Autonomous Robot Navigation: Path Planning on a Detail-Preserving Reduced-Complexity Representation of 3D Point Clouds

Rohit Sant, Ninad Kulkarni, Ainesh Bakshi, Salil Kapur, and Kratarth Goel

BITS-Pilani, K.K. Birla Goa Campus, Goa, India
{rhsant,ninadk1092,aineshbakshi,salilkapur93, kratarthgoel}@gmail.com

Abstract. Determination of a collision free, optimal path achieved by performing complex computations on an accurate representation of the terrain, is essential to the success of autonomous navigation systems. This paper defines and builds upon a technique – Spatially Important Point (SIP) Identification – for reducing the inherent complexity of range data without discarding any essential information. The SIP representation makes onboard autonomous navigation a viable option, since space and time complexity is greatly reduced. A cost based, dynamic navigation analysis is then performed using only the SIPs which culminates into a new time and space efficient Path Planning technique. The terrain is also retained in the form of a graph, with each branch of every node encountered, indexed in a priority sequence given by its cumulative cost. Experiments show the entire dataflow, from input data to path planning, being done in less than 700ms on a modern computer on datasets typically having 10^6 points.

Keywords: Autonomous navigation, path planning, range data, dynamic programming, spatially important points.

1 Introduction

Autonomous navigation on partially unknown terrains is a very popular problem statement in the field of robotics. Representation of a terrain in the robots configuration space is intricately related to deliberative path planning algorithms. Terrain can be represented in great detail with modern sensors, but the computational cost associated with real-time navigation on such terrain is often too much for onboard processing. It is with this background and the subsequent motivation that we propose an integrated approach towards terrain representation and path planning. In our technique for terrain data representation we work towards reducing its complexity while preserving the powerful nature of geometric 3D Models of the scene. We detect points essential to the representation of the terrain and refer to them as Spatially Important Points (SIPs). This is done by comparing 'importance' values of each point to those in its neighborhood and also to a template. This algorithm ensures retention of essential features which results in detail preserving, vastly compressed datasets which are easily reconstructable by simple linear interpolation. Our customized Navigation

Analysis works with the minimal terrain representation, forming clusters which are ascertained to be both absolutely and relatively navigable subject to mechanical and topological constraints as well as to the current state of the robot. Finally, it performs a modular cost based path planning implementation to obtain an optimal path on the same.

The organization of the paper is as follows: Section (2) describes related work in the field of autonomous navigation, Section (3) describes, in detail, the working of our Spatially Important Point algorithm, Reconstruction and Path Planning and their implementation is discussed. Results are tabulated in Section (4) and Section (5) concludes the paper.

2 Review of Existing Techniques

Reduced-complexity representation of 3D data is an implicit requirement for almost all research concerning 3D range data. Early work, for example that by Martin et al. [5] uses a median value to represent each 3D grid in an octree. Though this reduces complexity, it results in indiscriminate loss of detail. A variation on the same theme by Lee et al. [6] incorporates non-uniform grids, but the results still suffer due to the downsampling technique. A technique described in [3] uses multiple scans to identify redundancy, but this adds an unwanted restriction on the input and severely limits its usefulness in situations where multiple scans cannot be obtained. Moenning and Dodgson [4] have a different motivation; in context of our target, our results overshadow theirs. A notable implementation can be seen in [7], in which the authors approach this problem using curvature.

Since this is usually an interim step in a larger problem statement, most researchers do not bother with operating on raw data, preferring to use clustering [2], [13] usually with a k-nearest neighbors approach.

