

A Pipeline for the Segmentation and Classification of 3D Point Clouds

B. Douillard, J. Underwood, V. Vlaskine, A. Quadros, and S. Singh

Abstract. This paper presents algorithms for fast segmentation of 3D point clouds and subsequent classification of the obtained 3D segments. The method jointly determines the ground surface and segments individual objects in 3D, including overhanging structures. When compared to six other terrain modelling techniques, this approach has minimal error between the sensed data and the representation; and is fast (processing a Velodyne scan in approximately 2 seconds). Applications include improved alignment of successive scans by enabling operations in sections (Velodyne scans are aligned 7% sharper compared to an approach using raw points) and more informed decision-making (paths move around overhangs). The use of segmentation to aid classification through 3D features, such as the Spin Image or the Spherical Harmonic Descriptor, is discussed and experimentally compared. Moreover, the segmentation facilitates a novel approach to 3D classification that bypasses feature extraction and directly compares 3D shapes via the ICP algorithm. This technique is shown to achieve accuracy on par with the best feature based classifier (92.1%) while being significantly faster and allowing a clearer understanding of the classifier's behaviour.

1 Introduction

The natural operating environment of all mobile systems is three-dimensional. Models of the environment are often simplified to 2D to reduce complexity and increase computational feasibility, however, the assumptions required to achieve this come at a cost of reduced accuracy for all but the simplest of scenes. Three dimensional representations require fewer assumptions in general, allowing the most accurate representations of the world. Dense 3D range sensing offers increased perceptual ability. For example, it complements vision data by addressing illumination variation from monocular imagery and sparse reconstruction of geometry from stereo. However, it presents a

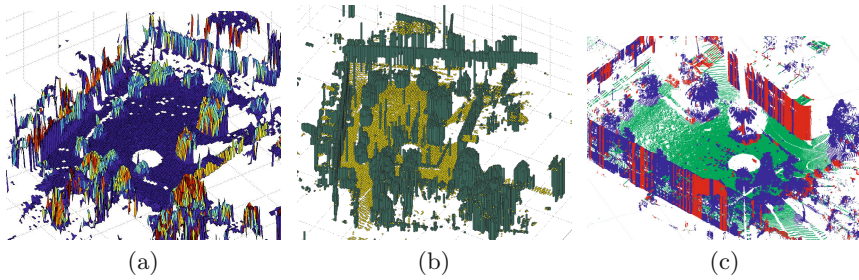


Fig. 1 Three examples of elevation models constructed from a Riegl scan. (a) Mean Elevation Map. Colours are mapped to the standard deviation of the height in each cell. Larger standard deviations correspond to the red end of the colour range. (b) Min-Max Elevation Map. Yellow cells are cells identified as belonging to the ground (i.e., the difference between the maximum and minimum height of the returns in the cell is below the threshold $hThresh$). Green cells are identified as containing an object. (c) Map produced by the approach described in [15]. The colour green corresponds to the class “ground”, red to the class “vertical surface” and blue to the class “vertical surface containing a gap”. This display is organised similarly to Fig. 4 in [15]. 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” (“vertical segments” is part of the terminology used in [15]).

voluminous flood of data (see Fig. 1), which is challenging to process in real time. Recent hardware developments (e.g. by Velodyne, Riegl, Ibeo) suggest a proliferation of 3D sensing from static and moving platforms. Utilising this data for tasks such as mapping and decision making requires quality conditioning, for which robust segmentation is critical. This not only leads to, but enhances classification, giving semantic meaning to nondescript segments.

Towards this, the paper presents two algorithms for segmentation and classification that form a processing pipeline that goes from raw 3D data, via 3D segmentation, to classification. The Segmentation Pipeline combines a mean elevation map representation for the ground and a voxel based map for 3D objects in the scene that yields a compact model, sieving the ground from objects. Segmented objects are then classified by a novel feature-less 3D method that uses ICP to align unidentified objects against a set of candidate template shapes.

2 3D Segmentation

In this section, a novel algorithm for joint terrain estimation and 3D object segmentation is presented. Several terrain modelling methods are evaluated, and experiments are presented to compare the new algorithm against current

approaches. The effectiveness of the algorithm is also tested in the context of tasks such as path planning and data alignment.

