2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Detecting Usable Planar Regions for Legged Robot Locomotion

Sylvain Bertrand[1], Inho Lee[1], Bhavyansh Mishra[1,2], Duncan Calvert[1,2], Jerry Pratt[1,2], and Robert Griffin[1,2]

***Abstract*— Awareness of the environment is essential for mobile robots. Perception for legged robots requires high levels of reliability and accuracy in order to walk stably in the types of complex, cluttered environments we are interested in. In this paper, we present a usable environmental perception algorithm designed to detect steppable areas and obstacles for the autonomous generation of desired footholds for legged robots. To produce an efficient representation of the environment, the proposed perception algorithm is desired to cluster point cloud data to planar regions composed of convex polygons. We describe in this paper the end-to-end pipeline from data collection to generation of the regions, where we first compose an octree in order to create a more efficient data representation. We then group the leaves in the tree using a nearest neighbor search into a planar region, which is composed of the concave hull of points that is decomposed into convex polygons. We present a variety of environments, and illustrate the usability of this approach by the Atlas humanoid robots walking over rough terrain. We also discuss various challenges we faced and insights we gained in the development of this approach.**

I. INTRODUCTION

Environmental knowledge and awareness remains a major challenge to creating robots that are capable of autonomously navigating in their environment. While recent years have seen tremendous progress driven by both the academic community and, the autonomous vehicle field, there is still much progress to be made. Enabling legged robots to navigate rough terrain poses a relatively unique perception challenge, as there are distinct requirements often different from those encountered by wheeled, tracked, or aerial platforms, who are primarily concerned with obstacle detection.

Lower-dimensional representations of spaces such as planar regions are often more tractable for other algorithms like footstep planning [1]. The problem of footstep planning is subtly different from the task of path planning for wheeled or tracked robots, where the main goal is to identify potential collisions and the *absence* of space, such as gaps and holes, with small features such as slight discontinuities and angles being unimportant. Footstep planning is often more interested in the detection of the *presence* of space, as that governs where the robot can step, which requires high accuracy. We believe that representing the environment as planar regions captures many of the salient features required for this type of footstep planning, such as inclination angle and potential environmental collisions, while providing an efficient convex representation.

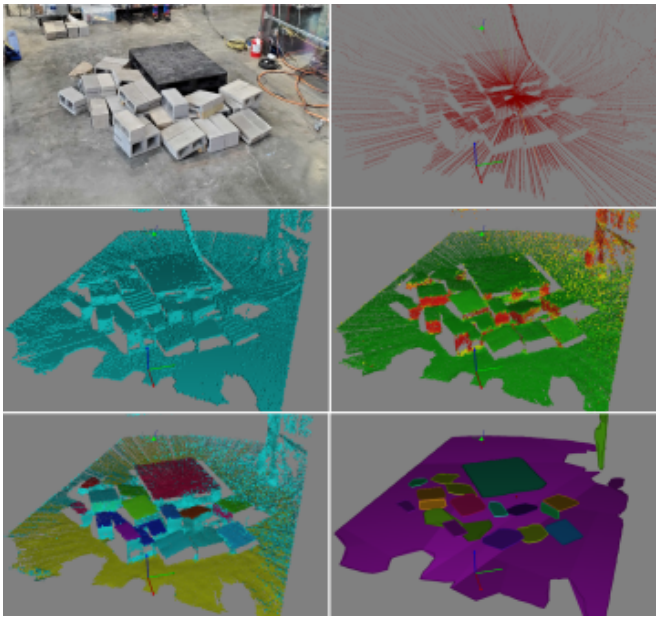

Fig. 1. Planar regions detection for rough terrain. Raw point cloud data (top right) is converted into an octree (middle left), which then estimates local normals estimated using RANSAC (middle right). After this, local normal vector are clustered via similarity and closeness into support plane (bottom left), which are finally polygonized as planar regions (bottom right).

In this work, we present an environmental awareness module that detects feasible planar regions for legged robots to use as footholds, as shown in Fig. 1. The primary objective of this module is to develop an adequate representation of the environment to enable the robot to autonomously find footstep plans, predominately for rough terrain, requiring a robust and reliable approach for environmental detection. The Atlas robot we use in our work is equipped with a MultiSense SL, which contains both a Multisense stereo pair and a Hokuyo UTM-30LX-EW spinning LIDAR [2]. From experimentation, we observed high sensor noise and limited range from the stereo pair, hence we chose to work with the Hokuyo LIDAR for its range and accuracy. While stereo vision is a well known, fairly proven method for obstacle detection and localization, even a few centimeters of surface error and degrees of normal error in a planar region can cause loss of balance for a walking robot with a high center-of-mass.

