted and measured marker position in the camera frame. By using a multi-variate Newton Raphson method we can iteratively minimize the error and find the camera position, rotation and internal parameters. We carried out several calibrations and we got an average error below one pixel in less than 1000 iterations.

B. Transformation to world frame and height map

The merged point cloud is obtained in the initial camera frame. It is necessary to transform it to camera final position and next to the robot frame in order to be able to perform foothold planning. Eq. 3 represent this transformation.

$$X_f = P^{-1}\, C\, X_i \tag{3}$$

where P is the pose that is to say the final camera position in the initial camera frame and C the matrix representing the camera initial position in the robot frame.

Since we have of the map in the robot frame, the point cloud is projected in 2D where all the pixels of the image corresponds to a ($5\,\text{mm}^2$/px) surface, and finally some filtering is performed to remove the outliers, fill holes and improve the height map quality. Again we use the filter presented in Section IV-A.2 followed by an edge preserving smoothing (anisotropic filtering). Fig. 7 shows the robot in its maps in our simulation environment.

We chose to use a height map as a world model shared between the SLAM and planning modules because of it is efficient in terms of computation, analysis and storage space, and accuracy to perform foothold planning.

C. Terrain cost map for foothold planning

Height maps provide morphology information as 2D images whose pixel represents the vertical distance of a finite squared cell from the ground level. A straightforward approach to use such type of information is to create a map that associates a cost to each cell extracted from the height map and then compute the most suitable sequence of footholds that leads to the goal. The cost map penalize:

- High frequencies (*i.e.,* discontinuities on the morphology, such as the edges of a rock);
- Small uniform areas (*e.g.,* holes or small flat rocks), since the robot has a non negligible foot area and it could miss such foothold;
- The distance to the robot nominal leg position (56 cm). The maximum extension and the the minimum retraction are respectively 10 cm and 20 cm.

To achieve these goals, we extract information from derivatives of the image and then we assign a cost that depends both on the direction and the intensity of those derivatives. The cost map computation involves three steps:

- Edge detection: the edges of a height map reflects the actual discontinuities of the ground;
- Morphology operator: since the foot has a finite area, we want to avoid also areas around edges;
- Gaussian filter: the cost decreases proportionally with the distance from the computed edges.

The final cost map is then linearly combined to the original height map to take into account the height of the cells (Fig. 8.b). In this figure, the values of the colormap vary between dark cold colors for lower costs and dark warm

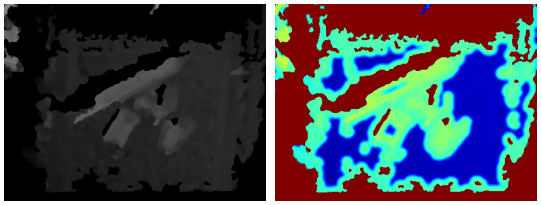

Fig. 8. Conversion of the height map (a) to a cost map (b).

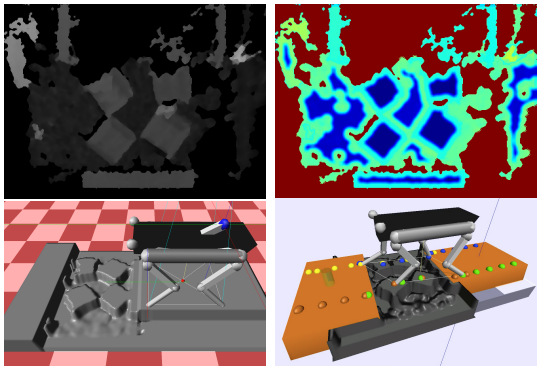

Fig. 9. Foothold planning results using an height map. (a) height map (b) cost map (c)(d) simulation environment without and with the footholds.

colors for higher costs. In particular, we compute the absolute difference between the robot ground level and the height map (for example the stepping stones in Fig. 9.b are darker than the ground because they are at the same level of the robot). Unknown heights where treated as special case by assigning them with the maximum cost.

VI. Experiments on foothold planning

To experimentally validate our map building pipeline we use our foothold planner presented in [1]. This path planning framework compute from a cost map the appropriate footholds to overcome challenging terrain like stairs or stepping stones. It is based on *virtual model* based controller that guarantees the overall compliant behavior of the system. It has been demonstrated using maps a priori built with a Kinect and the robot execute blindly its planned trajectory. We refer the reader to [1] for more details about the foothold computation and the controller.

In this paper, we used the same algorithm but a map built online, directly. It has to noted that dynamic re-planning is not performed. The sequence of foothold is computed by the robot just before starting moving (Fig 9.c). Next when the foothold sequence has been accomplished, we can update our position and compute new footholds. The results show the effectiveness of our map building pipeline in order to compute footholds.

VII. Conclusions

In this paper we propose a real-time SLAM solution using a pan and tilt stereo camera mounted on an hydraulically actuated quadruped robot that builds a map and keeps track of the robot's position. We validated the choice of the stereo vision set up, we detailed our stereo vision framework and presented some results of path planning using the previously built map. Experiments of locomotion on our hydraulically actuated quadruped robot were successful with all the computation on board thanks to dedicated implementation based on fast stereo depth computation, GPU based map building and mechanical motion compensation. This localization and mapping ability allows to perform navigation on rough terrain in a fully planned manner.

Our future work will be to integrate the online dynamic re-planning to compensate the drift in case of slippage, improve the accuracy of the map and reduce SLAM failures by integrating the IMU measurement in the SLAM pipeline. Finally we would like to improve the disparity computation rate by adding FPGA computation, to relieve the vision computer, that today is fully used for stereo computation.

References

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] A. Chilian and H. Hirschmuller, "Stereo camera based navigation of mobile robots on rough terrain," in *IEEE/RSJ IROS*, 2009.

[7] 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.

[8] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *IEEE/RSJ IROS*, 2008, pp. 3946–3952.

[9] 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.

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

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

[12] 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.

[13] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge Univ Press, 2000, vol. 2.

[14] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE ICRA*, 2011.

[15] K. Mühlmann, D. Maier, J. Hesser, and R. Männer, "Calculating dense disparity maps from color stereo images, an efficient implementation," *Int. J. of Computer Vision*, 2002.

[16] R. Szeliski and D. Scharstein, "Symmetric sub-pixel stereo matching," in *ECCV 2002*. Springer, 2002.

[17] Z. Shi and K. B. Fung, "A comparison of digital speckle filters," in *Geoscience and Remote Sensing Symp. Surface and Atmospheric Remote Sensing: Technologies, Data Analysis and Interpretation., Int.*, vol. 4, 1994, pp. 2129–2133 vol.4.

[18] F. Rovida, "Development of an active head for the hyq robot," 2014.

[19] G. Bradski, "Computer video face tracking for use in a perceptual user interface," *Intel Technology J.*, 1998.

[20] D. C. Brown, "Decentering distortion of lenses," *Photometric Engineering*, 1966.