2.1 *Terrain Modelling*

A generalised approach is taken based on the following terrain modelling techniques from the literature.

In **Elevation Map Mean**, also referred to as a $2^{1/2}$ D model, the terrain is represented by a grid in which each cell contains only one height, commonly the average height of the sensor data located in each grid cell [17]. This method will be referred to as an elevation map of type “Mean” (see Fig. 1(a)). Since this representation generates smooth surfaces by filtering out noisy returns (averaging is akin to low-pass filtering), it was chosen as the basis for the ground part of the model.

Elevation Map Min-Max computes the difference between the maximum and the minimum height of the sensor data in a cell [19]. A cell is declared occupied if its height difference exceeds a pre-defined threshold. 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 characterised by a strong gradient and can be identified as occupied. Its output is illustrated in Fig. 1(b).

Elevation Map Multi-Levels represents multiple levels of elevation and handles overhanging structures to allow the generation of large scale 3D maps by recursively registering local maps [15]. The output it produces is illustrated in Fig. 1(c). While the algorithm proposed here also discretises the vertical dimension in order to capture overhanging structures, it differs from the work in [15] in that the initial segmentation explicitly forms the two classes “ground” and “object above the ground”, where the ground includes the regions underneath overhanging objects. The set of classes chosen in [15] is different and may not facilitate segmentation. As can be seen on the buildings on the sides of the scene, windows or other causes of sparsity in the data imply that sections of walls are identified as belonging to the class “vertical surface containing a gap” (blue). Second, the ground in [15] is not used as a reference for vertical height. We suggest that such a reference is necessary when reasoning on the variation of height over a grid: the variation of height may be locally small and correspond to a flat surface above and distinct from the ground. These are handled by the Segmentation Pipeline.

Various other terrain modelling techniques were tested (ref. Table 1). They include local plane fitting [18], volumetric density mapping (which discretises the space in 3D and reasons about the voxels via ray-tracing) [8] and surface based segmentation (which stitches 3D surfaces obtained directly in the sensor frame into separate objects) [12]. A recently proposed 3D segmentation algorithm based on a convexity criterion [13], while fast, does not capture overhanging structures.

2.2 The Segmentation Pipeline

The processing pipeline applies the strategy of performing segmentation prior to classification, as argued for in the context of vision applications [11]. Other methods classify single laser returns first, based on the type of feature presented in Sec. 3, see for instance [1, 20, 14].

The Segmentation is composed of two main processes: (1) the extraction of the ground surface (lines A1.1 and A1.2 in Algorithm 1); (2) the segmentation of the objects above the ground (lines A1.3 and A1.4). The name 'Segmentation Pipeline' refers to the flow of data through the algorithm. Parts of the pipeline are processed *incrementally*, meaning that the output is derived as a function of the previous output and the new data, and other parts are *iterative*, meaning that they are repeatedly recalculated as a function of all current data.

```

Input      : pCloud
Output     : ground, voxelClusters
Parameters: res, sdThresh, hThresh
1 voxelGrid ← FillVoxelGrid(pCloud, res) ;
2 ground ← ExtractGroundSurface(voxelGrid, sdThresh, hThresh) ;
3 nonGroundVoxels = voxelGrid - ground ;
4 voxelClusters ← ClusterVoxelGrid(nonGroundVoxels) ;

```

Algorithm 1. The Segmentation Pipeline

The first step of the algorithm places new sensor data in a database called a voxel-grid (line A1.1). Data are stored in cubic voxels for efficient retrieval in 3D, and for batch processing per voxel (grid resolution is denoted *res* and set to 0.2 m). The data may additionally be filtered in some way, for example with a policy that considers the maximum data density, age, entropy or uncertainty. For this paper, all of the data are kept.