Path Planning has been studied continuously from the 1970's leading to extensive research in the field. Configuration Space approach originates in the work of Lorenzo-Perez [8] and underlies most Path Planning research. Brooks's planner [9] decomposes the configuration space into a graph of free, blocked, and mixed cuboids. Takahashi and Schilling [16] plan for a rectangular robot with polygonal obstacles via heuristic search of the generalized Voronoi diagram. Heuristic searches will try to find any path, but will do so faster than blind search [10]. This has led to a range of heuristic algorithms being adopted by or developed for autonomous navigation, such as A*, Dijkstra's, D* and D* Lite [9], [10]. Best first search algorithms such as these work with numerous types of configuration spaces to find the optimal path yet does not localize the effect of obstacles in the representation. Probabilistic algorithms such as artificial potential field algorithm and road map algorithms, modified A* search algorithm, and genetic algorithms are also used yet they are plagued by local minima problems a work around to which has been given by Hussein [9].

Like existing techniques Nashashibi [11], we approach 3D terrain representation and subsequent path planning as a composite problem statement.

2.1 Our Contribution

We develop a completely new technique for autonomous navigation starting from basic geometry, which enables our algorithm to achieve better efficiency, due to its specialized nature. With our reconstruction technique we demonstrate that our algorithm's low-complexity range data representation is accurate. We introduce the concepts of the dichotomy of navigability (relative and absolute), as also the technique of base detection. All these techniques are then used in an integrated manner to carry out a fast, structured navigation which uses dynamic programming and a graph storage format for path planning. The computations saved by our terrain representation [1] and the on-demand nature of our path planning technique provides a unique and useful solution to the problem statement. The technique is completely portable, and may be implemented on any robot equipped with a laser rangefinder.

3 Algorithm Description

The flowchart shown here gives a basic block diagram of our technique in full. Each component is subsequently explained in detail.

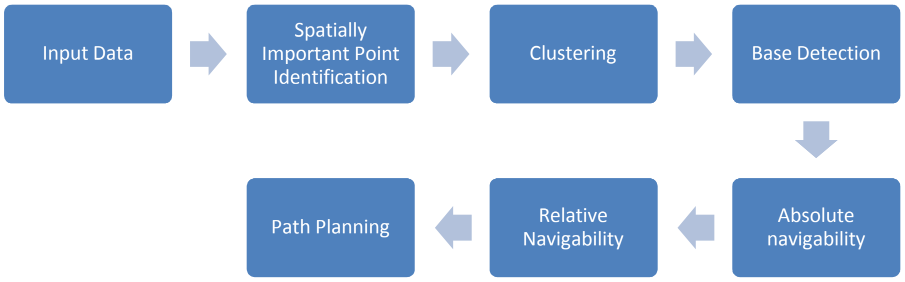

Fig. 1. Workflow of the technique described in the paper

3.1 Overview of Spatially Important Point Technique

An in-depth discussion on the Spatially Important Point Technique can be found in our supplementary material [1]. We summarize it here.

A Spatially Important Point, as defined by us, is a point where the terrain changes appreciably. The Spatially Important Point Technique finds points of significance in a terrain by carrying out a sort based on importance, where the importance function f is defined as in equation (1) in [1].

The defining characteristic of this function is that it identically equals zero for any consecutive point-pair on a flat terrain. Hence deviations of f from zero provide a good measure of the absolute change in the terrain. Comparisons with neighboring values of f is a good way of judging the relative change in the terrain. Using these concepts, equations (2-5) in [1] describe the procedure of identifying SIPs.

We discard everything but the Spatially Important Points got from the above procedure. All points which lie between two such consecutive constant-elevation Spatially Important Points can be regenerated by a linear interpolation between the said Spatially Important Points, thus eliminating the need to store them explicitly. We therefore hypothesize that a representation of the terrain by its Spatially Important Points is a complete representation. We hereon refer to this technique as Spatially Important Point Representation (SIPR). Our hypothesis is tested using the reconstruction technique which is built around linear interpolation; results can be seen in [1].