To extract the planar regions from the environment, we propose a novel combination of OctoMaps with nearest-neighbor planar region segmentation. Additionally, we outline a filter to remove the flying points that often occur around surface edges. OctoMaps are a useful and efficient

This work was funded through the ONR Grant N00014-19-1-2023 and by NASA Grant 80NSSC18M0071.

[1]Author is with the Florida Institute for Human and Machine Cognition, 40 S Alcaniz St, Pensacola, FL 32502, United States

[2]Author is with the University of West Florida, 11000 University Pkwy, Pensacola, FL 32514, United States

Email : {sbertrand,ilee,bmishra,dcalvert,jpratt,rgriffin}@ihmc.us

way to store large point clouds [3] and have been used directly for footstep planning with quadrupeds in the past [4]. Our nearest neighbor search merges octree nodes into planar regions using metrics such as perpendicular distance and normal vectors, which are then wrapped into concave hulls and decomposed to convex polygons.

To validate our environmental awareness module, we present several experiments on real world environments including cluttered rough terrain, stepping stones, and various obstacles. We additionally applied the presented perception module in conjunction with our recent footstep planner module on Valkyrie humanoid by NASA JSC and the DRC Atlas by Boston Dynamics in order to realize the autonomous locomotion [5], which we encourage the reader to review for other examples of this work's use.

II. RELATED WORK

There exist several interesting works in the literature related to planar region extraction from point clouds, including with applications to humanoid locomotion. The earliest example used scan-line grouping to find planar segments from stereo cameras, which are then combined using occupancy grid and floor height maps to generate a 3D map of the world for locomotion planning [6]. While this work is quite promising, it shows some accuracy issues that may be problematic when navigating 3D obstacles.

Another excellent work for the collection of planar regions is the Fast Sampling Plane-Filtering (FSPF) algorithm which uses RANdom SAmple Consensus (RANSAC) to find consistent planar neighborhoods within a given window inside depth images [7]. This work was extended to merge convex polygons over time by comparing the plane parameters found using eigenvector decomposition of the scatter matrix of points of each plane [8]. However, such an approach could possibly assign neighboring points to two overlapping planar regions. To address this, we propose to instead group the points into voxels and then generate and grow the planar regions in voxel-space, rather than point space.

The work by Marion [9] presents a novel algorithm for extracting planar polygons from point cloud data from either LIDAR or stereo camera by clustering the points based on RANSAC-based surface normal estimation. In the work, the unstructured point cloud is cleaned out by performing a voxel-grid filter and ground plane removal before performing surface normal estimation. However, the algorithm is only applied to point cloud generated from objects that are already convex. This approach was developed to work in conjunction with their mixed-integer quadratic program footstep planner, designed to autonomously find footstep plans during the DARPA Robotics Challenge (DRC) [10]. In practice, however, the problem was simplified to assume the world consisted of rectangles, greatly reducing the complexity of finding regions, and presenting a fairly valid assumption for the DRC Finals [9], [11]. It was extended to demonstrate continuous locomotion for a humanoid robot using only stereo sensors [12]. Here, they combined their mixed integer footstep planner [10] with a clustering routine for the stereo

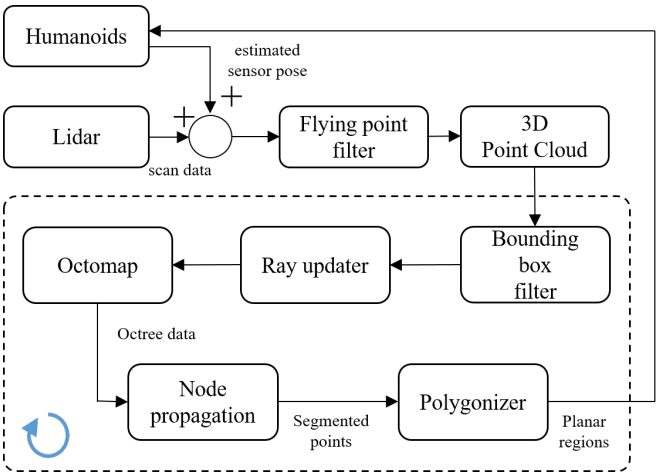

Fig. 2. Pipeline of the proposed environmental awareness algorithm for legged robots. In this case, the LIDAR points are filtered before being broadcast. They are then converted by the algorithm into planar regions, which are broadcast out to the network.

point cloud data described in [9] to fit rectangles to plan over rough terrain. While this work features impressive results, it makes assumptions as to the structure of the terrain (rectangles), and requires the robot to walk slowly to not distort the perception data beyond a usable amount.