The second step (line A1.2) separates the data in two categories: ground and non-ground (details in Algorithm 2). Points are considered in batches, defined by their membership in a single cubic voxel in space. A voxel is considered to contain ground data if two conditions are met: (1) the voxel must be a member of the lowest (in height) set of adjacent non-empty voxels in a vertical column (i.e. not part of an overhang); (2) the 3D points stored in that set of voxels must have a low residual ($< sdThresh$, 0.1 m) when fitted to a plane. This is expressed in lines A2.1 to A2.10. Note that line A2.3 is an efficient test for horizontal planarity in the case where the voxel set is thicker than two. An incremental algorithm is used to calculate the standard deviations, so lines A2.1 to A2.10 are all implemented incrementally (calculated efficiently for each individual new point). In line A2.11, ground cells are clustered by 2D adjacency to form partitions. From line A2.12 on, starting with the largest partition as the ground surface, the remaining partitions are

incorporated if they are within a height tolerance ($hThresh$, 0.2 m for this paper) of the nearest ground surface partition. This last ground selection criteria (from line A2.12) is the most expensive part of the whole pipeline in Algorithms 1 and 2, and other criteria can be used instead. For example, if after clustering on line A2.11, only the clusters over a size threshold are chosen, the pipeline cycle can be processed approximately 34 times faster, presenting a tradeoff between accuracy and speed.

Finally, all of the voxels in the database that contain data, but were not considered to be part of the ground surface are now semantically labelled as non-ground (line A1.3). Importantly, this includes overhanging regions of occupied space above the identified ground surface. These non-ground voxels are then geometrically segmented (line A1.4) by partitioning them according to 3D adjacency. The first half of the algorithm from line A1.1 to line A2.10 can be done *incrementally*, as sensor data become available. The second half (line A2.11 on) is done iteratively, at a rate determined by processing power (approximately 0.5Hz for Velodyne scans clipped to 30 m range, on a 3 GHz PC), or when needed for an application such as path planning.

```

Input : voxelGrid, sdThresh, hThresh
Output: ground
1 for column(i, j) ∈ voxelGrid do
2   RemoveOverhangingVoxels() ;
3   if More than two voxels left; continue;
4   cm ← CentreOfMass(column(i, j)) ;
5   pCloudi,j ← PointsInColumn(voxelGrid, i, j) ;
6   stddev ← VerticalStdDev(cm, pCloudi,j) ;
7   if stddev < sdThresh then elevation(i, j) = cm ;
8   isFlat(i, j) = true ;
9   else isFlat(i, j) = false ;
10 end
11 fPartitions ← FindFlatPartitions(isFlat, elevation) ;
12 lPartition ← FindLargestPartition(fPartitions) ;
13 ground ← lPartition ;
14 for partition ∈ fPartitions and ! ∈ ground do
15   {f, g} ← FindNearestCells(partition, ground) ;
16   if |elevation(f) − elevation(g)| < hThresh then
17     | ground ← {ground, partition} ;
18   end
19 end

```

Algorithm 2. ExtractGroundSurface

2.3 Experiments

2.3.1 Segmenting Riegl Data

An example of a model produced by Algorithm 1 is shown in Fig. 2. By considering the full 3D geometry, the ground is properly reconstructed under the canopy of the trees as well as under a fine rope (indicated by the black arrow). Similarly, the street lamps (on the left side) and the bike racks (indicated by the white arrow) are correctly retained in the map. Separate processing (with an Elevation Map Mean) would have removed these features as noise. However, occlusions cause some anomalies. The building on the left side is correctly clustered into one segment; however, the building on the right side is identified as two distinct segments because one of the palm trees occluding the building from the sensor, resulting in the vertical unobserved band. The majority of the trees are correctly segmented; but, the tree furthest in the background has been segmented into two components (trunk and palms) because the palms of the tree partly cover the trunk and the junction between them is not observed, thus Algorithm 1 identifies two segments.