Unlike [12], [4], [5] and [6] which build upon more expensive constructs like meshes, grids, triangulations and /or expensive post-processing using ICP [3], we develop a completely new technique starting from basic geometry and build up on it. Our algorithm achieves better efficiency, due to its specialized nature. To the best of our knowledge, our algorithm for low-complexity range data representation does indeed provide a unique and useful solution for detail preserving reduction of point data – thus removing the final drawback of range data.

3.2 Building on SIPR – The Navigation Algorithm

The output of SIPR is still a point cloud consisting of raw points belonging to the original dataset, with no aberrations. On this point cloud we perform mild clustering using the DBSCAN clustering algorithm. A cluster is a collection of Spatially Important Points belonging to the same geometric or topological feature. It is important because it imparts strong connectivity to the SIPs. We then perform a two-step navigability analysis on these clusters and retain those which are traversable given inherent mechanical constraints on the robot. The Navigation analysis initially marks out regions of the dataset which are absolutely non-navigable regardless of the position, orientation and state of the robot. The output of this stage is a portion of the dataset which is evaluated for navigability based on the current state of the robot. Novel procedures such as Base Detection and Cross Cluster Planing are used to determine vital distance and slope parameters. Our algorithm integrates path planning with this navigability analysis thereby greatly reducing the number of computations. We finally take a cost based approach to choose the best available path and retain a graph of all nodes previously visited. We define separate costs for separate outcomes, and the final composite cost can be chosen in accordance with the purpose of the navigation, which is a highlight of the modularity of this algorithm.

Clustering. Clusters are obtained by using the DBSCAN [14] algorithm while preserving a bias for obtaining smaller cluster size. The basis for this is averting misclassification of points into clusters to which they do not inherently belong.

Base Detection. Since the SIPR algorithm removes all regularity, the clusters obtained must be interpreted carefully- a cluster implies irregularity in the terrain and the robot must travel in the intervening region between clusters for maximum stability. The base detection algorithm finds 'ground level' points for each cluster to facilitate navigation given that all intervening spatial information between clusters has been done away with. The algorithm description is as follows:

- The centroid for each cluster is calculated using the standard centroid formula.
- The cluster is then divided into sectors of 10 degrees each, centered on the centroid, and the point with the least z coordinate is chosen in each of these sectors if one happens to exist.
- This computation is carried out for each cluster that is present thereby obtaining the base points for each cluster.

Absolute Navigability. This analysis removes all clusters that are absolutely non navigable subject to mechanical constraints of the robot. These clusters are obstacles that cannot be navigated under any actual circumstance by the robot, independent of direction of approach. The algorithm description is as follows:

- Check whether the height of a cluster, calculated by the absolute value of the maximum difference in the Z co-ordinates of the points, is less than a given height threshold. If it is, then the cluster is absolutely navigable.
- If a point is greater than the height threshold, we calculate the average of the maximum slope between all base point and this point and repeat it for all such points. If the average of all maximum slopes is greater than given slope threshold then the cluster is not navigable otherwise it is.
- All absolutely non navigable clusters are discarded while the others are retained.

Mathematical Expression:
Cluster C_j is retained

If

$$(z_{max} - z_{min}) < \tau_h \tag{1}$$

Else

$$avg(max(Sl(z', B(C_j)))) < \tau_s \tag{2}$$

Where
z' is the z-co-ordinate for a general point belonging to cluster C_j s.t. $z' > \tau_h$
$Sl(a, b))$ denotes the slope between points *a* and *b* measured w.r.t. the XY plane
$B(C_j)$ denotes the set of base points of cluster C_j
τ_h denotes the height threshold
τ_s denotes the slope threshold

Relative Navigability. Relative Navigability analysis is done on clusters retained after absolute navigability analysis. It proposes a technique to perform the reachability analysis for each cluster in the dataset. Reachability analysis ranks all clusters, found in a fixed radius search around a given cluster, in accordance with their reachability index.

- The reachability index of a neighbor of a given cluster is calculated whereby two parameters are taken into consideration: distance and slope.

