The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Hybrid Elevation Maps: 3D Surface Models for Segmentation

B. Douillard, J. Underwood, N. Melkumyan, S. Singh, S. Vasudevan, C. Brunner, A. Quadros

Abstract— **This paper presents an algorithm for segmenting 3D point clouds. It extends terrain elevation models by incorporating two types of representations: (1) ground representations based on averaging the height in the point cloud, (2) object models based on a voxelisation of the point cloud. The approach is deployed on Riegl data (dense 3D laser data) acquired in a campus type of environment and compared against six other terrain models. Amongst elevation models, it is shown to provide the best fit to the data as well as being unique in the sense that it jointly performs ground extraction, overhang representation and 3D segmentation. We experimentally demonstrate that the resulting model is also applicable to path planning.**

I. Introduction

This paper presents an algorithm to segment 3D point clouds into separate objects by using a model of the ground. The approach is developed in the context of an outdoor activity monitoring system which involves a number of robotic agents deployed in a campus environment. Each robotic agent is equipped with multiple modalities including laser sensors. This data must be: (1) integrated in a terrain model for navigation and (2) segmented into different objects for later classification. Current algorithms do not do both, so it is proposed to combine the two related tasks in a synergistic way.

The proposed algorithm builds on prior approaches to the generation of $2\frac{1}{2}$D maps; also called elevation maps. Two separate models are used for the ground and objects, utilizing simple, established map representations where they are most effective. The ground model is obtained by averaging the height of the returns falling in each 2D cell of the terrain grid. This results in a clean ground surface, but does not represent objects well. Objects are instead represented by detailed 3D voxels, generated efficiently by concentrating on occupied 2D cells and reasoning on the minimum and the maximum height of laser returns. Object voxels that belong to the ground are reintegrated into the ground model, allowing the algorithm to represent overhangs. This approach results in the fast and accurate segmentation of scenes.

This study uses data from a stationary Riegl sensor. Shown in Fig. 1(a), the Riegl sensor generates dense 3D point clouds, providing a high amount of geometrical detail to test segmentation techniques. It is, however, limited in extent. The dataset used is shown in Fig. 1(b).

II. Related Work

This section presents six models which were re-implemented in the context of this study and from which various aspects were retained in order to propose a novel segmentation algorithm.

The authors are with the Australian Centre for Field Robotics, Sydney, Australia; contact: b.douillard@acfr.usyd.edu.au

(a) (b)

Fig. 1. (a) The Riegl laser scanner. (b) Example of Riegl data: a 3D point cloud in which each laser return is augmented with RGB values (note that colour information is not used in this work).

A. Elevation Map Mean

Elevation maps are commonly classified as $2\frac{1}{2}$D models. The third dimension is only partially modeled since the terrain is represented by a grid in which each cell contains only one height. A standard approach to computing an elevation map is to average the height of the returns falling in each grid cell [6]. This method will be referred to as an elevation map of type "Mean". Its output is illustrated in Fig. 2(a).

The advantage of averaging the height of the laser returns is that noisy returns can be filtered out: the set of green vertical bars around the central white disk in Fig. II correspond to noisy returns; such spikes do not appear in Fig. 2(a) due to the averaging process. A disadvantage of the technique is that it cannot capture overhanging structures. This aspect is addressed by the proposed model.

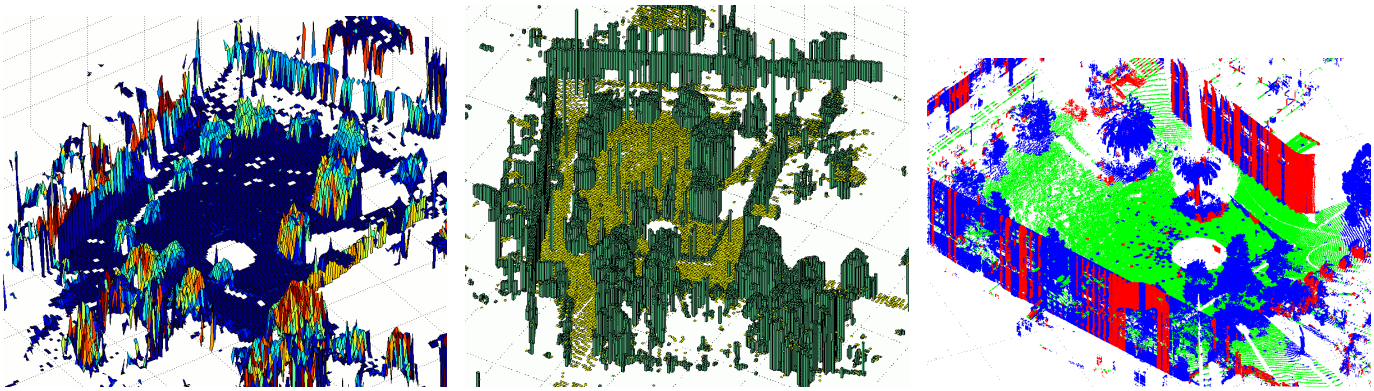
Since this representation is able to generate smooth surfaces by filtering out noisy returns, it was chosen as the basis for the ground part of the model.