An additional interesting example of planar region extraction being used by humanoid robots demonstrated using these regions to determine affordances. This approach, however, is difficult to evaluate as little information is given on the calculation of the regions themselves [13]. Another interesting approach is to use fusion data which is constructed by projecting LIDAR data to images[14]. This segments edges with several superpixels that are obtained at image level and achieves high accuracy with LIDAR rather than using RGBD. However, it requires well-calibrated projection model therefore only planar regions within a portion of the image are accurate and feasible.

While these works were concerned with finding occupied areas, Deits et. al. [15] present a clever technique for finding obstacle-free space for problems such as footstep planning in which they uniformly grow an ellipsoid encapsulated in a polytope, guaranteeing convex region generation. This was then used for path planning tasks for robots like Unmanned Aerial Vehicles [16].

While not finding reduced-order spaces, Armin et. al. [3] present a memory-efficient representation of 3D maps consisting of an octree-based data structure with probabilistic estimation of occupancy in the octree nodes. This has been shown to be a useful data structure for terrain mapping for legged locomotion [4], and is used heavily in the presented paper.

III. ROBOT ENVIRONMENTAL AWARENESS

The primary purpose of our environmental awareness algorithm is to take 3D point clouds, typically from LIDAR but also stereo or RGBD sensors, and convert them into lower-dimensional shapes that are more tractable for robot motion planning. We are particularly interested in utilizing

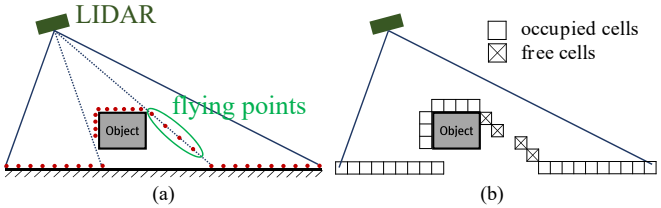

Fig. 3. Illustration of the flying point filter for a simple object. From the raw point cloud data (a), the flying points are removed by removing points that fall inline with the direction of the laser (b).

these lower-dimensional shapes to plan footholds for legged robots. To do this, we first filter these points to try and remove noise and other false data from the point cloud. We then convert the points into an octree based on an OctoMap [3]. After that, we perform a nearest neighbor search on the leaves in the octree to group them into regions. We then find the alpha-shape concave hull of each region, and finally decompose these regions into their composite convex polygons. This approach is outlined in more detail in the following section, and illustrated in Fig. 2.

A. Point Cloud Collection

Flying points, which are outlier points in free-space and do not belong to any physical object in the scene, are typically present in point clouds, particularly ones from LIDAR, and can be thought of as the result of objects having "shadows". The filtering of flying points is based on the observation that they always fall in line with the view direction of the laser ray [9], [17]. The filter compares the angle between the scanner view direction and the line segment connecting outlier points with their scan line neighbors, as shown in Fig. 3. When applied, a significant number of points are removed, shown in Fig. 4. If left in, these points result in an additional planar region found on the back side of the cinder block, along the edge of the block's shadow. Additionally, as LIDAR has a very short minimum range, it is not uncommon to collect points on the robot itself. To avoid this, we utilize convex outer approximations of each robot link, and remove any point contained within these bounding regions.

B. Octree Construction

A full 3D point cloud, particularly one collected over a period of time, represents a relatively intractable amount

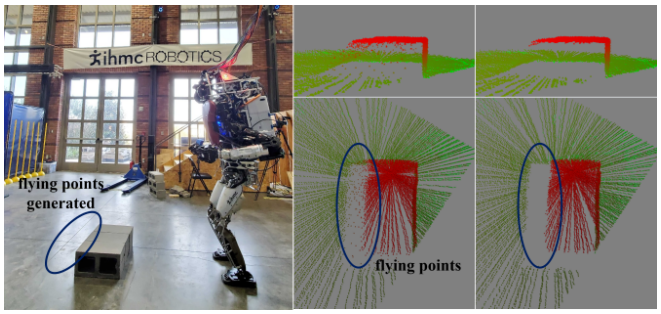

Fig. 4. Illustration of the flying point filter for a simple cinder block. The raw point cloud (top and bottom middle) is filtered to remove flying points (top and bottom right). The effect of removing flying points is clearly visible from this result.