- The minimum distance parameter $\mathcal{D}$ is obtained by selecting each base point of the given cluster and calculating the minimum Euclidian distance to all base points of the neighboring cluster.
- In a similar manner, the slope parameter $\mathcal{S}$ is obtained by calculating the average of the maximum slope between a set of all base points of the given cluster to those of the neighboring cluster. Mathematical expression:

$$\forall P_i \in \mathcal{B}(\mathbb{C}_r) \; and \; \forall Q_k \in \mathcal{B}(\mathbb{C}_s), \tag{3}$$

$$\mathcal{D}(\mathbb{C}_r, \mathbb{C}_s) = min\,(d(P_i, Q_k)) \tag{4}$$

$$\mathcal{S}(\mathbb{C}_r, \mathbb{C}_s) = \text{avg}(ma\,x(Sl(P_i, Q_k))) \tag{5}$$

Where
$B(\mathcal{C}_j)$ denotes the set of base points of cluster $\mathcal{C}_j$
$Sl(a, b))$ denotes the slope between points *a* and *b* measured w.r.t. the XY plane
$d(a, b)$ denotes the Euclidean distance between points a and b

Path Planning. This section amalgamates two concepts namely region growing and path planning into one. These regions are representative of the traversable path on a macro level. The exact traversable path can determined by performing a simple Voronoi path detection algorithm on the regions that are grown. Since we use a cost based approach, we begin by defining the various costs.

The costs, as defined by us are navigability cost, direction cost and destination cost.

Navigability Cost. $(\mathcal{C}_N)$This cost is determined by multiplying the distance and slope parameters. This is because both contribute equally to reach a subsequent cluster and large increase in either one would increase mechanical effort by the same factor.

Direction Cost. $(\mathcal{C}_D)$ Direction Cost is defined as the dot product of the vector joining current cluster centroid to the subsequent cluster centroid and the vector joining current cluster centroid and destination cluster centroid. A weighted average of these two costs is then taken and to it we add the destination cost.

Destination Cost. $(\mathcal{C}_F)$ It is defined as the Euclidian distance between the current cluster centroid and destination cluster centroid, thus forming a cumulative cost.

Cumulative Cost. $(\mathcal{C}_T)$ The weighted sum of the above costs is the cumulative cost given as

$$\mathcal{C}_T = a_1.\mathcal{C}_N + a_2.\mathcal{C}_D + a_3.\mathcal{C}_F \tag{6}$$

The constants a_1 and a_2 are chosen according to the purpose of the navigation. a_3 is chosen for scaling purposes.

Each cluster can now be considered as a node in a graph and paths connecting it to its neighboring clusters as its branches.

Starting from the current position of the robot, which we consider as the starting node, regions are grown on the basis of evaluating all branches, each of which have a cumulative cost (heuristically defined) assigned to them. From here on the words cluster and node will be used interchangeably. The algorithm description is as follows:

- Starting with the initial position, all clusters in the reachability index of the current cluster are considered by sorting the distance array. If the distance parameter of the neighboring cluster is greater than bot size it checks the slope parameter and finally ranks according to the product of the distance parameter $\mathcal{D}$ and the slope parameter $\mathcal{S}$ each cluster. This is to provide equal importance to both the parameters.
- The most favorable branch to the subsequent node from the current node is chosen based on the least cumulative cost and added to the optimum path. All other branches of the current node are sorted based on the same cost and are retained.
- Repeated recursively in order to get to the lowest cost node while continuously checking whether the subsequent node already exists in the optimum path. If it does, then this node is called a blacklist node, to which the robot doesn't traverse again as it would get stuck in a continuous loop.
- In the case where all subsequent nodes to a current node have already been black listed, it has to retrace its path to the previous node in the optimal path.
- Terminating condition is to reach the target node as set initially.