B. Elevation Map Min-Max

The technique presented in [8] and developed in the context of the DARPA Grand Challenge 2006 is used here as the way to accurately capture the height of the returns in each grid cell. It computes the difference between the maximum and the minimum height of the returns falling in a cell. A cell is declared occupied if its height difference exceeds a pre-defined threshold (threshold $hThresh$ in Table I). Height differences provide a computationally efficient approximation to the terrain gradient in a cell. Cells which contain too steep a slope or are occupied by an object will be characterized by a strong gradient and can be identified as occupied. This technique generate models which will be referred to as elevation maps of the type "Min-Max". Its output is illustrated in Fig. II.

The advantage of such a technique is that it does not make an approximation by avoiding averaging and by reasoning on

978-1-4244-6676-4/10/$25.00 ©2010 IEEE



(a) (b) (c)

Fig. 2. Three examples of elevation models generated using the data in Fig. 1. (a) Mean Elevation Map. Red indicates a high standard deviation of height. (b) Min-Max Elevation Map. Yellow cells are the ground, green cells contain an object. (c) Map produced by the approach in [5]. Green is the class "ground", red is "vertical surface" and blue is "vertical surface containing a gap".

the minimum and maximum height of the returns falling in a cell. On the other hand, as mentioned in the previous section, it is more sensitive to noise that an elevation map of the type "Mean".

Since this model does not make an approximation of the height of the return, it was chosen as the basis of the object part of the proposed model.

C. Multi-Level Elevation Map

The approach developed in [5] for representing multiple levels of elevation was tested in the context of this study. It is an extension of elevation maps which handles overhanging structures and allows the generation of large scale 3D maps (by recursively registering local maps). Our implementation of this technique builds on the "Min-Max" approach described in the previous section. The output it produces is illustrated in Fig. 2(c). While the algorithm proposed here also discretizes the vertical dimension in order to capture overhanging structures, it differs from the work in [5] in two ways. Firstly, the only classification proposed involves the two classes "ground" and "object above the ground". Further differentiating objects may not facilitate segmentation, as seen by walls classified differently around windows or other causes of sparsity in the data. Secondly, the ground in [5] is not used as a reference for vertical height. It is argued in this work that such a reference is necessary when reasoning on the variation of height over a grid. A flat surface may be above and distinct from the ground. To avoid the need of hand-crafting a sensor model, the implementation used here reasons on the minimum and the maximum height of the returns in the "vertical segments" in [5].

D. Volumetric Density Map

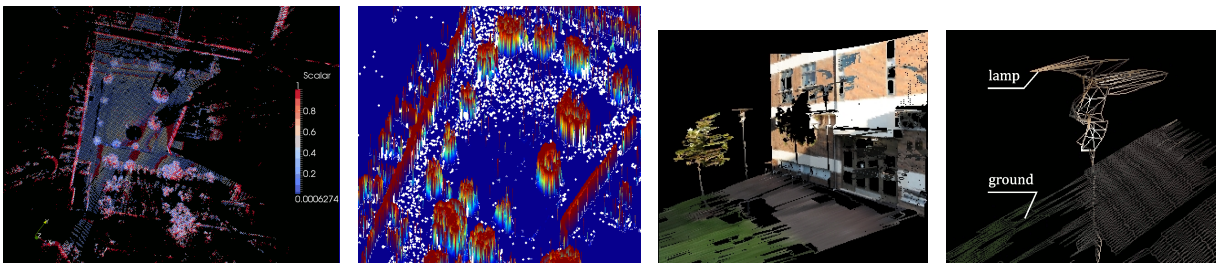
The approach proposed in [3] for discriminating soft and hard obstacles was evaluated in the context of this study. The output it produces is illustrated in Fig. 3(a). This technique breaks the world into a set of voxels and counts in each voxel the number of hits and misses of ray-traced laser data. A hit corresponds to a return that terminates in a given voxel. A miss corresponds to a laser beam going through a voxel. The regions containing soft obstacles, such as vegetation, correspond to a small ratio of hits over misses. This allows traversable vegetation and noise to be filtered out. However, as can be seen in Fig. 3(a), the tree canopies correspond to a soft obstacle and would be partially filtered out based on this approach. In the context of object recognition, this form of segmentation would not be straightforward, but instead may form a useful feature for classification, once segmentation has been performed.

