Dynamic Locomotion Terrain Recognition for the MIT Cheetah 3 Robot
6.869 / 6.819: Advances in Computer Vision

Gerardo Bledt
Massachusetts Institute of Technology
Cambridge, MA
gbledt@mit.edu, 6.869

Milo Knowles
Massachusetts Institute of Technology
Cambridge, MA
mknowles@mit.edu, 6.819

Abstract

Our goal in this project is to create an inexpensive, lightweight and robust vision system that can estimate the terrain in front of the MIT Cheetah 3 robot in real time. Using a stereo rig consisting of two $20 webcams, we are able to obtain metrically accurate 3D reconstructions of planar obstacles from image pairs. In addition, by fitting a polytope to noisy terrain point clouds, our representation of the terrain is robust to outliers and has a very compact parametrization. This allows real time communication of the sensed enfironment to the robot, which allows it to modify its trajectory plan according to the vision input.

1. Introduction

Currently, the MIT Cheetah 3 robot uses primarily completely blind locomotion. What this means is that there is no vision or LiDAR system that allows it to preemptively sense its environment or plan for upcoming obstacles. Dynamic gaits such as trotting have been stabilized under ideal flat-terrain conditions, but most interesting applications of robotics involve some kind of unstructured, non-flat, non-smooth terrain situations. Recent research efforts have allowed for reactive tactile environment sensing where the robot is able to interpret and build its knowledge of the environment when it senses unexpected contact with a surface [1]. This has allowed for the same nominal dynamic gaits to stabilize in rough terrains by reacting to the estimated foot contacts.

However, since this environment sensing is purely reactionary, the robot can only detect objects and uneven ground after it has already determined contact with the surface. This covers a lot of cases where small objects and holes are littered over terrain with a slowly changing gradient. A current challenge is when the robot encounters a large, sharp terrain disturbances, such as a box or a set of stairs, it will not be able to smoothly use this reactionary sensing to navigate the terrain. If the box or stair is higher than the nominal foot swinging height as it is in Figure 1, it will simply not be able to step over it and will not detect a support surface contact when it does hit the side. Similarly, if the box or stair is higher than its possible foot swing height, it will not be able to continue forward and crash. By incorporating some visual terrain recognition, the robot will be able to modify its swing foot trajectory to clear objects that are within its kinematic workspace, as well as feed terrain information to the contact detection algorithm for more accurate probabilistic ground height models.

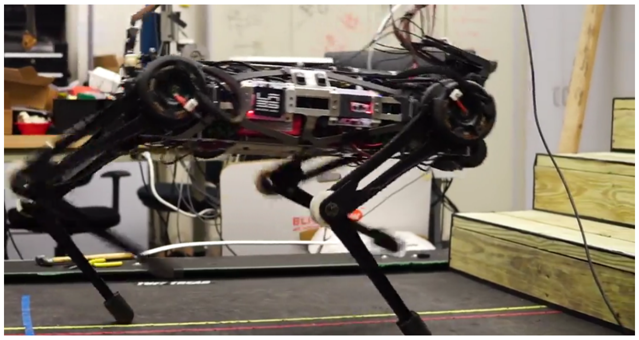
Figure 1. A robust controller is used for general robot locomotion in unstructured terrains, but a method of external environment sensing is needed for the robot to know that its nominal foot swing trajectory will not be able to clear the stair obstacle.

2. Related Work

In robotics, external environment sensing is an important problem that many robots must deal with. Many robots use LiDAR, but find that it generates a lot of data and takes too long to compute. Many others have shown very promising results using online stereo vision for terrain mapping when [5, 2]. As a future step, the stereo reconstruction may also be used as ground truth data to give the robot an accurate global positioning.

There are many stereo vision algorithms that are currently being used as described in [3]. This provides a wide survey of different techniques used for reconstruction of 3D information from stereo cameras. These methods outlined, as well as the successful results in robotics expressed the validity of using stereo reconstruction for terrain mapping during dynamic locomotion.

3. Approach

Our pipeline is summarized in Figure 2. We begin with a pair of RGB stereo images. From this image pair and calibration parameters obtained offline, we apply a stereo matching algorithm (See Section 3.1 to get a depthmap of the scene. Using camera instrinsic parameters, we can project this depthmap into 3D space and obtain a point cloud of the environment (See Section **??**). Using two different methods described in Section 3.2, we decompose this point coud into a set of planes. Finally, we reconstruct a polytope from these planes. A robot can use this geometric representation of the terrain to plan its foot trajectory.