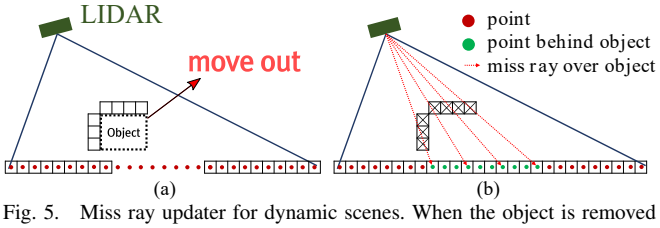

Fig. 5. Miss ray updater for dynamic scenes. When the object is removed and new points are collected, which are represented in red (a), the occupancy probability of nodes between the sensor and hit points (green) decays until it reaches a threshold (b), where it is then considered unoccupied.

of data, leading us to compress this data into a more efficient representation for map construction. Boundary-based (curves, surface meshes and implicit surfaces) and spatial-partitioning based (occupancy grids and octrees) are the two broad categories of higher-level 3D environment representations commonly used in robotics [1], with octrees being usable for highly-compressed representations of 3D space [18], [19]. However, representations that go a level further and represent objects as 3D solids (parameterized solid primitives, sweep objects, constructed solids) are also gaining prominence in robotic perception [1]. Here, we utilize a custom implementation of OctoMap [3], which is an efficient probabilistic octree for representing occupancy and aims to have fast updates with low memory consumption. Notably, we incorporated into our implementation some improved probabilistic updates of empty spaces, aimed at increasing tolerance to sensor noise.

As in [3], the equation for the probability $P(n|z_{1:t})$ of a leaf node n being occupied given the sensor measurements $z_{1:t}$ is estimated by

$$P(z_{1:t}) = \left[1 + \frac{1-P(n|z_t)}{P(n|z_t)} \frac{1-P(n|z_{1:t-1})}{P(n|z_{1:t-1})} \frac{P(n)}{1-P(n)}\right]^{-1}, \quad (1)$$

where $P(n)$ is the prior probability, $P(n|z_{1:t-1})$ is the previous occupancy estimate, and $P(n|z_t)$ is the probability of occupancy given the current measurement z_t. If we return a "hit point" in a certain cell, we can say with a probability $P(n|z_t) \approx 0.7$ that the cell is occupied, where 0.7 is tuneable. However, if the ray from the sensor to the hit point passes through another cell, we know that this cell is likely to be unoccupied, or $P(n|z_t) \approx 0.4$, allowing a slow decay in its occupancy estimate. This is shown in Fig. 5, where cells that were once occupied are no longer so after the object is moved.

With the introduction of sensor noise, though, it becomes harder to estimate whether or not a cell is occupied, as in Fig. 6. In this case, if we were to use the standard occupancy update, the points in nodes behind the green surface would say that the nodes in front are not occupied. To increase robustness against this type of noise, we introduce a filter such that the cells within a certain distance of the hit point, shown by the blue circle in Fig. 6, decay at a lower rate, $P(n|z_t) \approx 0.45$, as we are less confident that they are empty.

Additionally, we extended our implementation of OctoMap to not only include the probability of occupancy of each individual voxel, but also estimation of the surface normal. Inspired by other works [9], [20], [21], we use a

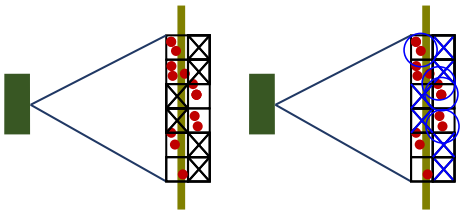

Fig. 6. Near miss filter. Noise in the perception data can cause nodes that are actually occupied to be occasionally detected as being free (left). By filtering nodes that are within a certain proximity of the hit point, they can have their probability decay at a lower rate, making them more robust to noise (right).

RANSAC algorithm [22] to estimate this normal. We then refine the node normal using least squares on all the points within a certain distance to the plane found by RANSAC, as in [9]. RANSAC is typically used as an initial estimate for the normal, as utilizing a least squares fit on all the points in the node does not capture discontinuities in surfaces well. Uniquely, though, because the entire OctoMap is updated at a fixed rate, we only require a single RANSAC iteration to be solved in each voxel per update. In practice, this results in RANSAC rapidly converging to a solution over the course of consecutive updates, rather than requiring convergence each time.