E. Ground Modelling via Plane Extraction

The approach developed in [7] for extracting mutli-resolution planar surfaces was tested in the context of this study. Its output is illustrated in Fig. 3(b). It involves discretising the world into two superimposed 2D grids of different resolutions. In our implementation, the two resolutions are set to one meter and twenty centimeters. Each grid cell in the two grids is represented by a plane fitted to the corresponding laser returns via least square regression. The least square error e_l is computed for a large plane (in the coarse resolution grid) and the least square error e_s is computed for the small plane (in the fine resolution grid). By comparing the values of e_l and e_s in overlapping planes, several types of regions can be identified: e_l and e_s are both small in sections corresponding to the ground; e_l is small while e_s is large in areas containing a flat surface with a spike (a thin pole for instance); both e_l and e_s are large in areas containing an obstacle. This method is able to identify the ground while not averaging out thin vertical obstacles (unlike an elevation map of type "Mean"). However, it is not able to represent overhanging structures. The latter case is addressed by the proposed algorithm.

F. Surface Based Segmentation

The method developed in [4] performs segmentation of 3D point clouds based on the notion of surface continuity. Surface continuity is evaluated using a mesh built from data. Such a mesh is illustrated in Fig. 3(d). The mesh is generated by exploiting the physical ordering of the measurements which implies that longer edges in the mesh or more acute angles formed by two consecutive edges directly correspond to surface discontinuities. This technique was applied to a portion of the data shown in Fig. 1. Fig. 3(c) presents the set of continuous surfaces (or 3D segments) extracted



(a) (b) (c) (d)

Fig. 3. (a) Map coloured by the ratio of hits over misses [3]. Hard obstacles such as walls are in red, with soft obstacles such as vegetation in blue. (b) Representation developed in [7]. Red corresponds to a large ratio e_l/e_s (see text), typical for objects, while the ground has a lower ratio. (c) The extracted continuous surfaces from the Surface Based Segmentation technique. (d) Two individual segments, a lamp and the ground.

by the algorithm. In Fig 3(d), two individual segments are presented. While this approach performs 3D segmentation, it does not identify the ground surface. The algorithm proposed here allows to jointly perform the two tasks.

G. Contributions

The novelty of this work is to present a terrain model which provides an explicit separation of the ground, segments objects in 3D and represents overhangs. None of the previously described algorithms can do all three tasks, while the proposed algorithm achieves this without a major increase in computation time. Objects are removed from a mean elevation map, providing a good model of the ground. The position and extent of objects are identified with a Min-Max elevation map, allowing a detailed voxel representation of objects to be built efficiently. The object model augments the ground model by identifying ground patches beneath overhangs.

III. The Hybrid Elevation Map Segmentation Algorithm

The Hybrid Elevation Map Algorithm is composed of two main processes: (1) the extraction of the ground surface using a mean elevation map; (2) the segmentation of the objects above the ground, using a min-max elevation map. This Hybrid algorithm is detailed in Algorithm 1, with parameters in Table I.

Name	Value
$gThresh$	0.5 ($\approx 27°$)
N_g	5
$hThresh$	20cm
$resGlobal$	40cm
$resLocal$	20cm
$minSegmentExtent$	10cm

TABLE I Algorithm 1 Parameters

These parameters are chosen *a priori* given the typical operating conditions encountered in the targeted applications: semi-urban environments (as shown in Fig. 1(b)) mapped for the navigation of Segways RPM 400 [1].

A. Ground Extraction

In Algorithm 1, line 1 computes a mean elevation map, with a grid resolution of $resGlobal$. This produces a map as in Fig. 2(a). The distorted obstacles are removed, which protrude from the smooth ground. This is detailed in Algorithm 2 and outputs the map in Fig. 4.

Input : $mapMean$, $resGlobal$, $gThresh$, N_g, $hThresh$
Output: gc

$gf \leftarrow$ `GenerateGradientField`($mapMean$) ;
$tf \leftarrow$ `ThresholdGradientField`(gf, $gThresh$) ;
$gc \leftarrow$ `ClusterGridCell`(tf) ;
$lc \leftarrow$ `IdentifyLargestCluster`(gc) ;
$gc \leftarrow$ `RmHighClusters`($mapMean$,gc,lc,N_g,$hThresh$) ;
$gc \leftarrow$ `CorrectGroundArtifact`($mapMean$, gc, $hThresh$);

Algorithm 2: `ExtractGroundSurface`

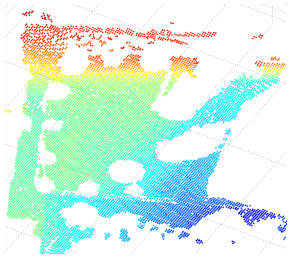


Fig. 4. Output of `ExtractGroundSurface` when applied to the data shown in Fig. 1. Higher sections are in red.

To remove obstacles that protrude from the mean elevation map, surface gradients are computed by the function `GenerateGradientField`. The gradients between a cell b and its neighbours are computed, with the largest retained as cell b's gradient.

A threshold is applied on gradients to identify obstacles (`ThresholdGradientField`), set to 0.5 in our implementation (a slope angle of 27°).