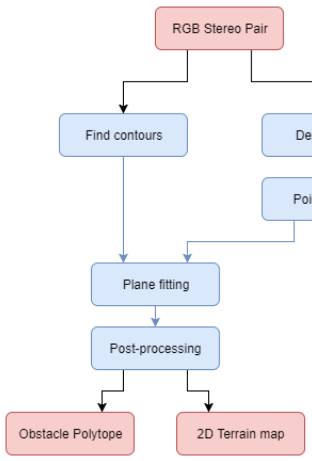

Figure 2. An overview of our obstacle reconstruction pipeline.

3.1. Stereo Matching for 3D Pointcloud

The goal for stereo vision is to take two separate pictures of the same scene from two cameras at known relative distances and orientations form each other and attempt to reconstruct the 3D version of the scene. This is the general principle behind the human vision system that has depth perception. In our experiment we set two parallel cameras about 8 cm apart in the same direction which results in pictures like the one in Figure 3.

In order to find the best match between the two images, we used a technique called Semi-Global Matching as described in [4]. The basic premise follows an optimization that compares the two images with a matching cost defined

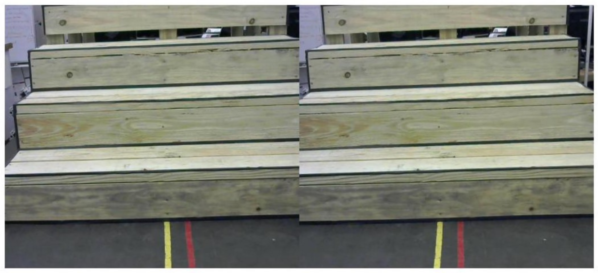

Figure 3. Am image pair taken by our stereo camera setup. The two cameras are parallel and have a baseline of approximately 8cm.

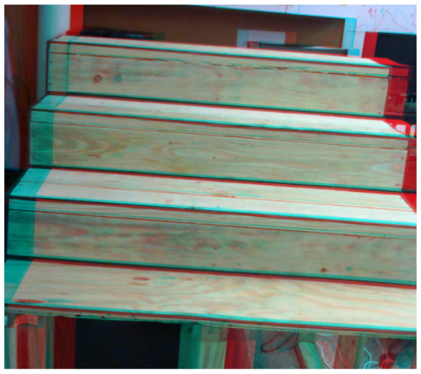

Figure 4. An example stereo anaglyph.

by the following cost function

$$E(D) = \sum_{p}(C(p, D_p) + \sum_{q \in N_p} P_1 T[|D_p - D_q| = 1] + \sum_{q \in N_p} P_2 T[|D_p - D_q| > 1])$$

Here, the optimization travels along many paths throughout the image and the costs from each path are summed and the lowest cost disparity is chosen for each pixel. The cost function has the first term being the sum of the pixel matching costs for the disparities, the second term being a penalty for 1 pixel disparity, and the third being a large cost for bigger pixel disparities.

The method results in a disparity image that gives the estimated depth to each pixel. This disparity is then used to create a 3D representation of the scene since we know the depth at each 2D pixel. This results in a pointcloud with Z being the distance from the camera locations. The original image in then overlaid onto the pointcloud as seen in Figure 5. Although the pointcloud is not an exact representation of the actual scene, many post-processing steps were taken to smooth out the noise and extract features from the pointcloud.

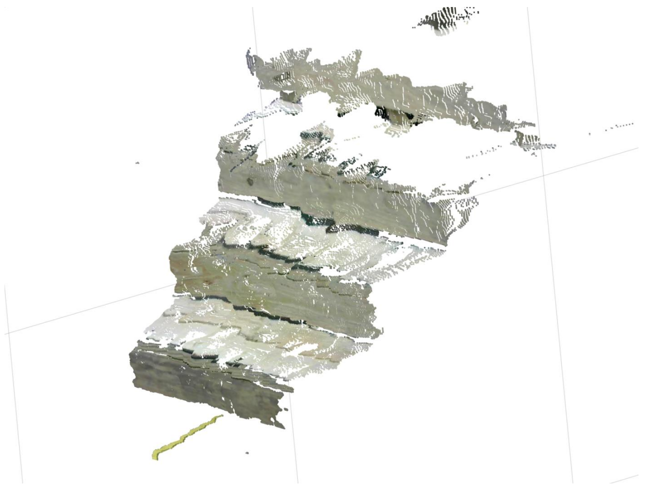
Figure 5. A pointcloud representation of the stairs in Figure 3.