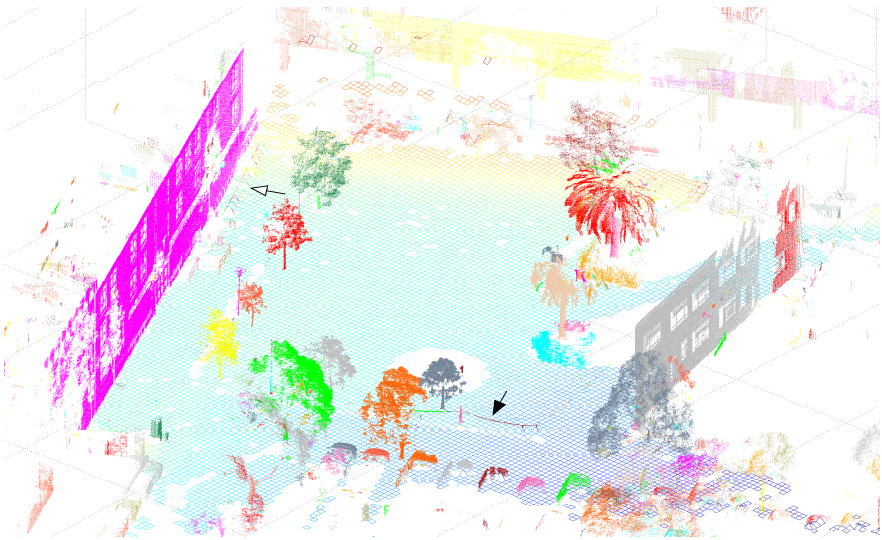


Fig. 2 Segmentation obtained with Algorithm 1 applied to the data shown in Fig. 1. The ground is represented by a mesh in which the colours are mapped to height; blue corresponds to lower areas and red to higher areas. The ground mesh represents the data structure *mapMean* produced by Algorithm 1. The 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 *voxelClusters*. The dataset contains a total of 1,733,633 points. The black and white arrows indicate a rope and a bike rack, respectively.

Table 1 Performance Evaluation

Segmentation Algorithm	Ground Extraction	Overhang Representation	3D Segmentation	Computation Time	RMSE
Map "Mean" [17]	no	no	no	0.92 seconds (C++)	1.725 m
Map "Min-Max" [19]	yes	no	no	0.92 seconds (C++)	3.453 m
Map Multi-level [15]	yes	yes	no	8.12 min (Matlab)	1.324 m
Ground Modelling via Plane Extraction [18]	yes	no	no	13.13 min (Matlab)	1.533 m
Volumetric Density Map [8]	no	no	no	36.04 seconds (C++)	0.297 m
Surface Based Segmentation [12]	no	yes	yes	≈ 10 hours (Matlab)	0 m
The Segmentation Pipeline	yes	yes	yes	30 seconds (C++)	0.077 m

The behaviour of the segmentation algorithm relative to state-of-the-art methods is compared in Table 1. The first three columns summarise their respective processing pipelines across the following points: (1) explicit extraction of the ground; (2) representation of overhanging structures (such as tree canopies); and (3) full 3D segmentation of objects. The Segmentation Pipeline is the only one which performs all three of these tasks. Computation times for each of the algorithms applied to the same dataset with a grid resolution (res) of to 20 cm is shown ("Matlab" or "C++" indicates the implementation type). Accuracy is noted by the Root Mean Square Errors (RMSE). That is, for voxel based models, the error given by the distance between a laser return and the centre of the voxel it belongs to, and for other models, the error given by the difference between the height of the estimated surface and the measured height.

2.3.2 Segmenting Velodyne Data

In this experiment a set of 50 Velodyne scans are aligned using the Iterative Closest Point (ICP) algorithm (detailed in Sec. 4.1) in two different ways: (1) directly aligning consecutive scans, (2) segmenting and only using the larger segments for alignment, using the Segmentation Pipeline. The larger segments are those with a square root eigenvalue above a threshold of 2 m.

The resulting two alignments are compared in terms of their sharpness, where sharpness is measured as follows: the space is overlaid with a 3D grid (the same grid is used for both aligned sets) and the number of occupied 3D cells is counted. The lower the number of occupied cells the sharper the point cloud. With a resolution of 10 cm, the point cloud obtained using only the 3D segments for alignment is 7.0% sharper than the other point cloud. The corresponding point cloud is shown in Fig. 3(a). For the 50 tests, the Pipeline component averaged 2.05 seconds of processing per scan.