The cells identified as belonging to the ground are connected based on adjacency to form clusters of ground cells. This is implemented by the function `ClusterGridCell`: it traverses the ground grid and assigns the same cluster Id to ground cells which are in contact of each other. The function `IdentifyLargestCluster` is then run in order to find the largest set of connected ground cells. This cluster becomes a ground reference and is used to decide if other small clusters in the scene belong to the ground. A reference is required because locally smooth clusters may exist that are not part of the ground surface. Function `RmHighClusters` filters out these cases.

The final step is the correction of artifacts generated during the computations of the gradients. The cause of these artifacts is explained in Fig. 5(a). Such correction is implemented by the function `CorrectGroundArtifact` which proceeds as follows. For each non-ground cell b,

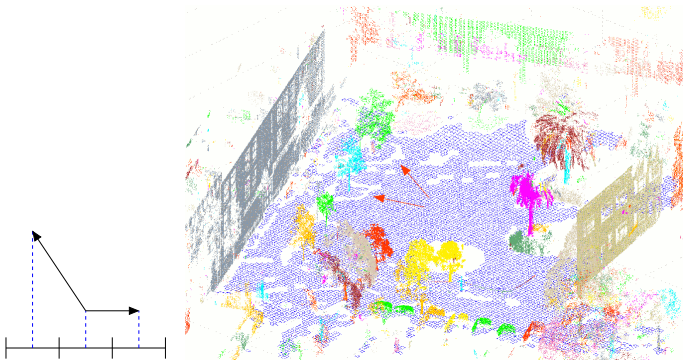
Input : pts
Output : $mapMean$, $listObject$
Parameters: $gThresh$, N_g, $hThresh$, $resGlobal$, $resLocal$, $minSegmentExtent$

$mapMean \leftarrow$ `GenerateMapMean(`pts, $resGlobal$`)` ;
$gc \leftarrow$ `ExtractGroundSurface(`$mapMean$, $gThresh$, N_g, $hThresh$,`)` ;
$mapMinMax \leftarrow$ `GenerateMapMinMax(`pts, $resGlobal$, $hThresh$`)` ;
$clusters \leftarrow$ `ClusterGridCell(`$mapMinMax$`)` ;
for $c \in clusters$ **do**
  $mapMinMaxLocal \leftarrow$ `GenerateMapMinMax(`c, $resLocal$, $hThresh$`)` ;
  $[voxelSet\ mapMean] \leftarrow$ `GenerateVoxelSet(`$mapMinMaxLocal$, $resLocal$, $mapMean$, gc, $hThresh$`)` ;
  $voxelClusters \leftarrow$ `ClusterVoxel(`$voxelSet$`)` ;
  $listObject \leftarrow$ `AugmentListObject(`$listObject$, $voxelClusters$, $minSegmentExtent$`)` ;
end

Algorithm 1. The Hybrid Elevation Map Segmentation Algorithm

it finds the neighbouring ground cells and computes their average height. If the b's own height is close (less than $hThresh$ apart), b is marked as a ground cell.

Since `CorrectGroundArtifact` modifies the set of ground cells, the operations on line 3 and 4 of Algorithm 2 need to be repeated on the corrected set of ground cells.



(a) (b)

Fig. 5. Ground artifacts. (a) Three cells, with their heights as vertical dotted lines. The arrows show the gradients. The middle cell is identified as containing a steep slope (and so is removed) while in fact it does belong to a flat area. (b) Mis-detection of ground cells. The two trees indicated by the red arrows are surrounded by blank areas which were incorrectly removed from the ground. This would prevent a path planner from finding a path under the canopy of the trees. These "holes" are not present in the output of Algorithm 2 (Fig. 8).

B. Object Segmentation

With the ground now identified, objects need to be accurately represented and segmented. This hybrid algorithm seeks to use the accuracy voxel grids provide, applied in a efficient manner that incorporates the ground model. Each object is to be represented by a set of voxels clustered together. In addition, as the height of the ground is known, the voxels of the ground below overhangs can be identified.

To begin with, a broad, "global" search for object regions is done using a low resolution "Min-Max" elevation map (as shown in Fig. II, implemented in line 3 of Algorithm 1). If the height range in a given cell is above a threshold $hThresh$, the cell is identified as an object. Occupied cells are clustered, as illustrated in Fig. 6(a).

For each cluster c, a local, detailed Min-Max elevation map is built. A voxel grid is then built upon this map to get a detailed representation of the region. Each vertical bar of the elevation map (the green bars in Fig. II) is divided into voxels. The voxels which do not contain any laser returns are disregarded. Voxels within one vertical bar are merged if they are in contact. The output of the function `GenerateVoxelSet` is illustrated in Fig. 6(b).

Voxels that belong to the ground are now identified using the previous ground model. The ground is then added back to the ground model (as can be seen in Fig. 8 under the canopy of the trees). Identifying ground voxels plays an important role in separating the objects during segmentation, detailed below.