C. Node Propagation

The next step in the process is to take the leaves of the octree and group them into distinct planar regions. To do this, we designed a nearest-neighbor search with memory, which is outlined in Algorithm 1 and described below. We start the search for new, candidate regions by iterating over nodes in the octree that have a surface variance below a certain threshold, placing these nodes as the initial location and normal of the regions. We then grow from this node to consider nearby neighbors using an efficient radius neighbor search with an additive rule that is generally outlined in [23]. To summarize, this search is designed to exploit the structure of octrees, allowing early pruning of the subtree if the entire octant (which is the subtree) is within the search radius, avoiding computing the distance to all the leaves in the subtree. Then, we check our additive rule, which asserts whether or not each leaf in the subtree is part of the root region, and adds that leaf to the region if so. The rule first checks the orthogonal distance from the plane to the node. As each node actually represents a cluster of points, we use the average location of all the points in the node to determine this orthogonal distance. After this, we remove candidate nodes with angle differences between their normal and the region's normal above a certain threshold.

Once the regions have been expanded, we recompute their normals and standard deviation vectors considering all the component hit points contained within the region using principal component analysis. We then remove regions whose quality falls below certain metrics. That is, if the standard deviation along any axis is above a certain threshold, it is removed. Additionally, if the "density" of the region (the average ellipsoidal area per point) is below a threshold, it is removed. Lastly, we check if regions are able to be merged by evaluating their proximity and normal differences.

On subsequent updates of the map, we first update the existing planar regions. To do this, we initially remove any node from the region where the additive rule is invalid, and then remove all empty regions. From there, we reiterate the expansion process, using all the existing regions as the starting point.

Algorithm 1 Node propagation for grouping nodes of like surfaces into planar regions

Input: Octomap octomap **Output:**PlanarRegions regions

```
1:  for node N in octomap do
2:      if regions.contains(N) then continue
3:      end if
4:      if WithinBox(N) and NotVisited(N) then
5:          region ← NewRegion()
6:          nodes ← FindNeighborNodes(N)
7:          region ← GrowRegion(nodes)
8:          regions ← regions.Add(Region)
9:      end if
10: end for
11: regions ← StdDevThreshold(regions)
12: regions ← EllipsoidalThreshold(regions)
13: regions ← MergeRegions(regions)
```

D. Convex Polygonization

Once the octree leaves have been grouped into subsets, these groups can be used to first calculate the concave hull that composes the planar region, then decomposed into composite convex polygons. While a naive approach would be to find the convex hull of all the points contained in the planar region, this is not representative of the real world, e.g. it would ignore the interior of an open doorway. Instead, we choose to use a tuneable representation of the concave hull via the alpha-shape approach [24], [25]. This is chosen over other methods like the k nearest neighbor search for finding concave hulls, as the alpha shape method allows us to control the density of the points on the hull while still being fairly representative of the physical world. By modifying the alpha parameter in the algorithm, we are able to tune the "roughness" of the edges of these planar regions, as shown in Fig. 7. Changes in this alpha parameter directly correlate with the resolution of the concave hull.

Once the alpha-shape based concave hull of the planar region has been found, the next step is to decompose it into composite convex polygons. To satisfy our computational requirements, we adopt the method presented by Lien and Amato [26], which introduces several metrics for concavity, of which we use only the simplest depth-based metric. The algorithm works by finding these pockets of concavity, and then taking the "deepest" vertex in the concave pocket and splitting the concave hull with a bisector at that point, as shown in Fig. 8. While it is worth noting that this approach is only guaranteed to produce polygons that are *approximately* convex within some threshold of concavity, we have found

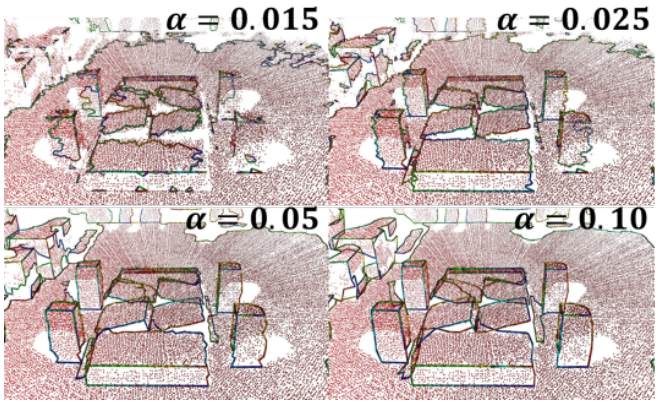

Fig. 7. Alpha shapes calculated concave hulls of planar regions. As the alpha parameter of these shapes is increased, the resolution of the resulting planar region decreases. Because the node size in our octree is 2 *cm*, we typically use $\alpha = 0.02$.