4 Results

The algorithm was applied on a number of datasets, and the entire flow for a single dataset has been tabulated below. We deal with an input terrain represented as a point cloud in 3D. For the purposes of this research, we used the Canadian Planetary Emulation Terrain 3D Mapping Dataset [5]. This data is represented as co-ordinates on a spherical (Range, Elevation, Azimuth) system. All images are screenshots of point clouds viewed in the PCD viewer, part of Point Cloud Library [15].

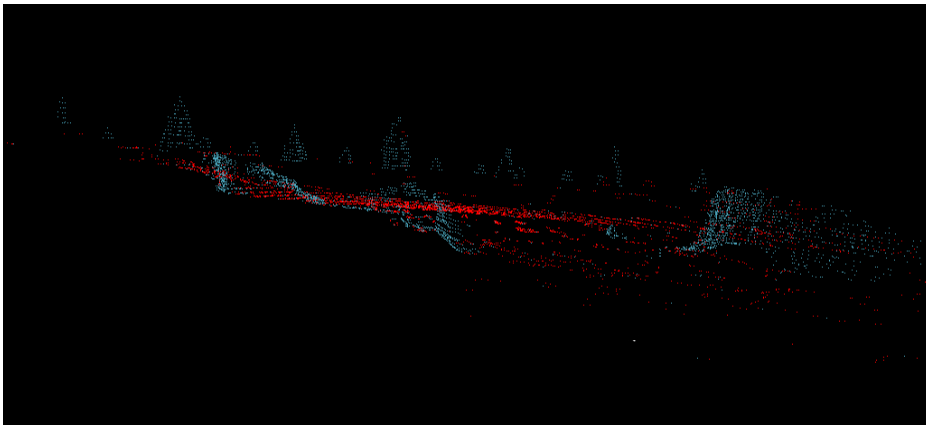
Fig. 2. A side view of a terrain intended to showcase the heights of the clusters rejected (in blue) by the absolute navigability algorithm

4.1 Absolute Navigability

Figures 2 and 3 show the results of absolute navigability on a terrain, with segments in blue indicating absolutely non-navigable sections of the terrain. The algorithm does a good job of picking out parts of the terrain which cannot be scaled by a typical hexapod (the dimensions of the robot determine the limits of absolute navigability). Trees and large contours are well left out.

Since our clustering is very light, it is possible that a single feature gets broken into multiple small clusters which may individually be absolutely navigable. However our treatment of relative navigability ensures these clusters are never chosen for traversal.

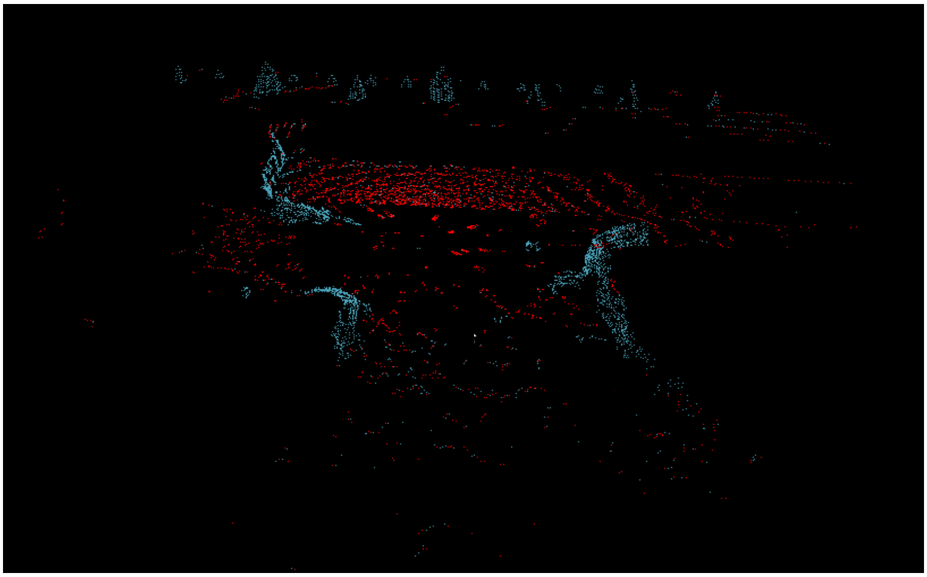
Fig. 3. A reverse angle view of the terrain in Figure 2