The identification of the ground voxels is implemented in the function `GenerateVoxelSet` and proceeds as follows. For a given cell c, the process looks up the N_g closest ground cells in the grid $mapMean$ and uses the mean of their height as an estimate of the ground height. If the returns in the lowest voxel in c have an average height which is below $z_g + hThresh$, the voxel is marked as belonging to the ground.

Once a ground voxel is identified, the ground surface can be refined. Recall that at this stage c belongs to the local, detailed elevation map $mapMinMaxLocal$ and that the ground surface is represented by the global elevation map $mapMean$. To integrate into the global map the fact that a voxel from a local map belongs to the ground, the function `GenerateVoxelSet` finds the cell c_{global} in the grid $mapMean$ which is the closest to c. c_{global} is then updated: the mean height in the cell c_{global} is re-computed using only the returns that falls into the lower voxel in c. This is the point in the process at which the reconstruction of the ground under overhanging structures is performed. This approach involves a communication between local and global data structures by which object segmentation contributes to ground segmentation. Voxels identified as belonging to the ground can be seen in Fig. 6(b), as well as in Fig. 8.

With the detailed object voxels clearly identified as in Fig. 6(b), they can be clustered together to form full object segments (implemented by the function `ClusterVoxel`). Its output is a set of 3D segments (or clusters); as illustrated in Fig.6(c). `ClusterVoxel` proceeds by grouping contacting voxels into the same cluster. Voxels identified as belonging to the ground are interpreted as separators between clusters. This explains why an accurate estimation of the

ground height as well as an accurate identification of the voxels belonging to the ground is crucial to segmentation.

The last stage in the formation of the 3D segments is the identification of noisy returns. This is achieved as follows. The function `AugmentListObject`, in addition to growing the list of 3D segments, identifies the voxels which potentially contain noisy returns. Such voxels are the ones which satisfy the two following conditions. (1) They belong to a segment which is not in contact with the ground. (2) The extent of the segment, in each of the three directions defined by the reference frame axes, is below the threshold $minSegmentExtent$ (set to 10cm). Once the noisy returns are identified, they are withdrawn from the map. The set of returns identified as noisy in the data shown in Fig. 1 is displayed in Fig. 7.

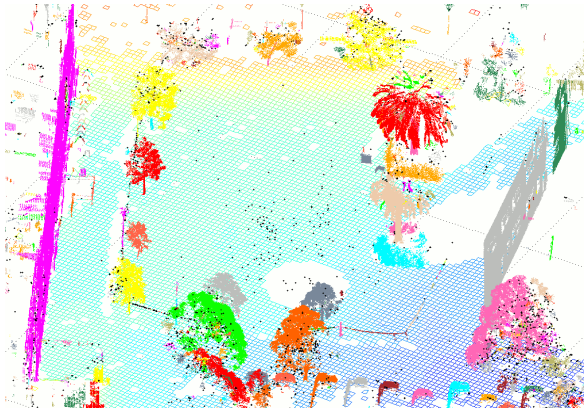


Fig. 7. The set of returns identified as noisy by the function `AugmentListObject`. Each noisy return is indicated by a black dot. The background map is the same as the one shown in Fig. 8.

C. The Hybrid Terrain Model

An example of a model produced by Algorithm 1 is shown in Fig. 8. A number of aspects are now discussed.

First, it can be seen that the ground is properly reconstructed under the canopy of the trees in the central area as well as under the rope indicated by the black arrow. This is achieved by the function `GenerateVoxelSet`; as detailed in Sec. III-B. The building on the left side of the scene is correctly clustered into one segment. The building on the right side of the scene is identified as two distinct segments. This is due to one of the palm trees occluding the building from the sensor; this results in the vertical unobserved band which can be seen on the furthest end of the building's wall. The majority of the trees are correctly segmented. The palm tree furthest in the background has been segmented into two components: the trunk and the palms. As in the case of the building on the right side of the scene, missing data due to occlusions causes over-segmentation. Since the palms of the tree partly cover the trunk, the junction between the trunk and the top part of the tree is not observed and Algorithm 1 identifies two different segments. The cars in the foreground are correctly segmented, apart from the case shown in Fig. 6(c). The street lamps of the left side of the scene are also correctly segmented. The bike racks along the left building (indicated by the white arrow), are correctly retained in the map and not filtered out as noise (unlike in an elevation map of the type "Mean"). Due to small number of returns however, these bike racks are over-segmented (more than one colour per bike rack).

Since the data structure $listObject$ contains the original data organised into segments (note that the voxel based representation is only intermediate), fine details are conserved; for instance, the frame of the windows on the building on the right, or the rope at the border of the grassy area in the bottom part of the image.