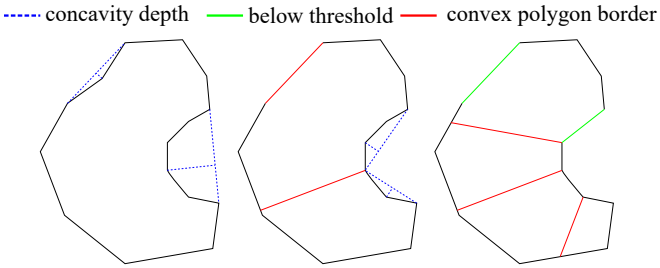

Fig. 8. The concave hull can be approximately decomposed into convex shapes using the algorithm presented by Lien and Amato [26]. The blue lines show the simple depth based concavity estimate to iteratively break the concave hull down into its composite convex polygons, shown by the red lines. The green lines represent the approximation of the concavities below the threshold as convex polygons.

that, in practice, this approximate solution is more than adequate.

IV. RESULTS

We tested the proposed planar region detection algorithm in various environments, including some that legged robots would use for walking such as cinder blocks and rocks, curved objects, an indoor environment scene, and stairs. The detection of planar regions in a cinder block field, which is a challenging terrain for locomotion, was very successful, as was the detection of stairs, as shown in Fig. 9. It is worth noting that, partly due to the angle of the sensor when mounted on the head of the robot, the algorithm tends to perform better at detecting the tops of surfaces than the vertical faces. As we are primarily concerned with using this information for finding steppable regions, this is of little concern. It is also worth noting that, by using concave hulls, the detected planar regions are almost always inner-approximations of the actual regions, unless two regions are mistakenly merged during the node propagation. While this is advantageous for footstep planning, it can be problematic during obstacle detection. This is very clearly seen in the results in Fig. 10, where much of the detected surfaces are smaller than the actual objects.

There are a variety of parameters of concern during the segmentation process. Here, we describe the importance and selection process of these parameters, with a particular focus on their effects for humanoid robots. The minimum size of

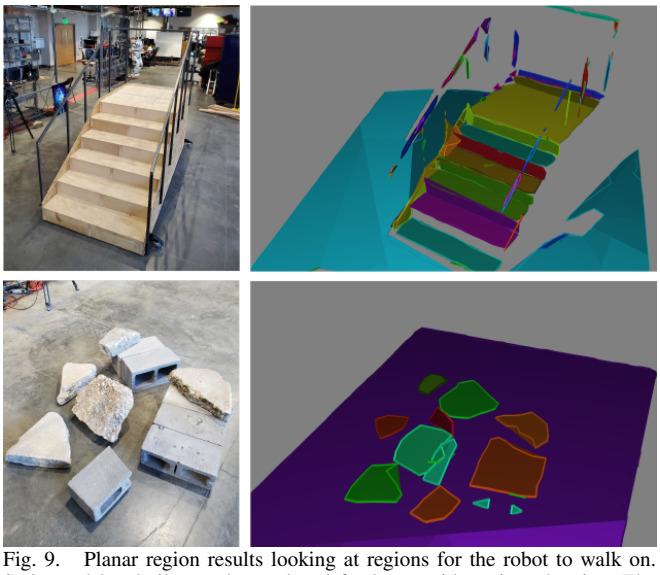

Fig. 9. Planar region results looking at regions for the robot to walk on. Stairs and handrails are detected and for humanoid motion planning. The tops of cinder blocks and concrete are well detected for planning footholds.

the planar regions is highly related to the geometric feature of the robot. If we require a large number of octree nodes in each segmented region, there will be very few valid regions, greatly reducing the duration of polygonization process, with the inverse also being true. However, this minimum value is strongly dictated by the dimensions of the robot in use. For example, the Atlas and Valkyrie robots have feet approximately 0.11 m x 0.22 m and 0.15 m x 0.25 m, respectively. If we then define the resolution of the octree node as 2 cm, the minimum size of the planar region that the robots can use with full foothold support is 121 for Atlas and 188 for Valkyrie. However, the force distribution within the foot in the robot's dynamic also affects the minimum area of support that is acceptable to walk and retain balance. In our application, we tend to use most of the foot, but retain some tolerance to partial footholds. Therefore, we select a minimum planar region size to be 50-100 nodes to provide a properly sized foothold for footstep planning.