3.2. Planar Decomposition of Point Clouds

In order to estimate a polytope representation of an obstacle, decompose its point cloud into a set of planes. We demonstrate two main approaches for finding planar surfaces of an obstacle. We describe the Contour Method in Section 3.2.1 and the Orientation Method in 3.2.2.

Each method has its advantages and limitations. While the Contour Method can generalize to planar surfaces with arbitary orientation, it is highly sensitive to the texture and edges of obstacles. The Orientation Method is more robust to appearance, but is limited to vertical and horizontal planes.

In both approaches, MSAC (M-Estimator SAmple Consensus [7]) is the algorithm used for plane estimation. In [7], Zisserman et. al. show that MSAC is more robust than RANSAC for estimating geometric relationships in point cloud data, at no additional computational cost.

3.2.1 Contour Method

As the name of this approach suggests, we detect large contours in the original RGB image and project these contours into the point cloud to find subsets points that are likely to be coplanar.

We use MATLAB's *imcontour* function from the Computer Vision Toolkit to efficiently find contours. Internally, this function uses the Theo Pavlidis algorithm [6] to find contours at a variable number of levels.

Contours are then sorted by area, and pairs of contours with Intersection-over-Union (IoU) above a threshold of 0.3 are replaced by the larger of the two contours. Highly overlapping contours are likely to lie on the same plane, and we observe empirically that our reconstruction performs better when there are no duplicate planes.

We then project our candidate contours into the point cloud, and find 3D points that originally from inside of this contour in the RGB image. A plane is fit to each planar subset of the point cloud using MSAC.

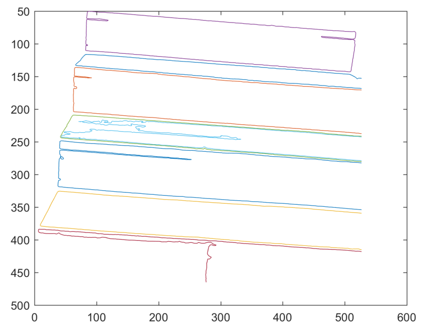

Figure 6. An example contour plot of the stairs in Figure 4. Here we have shown the eight largest contours in the image by area.

3.2.2 Orientation Method

The Contour Method, as one might expect, is highly sensitive to lighting effects and the sharpness of edges on the obstacle. In this section, we propose a second method that relies on *au priori* knowledge of plane orientation to remove our dependence on image contours.

In this method, we assume that the obstacle is composed of a predetermined number of planes that are either horizontal and vertical relative to the world frame. While performing MSAC, we sample three points from the point cloud and compute the parameters of a plane with them. We require that the normal of this plane is within an angular tolerance of a reference normal vector (i.e the z-axis for vertical planes and the y-axis for horizontal planes). If the robot frame is not aligned with the world frame, we can rotate our reference normals before performing MSAC. After performing a fixed number of iterations, MSAC selects a horizontal or vertical plane with the lowest cost. We then remove the inlier points from the point cloud, and iteratively fit horizontal and vertical planes to the residual point cloud.

Although our algorithm currently looks for a fixed number of planes, it would be straightforward to add a condition that stops extracting planes once the residual point cloud is too small, or the cost of MSAC planes is above a threshold.

3.3. Polytope Reconstruction from Unordered Planes

At this stage in the pipeline, we have an unordered set of planes, each represented by four parameters: a surface normal and offset constant. There are many ways that these planes could intersect to form a polytope, but only one that actually matches our ground truth observation. We must make another assumption here about the composition of the polytope.

When using the Contour Method in Section 3.2.1, we assume that the depth and height ordering of planes in 3D space is given by the vertical ordering of planes in the RGB image. Horizontal planes that appear higher in the RGB image are also higher in the scene. Similarly, vertical planes that appear higher in the RGB image are deeper in the scene.

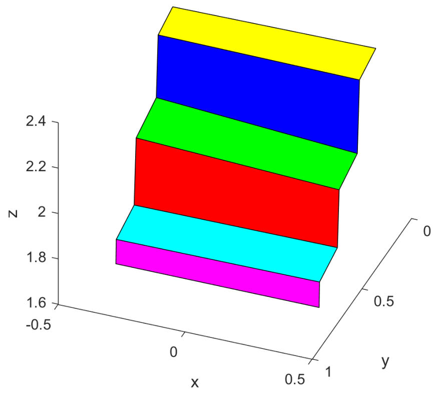

Figure 7. A polytope reconstruction of the image in Figure 3, using the Contour Method (See Section 3.2.1). The pink, red, and blue planes correspond to horizontal faces of the stairs. The light-blue, green, and yellow planes are vertical faces.