2.3.3 Segmentation for 3D Path Planning

The Segmentation Pipeline works as part of a system. A shortest-path planning experiment shows that the model has sufficient accuracy and speed for driving robot motion. In this case, trajectories for an eight-goal tour were generated using a potential field approach (namely the Wave-Front algorithm [3]) using the terrain models from the aforementioned approaches. The proposed

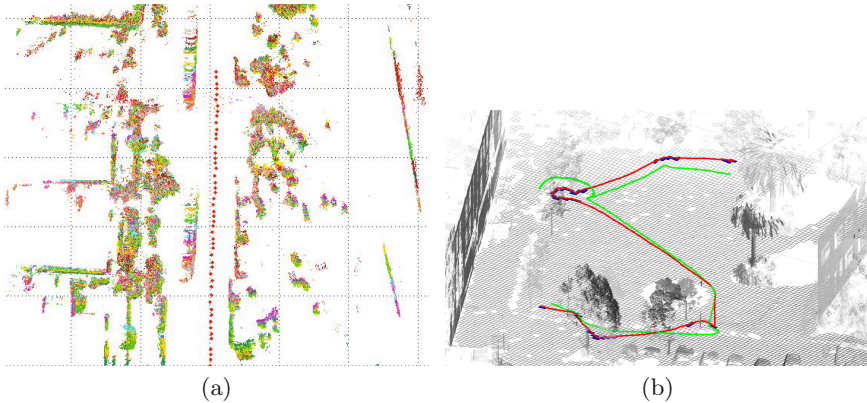


Fig. 3 (a) Top view of an example of alignment of 50 Velodyne scans acquired in a suburban area (the corresponding dataset is described at [4]). One colour corresponds to one scan. The red dots indicate the estimated (via ICP) trajectory of the platform carrying the Velodyne sensor. In this sequence, it can be seen that the experimental platform drove by two other vehicles which were moving in the opposite direction. (b) 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).

model reduced the total path length by 8% (see Fig. 3(b)). While other planners (e.g., RRTs) and tour reordering could have been performed, this would not necessarily give shorter routes and was beyond the scope of the experiment. By separately classifying the ground, the Segmentation Pipeline terrain model provides high-resolution terrain navigation, reduces the complexity of the path planning (via operations over a segmented ground model), simplifies collision checking (via a reduced workspace) and affords more informed processing (by distinguishing overhangs from obstacles). It also checks for appropriate clearance around obstacles with complex geometry (e.g., routing under instead of around the rope).


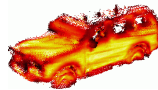
3 3D Features for Classification

The previous section introduced an algorithm for segmenting 3D point clouds. The next stage is to classify the individual segments, yielding a semantic labelling of the map. This section discusses a number of approaches for building 3D descriptors for classification and presents a set of experiments to compare

Table 2 Left: PCA features definition. Right: examples of PCA features.

Feature Name	Explanation	Measure
Surfaceness	$\lambda_0 \approx \lambda_1 \gg \lambda_2$	$\lambda_1 - \lambda_2$
Linearness	$\lambda_0 \gg \lambda_1 \approx \lambda_2$	$\lambda_0 - \lambda_1$
Scatterness	$\lambda_0 \approx \lambda_1 \approx \lambda_2$	λ_2

Feature: Linearness	Surfaceness
Radius: 0.2 m	0.2 m
Sensor: Riegl (fixed)Sick Laser (moving platform)	

them. The next section introduces a feature-less approach to the classification of 3D point clouds.

Principal Component Analysis (PCA) applied locally in 3D point clouds provides the saliency features described in Table 2; where the λ_i s are the eigenvalues ($\lambda_0 \geq \lambda_1 \geq \lambda_2$) of a subset of points in a local spherical region [10]. Examples of PCA features are shown in Table 2. The plots show the magnitude of the Linearness and Surfaceness features for a tree and a car respectively (colours are mapped to feature value) for a given spherical radius about each point.

To combine the results from all the points and perform classification, the whole point cloud is firstly voxelised. Each voxel summarises the encompassed points by providing the minimum, maximum and average feature values. The size and number of voxels is kept constant over all objects, and the object is aligned at the centre of the voxel grid for consistency. This implementation used 40 cm cube voxels, with a total of 17,000 voxels in the grid. The results from all voxels are then enrolled into a high dimensional feature vector that fully describes the object. Classification is then based on K Nearest Neighbours (KNN), and results are presented in Sec. 3.1.