4.2 Path Planning

This image (Figure 4) is the final path generated by our algorithm with $a_1 = 100$ and $a_2 = 800$. The destination cost coefficient a_3 was set to -10, and appropriate scaling was done on the costs to bring them to the same order of magnitude. The blue pixels indicate the original terrain, the red pixels denote the Spatially Important Points and the white line corresponds to the path that will be traversed. (The white line actually merely indicates the clusters around which the path is to be planned. The actual step-by-step path can be found by further analysis using these clsuters as obstacles between which Voronoi regions may be found.)

The costs chosen above correspond to a real-world scenario for a robot that need not map and need not carry payloads – the objective being reaching the destination in a short path. . The high contribution of the navigability ensures that the path taken by the robot does not correspond to a difficult to traverse path; we can argue that this sort of value will be permissible, given that the robot will already have removed the non-navigable regions through its absolute and relative navigability analyses.

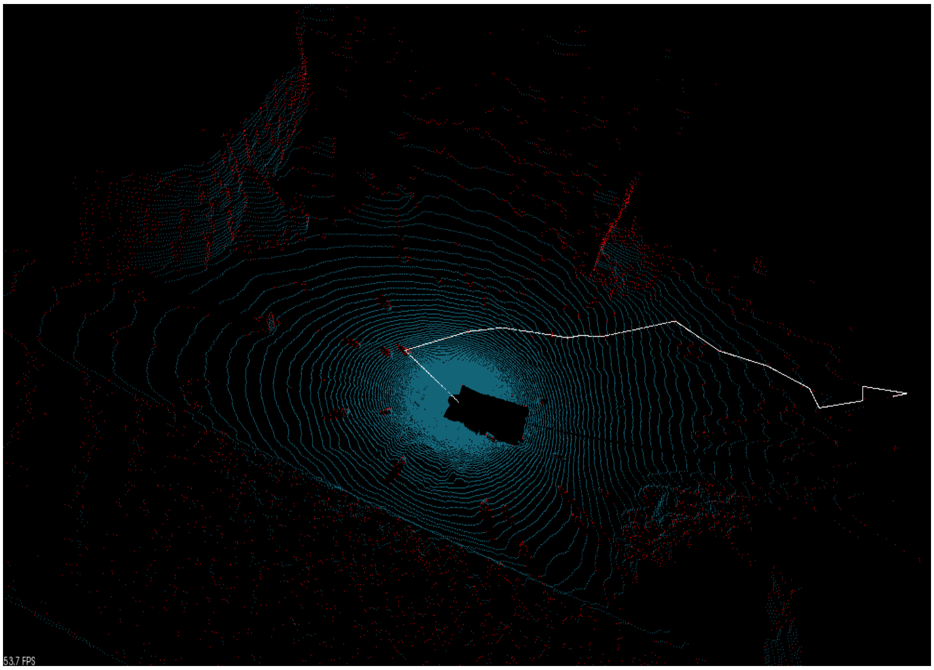

Fig. 4. Output image showing the final path (in white) overlaid on the dataset (blue) and its SIPs (red)

5 Conclusions and Future Work

The navigation algorithm described in this paper, incorporating Spatially Important Point Identification (SIPI) is a tailored and fundamentally strong approach towards a grassroots path planning algorithm. The algorithm has been broken down into stages and explained through this paper, and results at the outputs of various blocks in the workflow have been shown for one terrain.

The applications of this algorithm are immediate in the fields of robotic rescue, reconnaissance, mapping and exploration. It is a complete, self-sufficient system that has been built from the ground up, with integrated systems from the sensor inputs stage to the final navigation stage. It hence achieves the kind of efficiency that non-native systems cannot achieve.