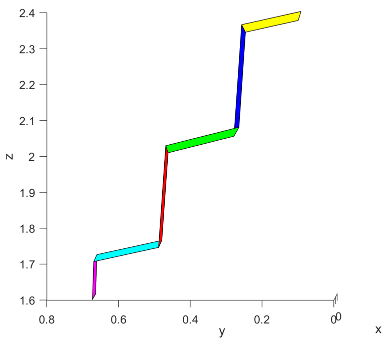

Figure 8. The polytope from Figure 7 viewed from the side. The vertical faces of this polytope (light-blue, green, yellow) are very close to the groundtruth dimension of 20cm. Similarly, the remaining horizontal faces are close to their groundtruth dimension of 31cm.

When using the Orientation Method in Section 3.2.2, we use a sampling based approach to order planes. First, we sort all vertical planes based on their distance from the camera, then find the horizontal plane that is "active" between each pair of vertical planes. The "active" horizontal plane is the one that has the smallest average distance to a sample of points in the region between the vertical planes.

Finally, once an ordering of planes has been established, we can compute lines of intersection between these planes,

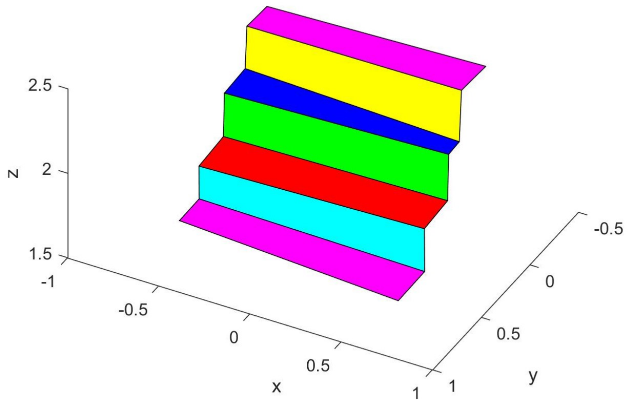

Figure 9. A polytope reconstruction of the image in Figure 3, using the Orientation Method (See Section 3.2.2). Here, the pink, red, and blue planes are vertical, and the light-blue, green, and yellow are horizontal. Because the orientation method is not limited by image contours, we are able to construct a polytope with more faces than in Figure 7

.

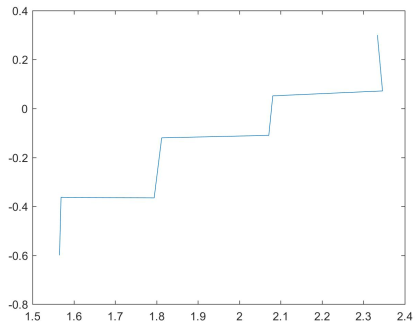

Figure 10. The polytope from Figure 9 viewed from the side. Again, we see that vertical and horizontal planes are close to the groundtruth dimensions.

and obtain a set of vertices for the polytope. See Figures 7 and 9 for example reconstructions. See Figures 8 and 10 for a metric comparison of our reconstructions to groundtruth.

4. Results

4.1. Testing with Other Obstacles

To demonstrate the robustness of our algorithm, we test it on another set of stairs with a smooth texture and a different number of steps. Despite these differences, we still obtain a reasonable polytope model of the stairs using our algorithm with the Orientation Method (See Section 3.2.2) and the exact same parameters that we used to build the polytope in Figure 9. Results could likely be improved by dynamically choosing parameters such as the number of horizontal and vertical planes, the reference normal tolerance, and the number of MSAC iterations online.

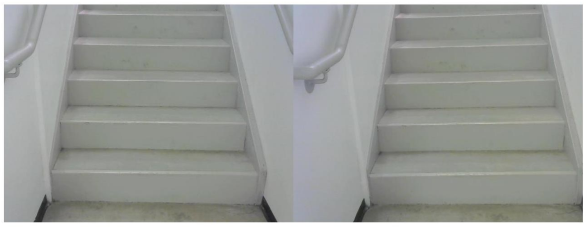

Figure 11. A set of stairs from Simmons Hall. Notice that these stairs have less texture and smoother edges than the stairs from Figure 3, causing the Contour Method to perform poorly.