The other major parameter that dominates the computation duration is the octree voxel size. With a lower value, we can achieve highly accurate planar region location and orientation, but this comes at the cost of significantly longer computation durations in the OctoMap update and the planar

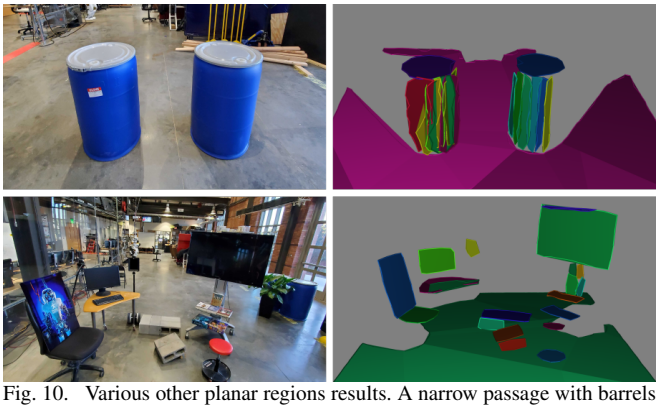

Fig. 10. Various other planar regions results. A narrow passage with barrels can estimate the curved surfaces with multiple, thinner planar regions. Various objects in the world are detected as large, flat regions.

Fig. 11. Results showing our robot, Atlas, collecting sensor data for planning and execute a footstep plan over rough terrain, using this work combined with our other work in [5].

region extraction. So we can increase the resolution up to the tolerance of the humanoid robots' locomotion capabilities, while considering the inaccuracy of the planar regions. This is highly related with the density of the point cloud data we collect with the LIDAR. We found that 2 cm was the best value for our humanoid robots, providing an appropriate balance of resolution and computational speed. To further help reduce the computational burden, we introduced the bounding box filter in Fig. 2 to reduce the workspace size and overall octree size. This 2 cm node resolution then drove our choice of alpha-shape value, which we also selected as $\alpha = 2$, the affects of which are highlighted in Fig. 7.

V. DISCUSSION AND FUTURE WORK

While the presented method has been shown in practice to be fairly computationally efficient, it still can only achieve an update rate of approximately 2 Hz on point clouds from stereo cameras when being performed entirely on a CPU. The most computationally expensive part of the process by far, though, is the update of the octree and estimation of surface normals. This type of problem lends itself nicely to parallelization and incorporation onto GPUs, which is an area of active research. To this point, though, the computational speed of the OctoMap update has been of little concern due to the duration required for data acquisition when using LIDAR, which we used in all the presented results.

While this produces qualitatively accurate planar regions, it comes at the cost of slow point-cloud acquisition speeds, requiring many scans of the LIDAR to provide the same quantity of data as a single camera frame. However, these camera frames often distort when the robot is moving, producing highly noisy data that is nearly unusable for planar region calculation. We are exploring methods for extracting planar regions from individual frames, then combining these frames to determine entire maps. For this to be feasible, the robot also requires localization so the different frames stitch together properly, which is another area of exploration.

A logical next step is also the combination of multiple sources of perception sensor data of different data types, such as LIDAR and stereo. We believe that, because of its probabilistic nature, the OctoMap data structure is well suited to this need, and are exploring how best to utilize highly accurate data like LIDAR in combination with highly dense data like stereo.

We have explored using knowledge of neighboring regions to help refine their intersecting edges. Detecting edges from point clouds can be challenging, particularly when using fairly sparse LIDAR data. However, by calculating where two planes intersect, we can compute what their expected edge should be. Given some heuristics, such as maximum plane extension distance, this can be used to greatly improve the edge detection capabilities of this algorithm. In practice, we do not use this feature extensively, but if combined with edge detection algorithms from vision-based sensors, it is very promising.

A. Source Code and Media

The implementation of our algorithms can be found on our GitHub page, `https://github.com/ihmcrobotics`.

VI. CONCLUSION

In this work, we presented a planar region detection module that aimed at extracting lower-dimensional shapes from 3D point clouds to be used by legged robots, and footstep planning in particular. To achieve this, we combined OctoMaps, which are an efficient method for aggregating large amounts of 3D point cloud data, with a nearest-neighbor search approach for calculating planar regions from the resulting octree. We believe that the chosen combination of algorithms, including the OctoMap [3], nearest-neighbor search [9], alpha-shape concave hull calculation [24], [25], and convex decomposition[26] result in an environmental awareness module that is a usable and reliable tool for our legged locomotion algorithms. We introduced several unique features that increase the computational efficiency and reliability of the resulting octree and nearest-neighbor searches. We also provided an in-depth discussion on our insights into the difficulties of this process, and include a variety of potential computational and performance improvements. We then illustrated the resulting algorithm on a variety of environments, including rough terrain, stairs, barrels, and various indoor objects. Lastly, we demonstrated the applicability of this approach by combining it with our previous footstep planning work [5] on our humanoid robot, Atlas.