IV. Performance Evaluation

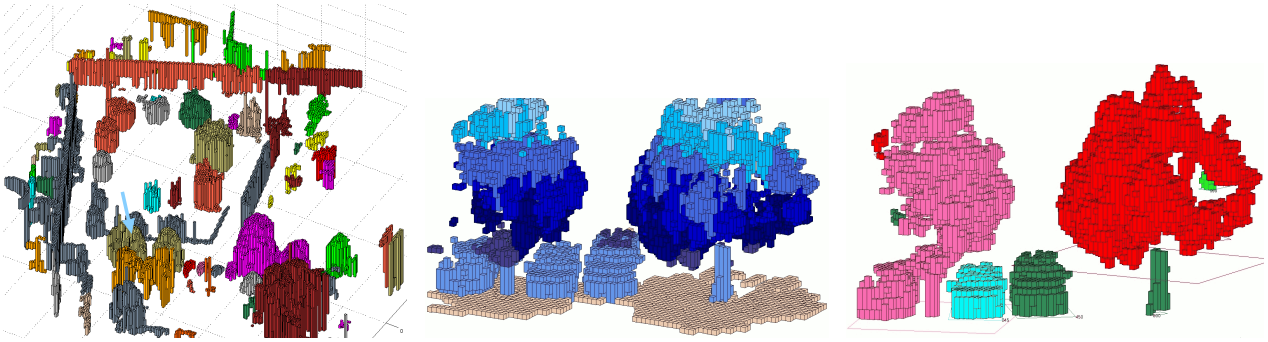
This section presents an evaluation of the Hybrid Elevation Map Segmentation Algorithm as well as a comparison to the mapping techniques reviewed in Sec. II. The sets of results are organised into two tables. Table II presents a comparison of the various methods evaluated in this work. Tables III presents a break down of the computation times for the steps in the Hybrid algorithm.

To begin with, the results in Table II are analysed. The first three columns indicate whether each of the algorithms presented in Sec. II achieve the following tasks: (1) explicit extraction of the ground (as opposed to extracting 3D surfaces without explicitly specifying which ones belong to the ground); (2) representation of overhanging structures such as the canopy of trees; (3) full 3D segmentation of objects. It can be seen from Table II that the proposed algorithm is the only one which performs all of these three tasks.

The fourth column provides computation times for each of the algorithm applied to the dataset shown in Fig. 1. In each case, the resolution of the main grid is set to 40 cm. The mention "Matlab" indicates that the algorithms were tested using a Matlab implementation. The computation times show that the proposed algorithm is the second slowest. As implied by the three first columns of the table, it is the only approach that jointly performs ground extraction, overhang representation and 3D segmentation. Performing these three tasks logically require more computations.

The values provided in this last column of Table II are Root Mean Square Errors (RMSE). The RMSE is the square root of the mean square error. The error distance used is adapted to each model in the following way: in the voxel based models, the error is given by the distance between a laser return and the centre of the voxel it belongs to; in the other models, the error is given by the difference between the height of the estimated surface and the measured height.

The RMSE values were computed using the samples contained in a hundred by hundred meter area centred at the middle of the data set displayed in Fig. 1. The RMSE values show that the proposed algorithm achieves the second best performance after the Surface based Segmentation algorithm. The modelling of the ground performed by the Hybrid algorithm explains the difference with the Surface based Segmentation approach: the ground model is produced by averaging the height of the returns falling onto the ground which results in a small but non zero RMSE value. On the whole, the proposed approach generates an RMSE value at



(a) (b) (c)

Fig. 6. Segmentation of the objects above the ground. (a) Illustration of the output of `ClusterGridCell` when applied to the map shown in Fig. II. One colour corresponds to one cluster. (b) Illustration of the output of `GenerateVoxelSet` when applied to the cluster indicated in (a) by the light blue arrow. Colours are mapped to height. The voxels identified as belonging to the ground are indicated by a beige colour. (c) Illustration of the output of the function `ClusterVoxel` when applied to the set of voxels displayed in (b). One colour corresponds to one 3D segment (or cluster). The rectangles indicate the footprint of the various segments. The segmentation is correct with respect to the two cars in the middle. On the other hand, the tree and the car on the left are gathered in the same segment due to the proximity of a branch of the tree to the roof of the car. On the right, the tree is represented by two segments due to the lack of data which prevents the algorithm from connecting the trunk of the tree to its foliage.