As future work, the authors have laid the foundations for a machine learning module that seamlessly integrates with the path planning module, creating an efficient robot that can truly autonomously navigate a terrain, and continue to learn while on the job.

References

1. Sant, R., Goel, K., Kulkarni, N., Bakshi, A., Kapur, S.: Spatially Important Point Identification: A New Technique for Detail-Preserving Reduced-Complexity Representation of 3D Point Clouds (March 22, 2013), Retrieved from Google Drive, `http://goo.gl/gUGgJ`
2. Zhao, Y., He, M., Zhao, H., Davoine, F., Zha, H.: Computing Object-based Saliency in Urban Scenes Using Laser Sensing. In: International Conference on Robotics and Automation, pp. 4436–4443. IEEE, Saint Paul (2012)
3. Swadzba, A., Vollmer, A., Hanheide, M., Wachsmuth, S.: Reducing noise and redundancy in registered range data for planar surface extraction. In: 19th International Conference on Pattern Recognition, pp. 1–4. IEEE, Tampa (2008)
4. Moenning, C., Dodgson, N.A.: A New Point Cloud Simplification Algorithm. In: Proceedings of 3rd IASTED Conference on Visualization, Imaging and Image Processing, pp. 1027–1033. IASTED, Benalmádena (2003)
5. Tong, C., Gingras, D., Larose, K., Barfoot, T.D., Dupuis, E.: The Canadian Planetary Emulation Terrain 3D Mapping Dataset. International Journal of Robotics Research (IJRR) (2012)
6. Lee, K.H., Woo, H., Suk, T.: Data Reduction Methods for Reverse Engineering. Int. J. Adv. Manuf. Technol., 735–743 (2001)
7. Song, W., Cai, S., Yang, B., Cui, W., Wang, Y.: A Reduction Method of Three-Dimensional Point Cloud. In: 2nd International Conference on Biomedical Engineering and Informatics, pp. 1–4. IEEE, Tianjin (2009)
8. Brooks, R.A., Lorenzo-Perez, T.: A subdivision algorithm in configuration space for find-path with rotation. IEEE Transactions on Systems, Man and Cybernetics SMC-15(2), 225–233 (1985)
9. Jensen, R.M., Bryant, R.E., Veloso, M.M.: SetA*: An efficient BDD-based heuristic search algorithm. Tech. rep., Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA (August 2002)
10. Beginners guide to pathfinding algorithms, `http://aidepot.com/Tutorial/PathFinding.html` (visited on September 3, 2006)
11. Nashashibi, F., Fillatreau, P., Dacre-Wright, B., Simeon, T.: 3-D autonomous navigation in a natural environment. In: Proceedings of the 1994 IEEE International Conference on Robotics and Automation, May 8-13, vol. 1, pp. 433–439 (1994)
12. Moorthy, I., Millert, J.R., Hut, B., Berni, J.A., Zareo-Tejada, P.J., Lit, Q.: Extracting tree crown properties from ground-based scanning laser data. In: IEEE International Geoscience and Remote Sensing Symposium, pp. 2830–2832. IEEE, Barcelona (2007)
13. Zhao, H., Liu, Y., Zhu, X., Zhao, Y., Zha, H.: Scene Understanding in a Large Dynamic Environment through Laser-Based Sensing. In: International Conference on Robotics and Automation, pp. 127–133. IEEE, Anchorage (2010)
14. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: 2nd ACM International Conference on Knowledge Discovery and Data Mining (KDD), pp. 226–231. ACM, Portland (1996)
15. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–4. IEEE, Shanghai (2011)
16. Takahashi, O., Schilling, J.: Motion planning in a plane using generalized Voronoi diagrams. IEEE Transactions on Robotics and Automation 5(2), 143–150 (1989)