REFERENCES

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[2] Carnegie Robotics, *MultiSense SL*, `http://docs.carnegierobotics.com/SL/`, [Online; accessed 28-Februrary-2020], 2017.

[3] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[4] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping, and whole-body control," *IEEE Transactions on Robotics*, 114, 2020.

[5] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt, "Footstep planning for autonomous walking over rough terrain," *arXiv preprint arXiv:1907.08673*, 2019.

[6] J.-S. Gutmann, M. Fukuchi, and M. Fujita, "3D perception and environment map generation for humanoid robot navigation," *The International Journal of Robotics Research*, vol. 27, no. 10, pp. 1117–1134, 2008.

[7] J. Biswas and M. Veloso, "Depth camera based indoor mobile robot localization and navigation," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 1697–1702.

[8] ——, "Planar polygon extraction and merging from depth images," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 3859–3864.

[9] J. P. Marion, "Perception methods for continuous humanoid locomotion over uneven terrain," Master's thesis, Massachusetts Institute of Technology, 2016.

[10] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2014, pp. 279–286.

[11] M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C. P. D'Arpino, R. Deits, M. DiCicco, D. Fourie, *et al.*, "An architecture for online affordance-based perception and whole-body planning," *Journal of Field Robotics*, vol. 32, no. 2, pp. 229–254, 2015.

[12] M. F. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake, "Continuous humanoid locomotion over uneven terrain using stereo fusion," in *2015 IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 881–888.

[13] P. Kaiser, D. Kanoulas, M. Grotz, L. Muratore, A. Rocchi, E. M. Hoffman, N. G. Tsagarakis, and T. Asfour, "An affordance-based pilot interface for high-level control of humanoid robots in supervised autonomy," in *2016 IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, IEEE, 2016, pp. 621–628.

[14] I. Lee, J. Oh, I. Kim, and J.-H. Oh, "Camera-laser fusion sensor system and environmental recognition for humanoids in disaster scenarios," *Journal of Mechanical Science and Technology*, vol. 31, no. 6, pp. 2997–3003, 2017.

[15] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic foundations of robotics XI*, Springer, 2015, pp. 109–124.

[16] R. Deits and R. Tedrake, "Efficient mixed-integer planning for UAVs in cluttered environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 42–49.

[17] I. Shim, S. Shin, Y. Bok, K. Joo, D.-G. Choi, J.-Y. Lee, J. Park, J.-H. Oh, and I. S. Kweon, "Vision system and depth processing for DRC-HUBO+," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 2456–2463.

[18] D. C. Garcia, T. A. Fonseca, R. U. Ferreira, and R. L. de Queiroz, "Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts," *IEEE Transactions on Image Processing*, vol. 29, pp. 313–322, 2020.

[19] M. Slomp, H. Kawasaki, R. Furukawa, and R. Sagawa, "Temporal octrees for compressing dynamic point cloud streams," in *2014 2nd International Conference on 3D Vision*, vol. 2, 2014, pp. 49–56.

[20] M. Alehdaghi, M. A. Esfahani, and A. Harati, "Parallel RANSAC: Speeding up plane extraction in RGBD image sequences using GPU," in *2015 5th International Conference on Computer and Knowledge Engineering (ICCKE)*, 2015, pp. 295–300.

[21] W. Yue, J. Lu, W. Zhou, and Y. Miao, "A new plane segmentation method of point cloud based on mean shift and RANSAC," in *2018 Chinese Control And Decision Conference (CCDC)*, 2018, pp. 1658–1663.

[22] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[23] J. Behley, V. Steinhage, and A. B. Cremers, "Efficient radius neighbor search in three-dimensional point clouds," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 3625–3630.

[24] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Transactions on information theory*, vol. 29, no. 4, pp. 551–559, 1983.

[25] H. Edelsbrunner and E. P. Mücke, "Three-dimensional alpha shapes," *ACM Transactions on Graphics (TOG)*, vol. 13, no. 1, pp. 43–72, 1994.

[26] J.-M. Lien and N. M. Amato, "Approximate convex decomposition of polygons," in *Proceedings of the twentieth annual symposium on Computational geometry*, 2004, pp. 17–26.