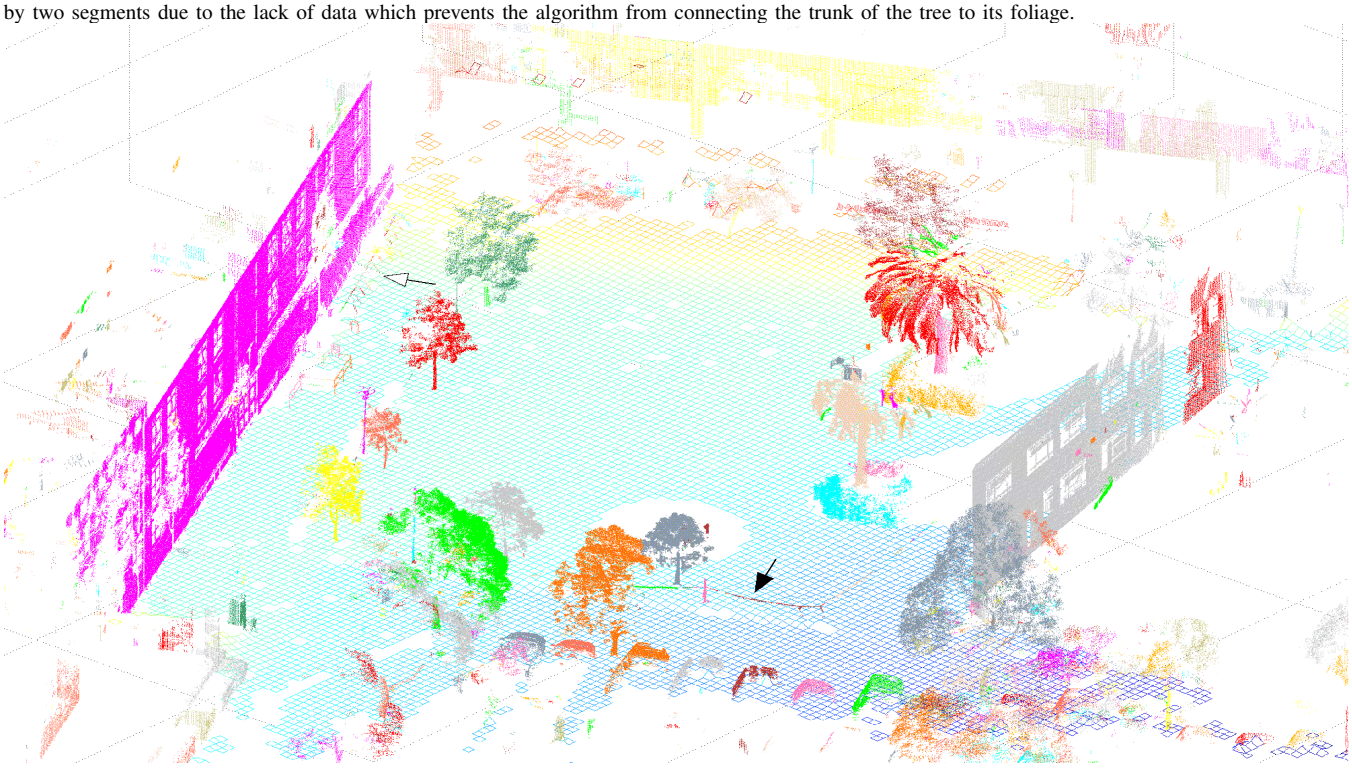


Fig. 8. Segmentation obtained with Algorithm 1 applied to the data shown in Fig. 1. The ground is represented by a mesh in which the colours are mapped to height; blue corresponds to lower areas and red to higher areas. The ground mesh represents the data structure $mapMean$ produced by Algoritm 1. The segmentation of the objects above the ground is also colour coded: one colour corresponds to one 3D segment. Each coloured 3D point cloud is an element of the data structure $listObject$. The dataset contains a total of 1,733,633 points. The black and white arrows indicate a rope and a bike rack, respectively.

Algorithm	Ground Extraction	Overhang Representation	3D Segmentation	Computation Time	RMSE
Map "Mean" [6]	no	no	no	1.77 min (Matlab)	1.725 m
Map "Min-Max" [8]	yes	no	no	1.76 min (Matlab)	3.453 m
Map Multi-level [5]	yes	yes	no	8.12 min (Matlab)	1.324 m
Ground Modelling via Plane Extraction [7]	yes	no	no	13.13 min (Matlab)	1.533 m
Volumetric Density Map [3]	no	no	no	10.01 min (Matlab)	0.297 m
Surface Based Segmentation [4]	no	yes	yes	$\approx$ 10 hours (Matlab)	0 m
Hybrid Map	yes	yes	yes	14.28 min (Matlab)	0.077 m

TABLE II PERFORMANCE EVALUATION

least two orders of magnitude smaller than the other models that explicitly represent the ground (elevation maps of the type "Mean", "Min-Max" and "Multi-level"). This is due to the ability of the hybrid approach to separate the objects above the ground.

Detailed computation times of each of the function called by Algorithm 1 are given in Table III. These values were also obtained using the data shown in Fig. 1. For the last three functions, the indicated time is the weighted average of the computation time per cluster (clusters generated by the function `ClusterGridCell`). The weights are the number of voxels generated for each cluster (by the function `GenerateVoxelSet`). Using the numbers of voxels as weights avoids underestimating computation times in a dataset containing a dominating number of small clusters.

The elevation map of type "Mean" and "Min-Max" were also implemented in C++: computation time is in that case reduced from 1.7 minutes to 0.9 second for the dataset shown in Fig. 1. This suggests that the implementation of the proposed algorithm for real-time application is feasible given that standard elevation maps are generated in less than a second on a set of 1.7 million points.