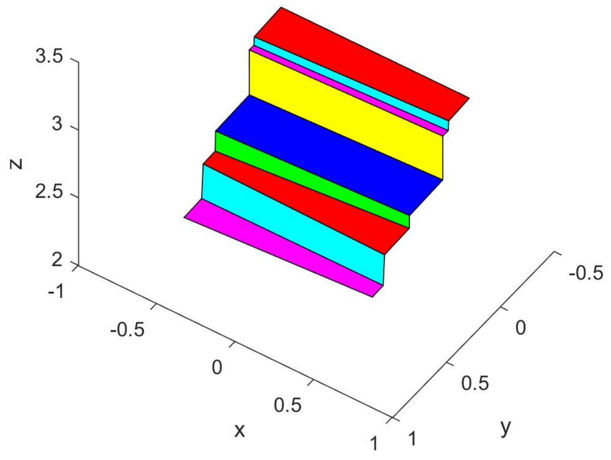

Figure 12. The final polytope reconstruction of the Simmons stairs.

4.2. Testing with the Cheetah 3

Since the robot previously used only reactive sensing of the environment this meant that the only way for the robot to sense obstacles and terrain was to physically interact with them. A robust reactive controller has been achieved and allows the robot to traverse highly unstructured terrains while remaining stable. However, while this makes the robot robust, there are situations where the terrain is not traversable easily without prior knowledge of the environment. Adding the vision system to the overall control system as an input to the path planning as seen in Figure 13 will allow the robot to know if an obstacle is present, where it is, and its dimensions. Then it can decide how to modify its locomotion plan accordingly. For purposes of this demo the robot will attempt to detect the stair height and distance and modify its swing leg trajectory to place it on the stairs as its nominal swing foot height is lower than an average step.

It is not enough to just find the dimensions of the object in front of the robot. This doesn't have any physical meaning to the robot until we find the relationship between the

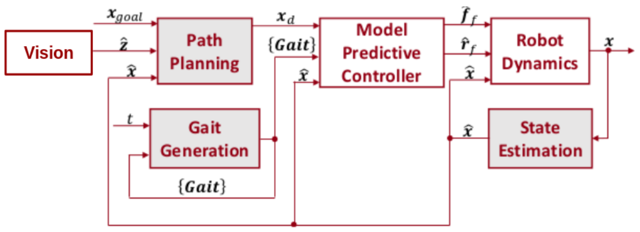

Figure 13. .

reconstructed object to the robot's coordinate frame. We take the points of interest of the object in the 3D camera frame, ${}^{\mathscr{C}}p_o$, and apply the following transformation

$$
{}^{\mathscr{I}}p_o = {}^{\mathscr{I}}H_{\mathscr{C}}{}^{\mathscr{C}}p_o \tag{1}
$$

where ${}^{\mathscr{I}}H_{\mathscr{C}}$ is the homogeneous transformation matrix between the camera frame and the Inertial frame. This allows the robot to know the distance and angle to the stairs relative to its body as seen in Figure 14. With this, the robot now has knowledge about its environment without the need to physically interact with it. With the robot holding an internal model of the environment with respect to its state, the path planner can adapt the nominal gait and foot placement trajectory to deal with the stairs at the detected height. The foot trajectory is modified when the robot's center of mass has traveled close enough to the detected stairs

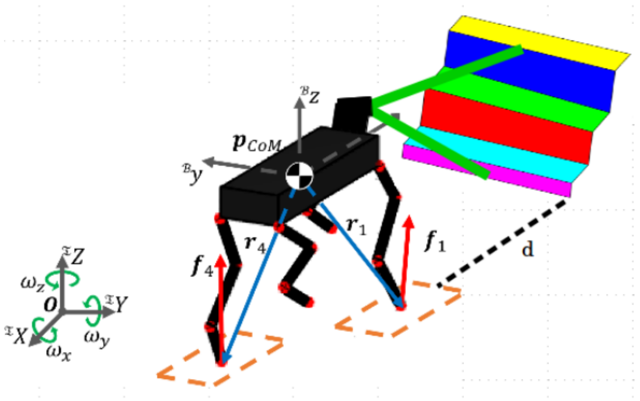

Figure 14. The robot must transform the coordinates of the stairs in the 3D reconstructed camera frame into the world frame in relation to the robot.

$$
({}^{\Psi}p_{CoM,x} - {}^{\Psi}p_{0,x}) < (d - d_{buffer}) \tag{2}
$$

where ${}^{\Psi}p_{CoM,x}$ is the yaw rotated CoM position ignoring pitch and roll effects, ${}^{\Psi}p_{0,x}$ is the yaw rotated position at the last vision input, d is the distance to the object at the detection time, and d_{buffer} is a user defined buffer distance so that the robot begins lifting the legs slightly before the actual stair.

The vision system was successfully integrated with the robot software pipeline. Testing was first done in simulation where the webcams were used to detect the stairs and the robot was walked forward in simulation. The nominal trotting swing height for the leg trajectories is about 15 cm and the stair height was 21 cm. Therefore, the path planner made the decision to raise the height of the trajectory to about 30 cm, which was approximately 10 cm over the detected object height. When it reached the location of the 3D reconstructed stairs, the robot modified its foot trajectory appropriately.

The pair of cameras was mounted on the physical robot and experiments were run with the real stair set on a treadmill. Figure 15 shows the robot successfully climbing the stairs on its first attempt. With the terrain-blind controller, the most common failure mode was the robot being unable to get its leg over the stair and tripping. With the vision input, this was no longer the case as it knew how high to lift its leg and where to place the feet to aquire an adequate foothold.

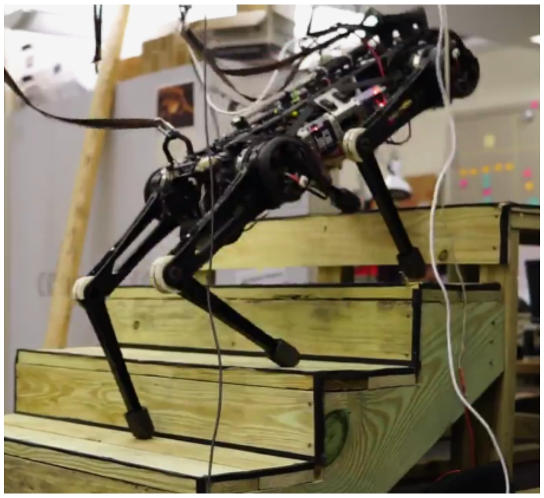
Figure 15. With the vision input, the robot is able to stably figure out how high to raise the feet in order to place them onto the stairs and successfully climb up.

5. Conclusion

The presented vision system yielded promising results for use as a general terrain sensing solution for the MIT Cheetah robot during locomotion. Successful experiments showed its ability to accurately detect the location and dimensions of the stairs. Terrain information being provided to the robot helps the robot's path planning algorithms make better decisions when facing obstacles. Stability is improved as it anticipates contacts more accurately, rather than blindly relying on its reactive interactions with the physical world.

However, several practical limitations were noted when implementing the system. First and most importantly, the results were extremely sensitive to the camera calibration. As soon as the cameras moved slightly from the position they were calibrated in, the algorithms would return poor reconstructions. Similarly, if there was a slight lag between the frames that were taken on each camera, it would essentially act as if the cameras had moved with respect to each as the robot moves. The stereo algorithm also relies on having enough distinct features for disparity between the images. The work presented is a good first step towards robust robot vision, but more work would still be needed in the future. Overall, the system was successful in allowing the robot to climb stairs reliably under the experiment conditions.

6. Individual Contribution

References

[1] G. Bledt, P. Wensing, S. Ingersoll, and S. Kim. Contact model fusion for event-based locomotion in unstructured terrains. *(submitted) 2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[2] M. F. Fallon, P. Marion, R. Deits, T. Whelan, M. E. Antone, J. McDonald, and R. Tedrake. Continuous humanoid locomotion over uneven terrain using stereo fusion. In *Humanoids*, pages 881–888. IEEE, 2015.

[3] R. A. Hamzah and H. Ibrahim. Literature survey on stereo vision disparity map algorithms. *J. Sensors*, 2016:8742920:1–8742920:23, 2016.

[4] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814 vol. 2, June 2005.

[5] J. Z. Kolter, Y. Kim, and A. Y. Ng. Stereo vision and terrain modeling for quadruped robots. In *2009 IEEE International Conference on Robotics and Automation*, pages 1557–1564, May 2009.

[6] T. Pavlidis. Algorithms for graphics and image processing. 1982.

[7] P. H. S. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 2000.