Function	Computation Time (Matlab)
`GenerateMapMean`	90 secs
`ExtractGroundSurface`	71 secs
`GenerateMapMinMax`	92 secs
`ClusterGridCell`	15 secs
`GenerateVoxelSet`	11 secs (average per cluster)
`ClusterVoxel`	13 secs (average per cluster)
`AugmentListObject`	9 secs (average per cluster)

TABLE III HYBRID ALGORITHM COMPUTATION TIMES

To also demonstrate that the 3D model generated by the Hybrid Algorithm can be applied to path planning, trajectories were generated in the map shown in Fig. 8 using a Wave-Front Planner [2]. The resulting trajectory is shown in Fig. 9. By separately classifying the terrain, the Hybrid algorithm's terrain model reduces the complexity and of the path planning operations. It provides high-resolution terrain navigation and obstacle avoidance, particularly with overhangs, yet allows for planning operations to be performed efficiently in a reduced (2D) workspace. When used with the Hybrid Algorithm the planner takes advantage and plans on the segmented ground model. It is also able to check for appropriate clearance around obstacles with complex geometry and hence navigate through regions with overhanging features more efficiently. In a tour of ten randomly selected goal points (*c.f.* Fig. 9) this reduced the length of travel paths by 11%.

V. CONCLUSION AND FUTURE WORK

This work has introduced an algorithm to perform segmentation of 3D point clouds. The algorithm extends two types of elevation maps: elevation maps of type "Mean" and elevation maps of type "Min-Max". The approach was shown to be unique because it jointly performs ground extraction, overhang representation and 3D segmentation. In addition, it was shown that amongst elevation models it provides the best fit to the data and that it can be applied to path planning.

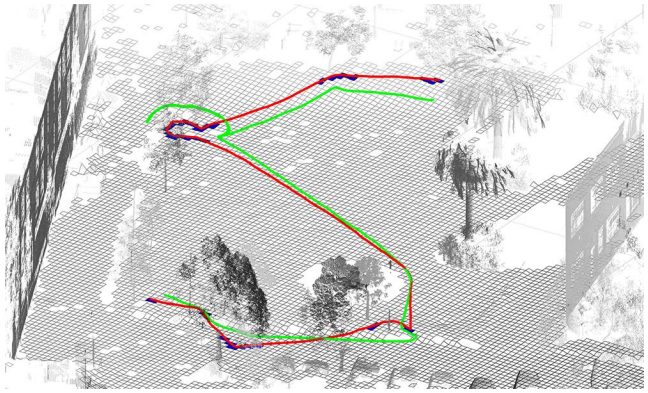


Fig. 9. Sample trajectory calculated based on the segmentation. The Segmentation Pipeline model allows a motion planner to factor 3D terrain features. The red line shows a trajectory automatically generated from a number of user-specified goal locations situated under overhangs (e.g., tree canopies). The blue cells are those containing an overhang and traversed by the trajectory. For comparison, the green line shows a path generated using an Elevation Map Mean (to points nearest the user selected points when these are under overhangs). This path is longer and gets confused by sparse terrain features (such as the rope).

Future work will focus on several aspects including testing the method on laser data provided by SICK and Velodyne sensors, real-time implementation and classification of the segmented point clouds.

VI. ACKNOWLEDGEMENTS

This research was undertaken through the Centre for Intelligent Mobile Systems (CIMS), and was funded by BAE Systems as part of an ongoing partnership with the University of Sydney. It was also supported by the Rio Tinto Centre for Mine Automation and the ARC Centres of Excellence Programme funded by the Australian Research Council and the New South Wales State Government. The authors would like to thank Matthew Johnson-Roberson for useful discussions.

REFERENCES

[1] Segway Robotics. http://rmp.segway.com.
[2] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, 2005.
[3] A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, T. Pilarski, P. Rander, S. Thayer, N. Vallidis, and R. Warner. Toward reliable off road autonomous vehicles operating in challenging environments. *International Journal of Robotics Research (IJRR)*, 2006.
[4] N. Melkumyan. *Surface-based Synthesis of 3D Maps for outdoor Unstructured Environments*. PhD thesis, University of Sydney, Australian Centre for Field Robotics, 2008.
[5] P. Pfaff and W. Burgard. An efficient extension to elevation maps for outdoor terrain mapping and loop closing. *International Journal of Robotics Research (IJRR)*, 26(2):217–230, February 2007.
[6] B. Siciliano and O. Khatib. *Springer handbook of robotics*. Springer, 2008.
[7] R. Simmons, L. Henriksen, L. Chrisman, and G. Whelan. Obstacle avoidance and safeguarding for a lunar rover. In *AIAA Forum on Advanced Developments in Space Robotics*, 1996.
[8] S. Thrun and *et al.* Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.