Moment grids are a detailed local shape descriptor, introduced by Bosse and Zlot [2] for finding matching keypoints in place recognition. In the context of this study they are applied to object classification. Firstly a number of salient keypoints from the full point cloud are selected. A local grid is placed at each keypoint, aligned with the local principle axes to make it invariant to rotation and translation. This implementation used a $3 \times 3 \times 3$ grid with a total size of 80 cm, comprised of 27 overlapping 40 cm cubic cells. For each grid cell, the points encompassed are used to calculate a number of descriptors (11 in this implementation) based on moments up to the second order. The descriptors from each cell are combined in a vector (resulting in a dimensionality of 297). This vector provides a detailed spatial description of the keypoint region.

In the previous section, local PCA features across the object were combined in a 17,000 voxel grid. The much higher dimensionality of moment grids means this is not a feasible approach without dimensionality reduction. Instead, keypoints from labelled point clouds are placed in feature space.

Classification of new keypoints is done by finding the nearest labelled keypoints in feature space using KNN. Keypoint selection can be performed using an even subsample [2].

The Spin Image was introduced by Johnson *et al.* [5]. A Spin Image is built by first creating a 3D surface mesh in which each point is a vertex. Matching two objects requires computing the description at each vertex of the associated mesh; matching mechanism which allows to take into account multiple view points. In our implementation the surface mesh is obtained by voxelisation of the point cloud (with a voxel size of 20 cm); each vertex of the mesh then corresponds to the centre of one voxel. Classification is based on KNN (as in [6]).

The Spherical Harmonic Descriptor is a global feature, that is, a descriptor representing the whole point cloud, unlike the three previous descriptors which are local features. The Spherical Harmonic Descriptor was introduced by Kazhdan *et al.* [7] with the aim of addressing the angular registration problem. One of the main challenges in 3D shape matching arises from the requirement of matching similar objects which appear different due to different 3D orientations. The Spherical Harmonic Descriptor avoids this issue since it is invariant to 3D rotations. On the other hand, the proposed approach attempts to explicitly address the angular registration problem by means of the ICP algorithm. Once the Spherical Harmonic Descriptor is computed, classification is based on KNN (as in [7]).

3.1 Experiments

The set of 3D descriptors mentioned above were compared in the context of classification using the dataset presented in Fig. 4(a). The corresponding results are given in Table 3.

In the first row of the table, Global PCA refers simply to the eigenvalues computed for the full point cloud, that is, the feature vector contains only three dimensions. The associated classification performance provides a baseline. Since classification is based on KNN, the following numbers of neighbours were systematically tested: 1, 11, 101, 1001. The second column of Table 3 gives the range of accuracies achieved as well as the corresponding number of neighbours. The precision and recall values correspond to the best

Table 3 Classification Performance

Feature	Accuracy	Precision (in %) \ Recall (in %)					Feature Computation Times				
		Car	Pole	Tree	Wall	Mean	Std Deviation				
Global PCA Features KNN	50.0% (K=101) to 68.4% (K=1)	62.5	62.5	50.0	40.0	73.7	73.7	71.4	83.3	3 ms	6 ms (Matlab)
Global Grid PCA Features (0.2 m) KNN	44.7% (K=11) to 63.2% (K=1)	87.5	58.3	25.0	25.0	63.2	70.6	57.1	80.0	77 ms	266 ms (C++)
Keypoints PCA Features (0.2 m) KNN	61.4% (K=1) to 64.1% (K=11)	0.0	nan	0.0	nan	100	61.3	85.7	75.0	103 ms	379 ms (C++)
Spin Image KNN	63.2% (K=1001) to 92.1% (K=1)	87.5	100	75.0	75.0	94.7	94.7	100	95.7	5.5 min	11 min (Matlab)
Spherical Harmonic Descriptor KNN	50.0% (K=1001) to 84.2% (K=1)	100	88.9	75.0	60.0	78.9	93.8	85.7	75.0	0.7 s	0.8 s (Matlab & C)
Keypoints Moments Grid KNN	51.3% (K=1001) to 61.5% (K=1)	0.0	nan	0.0	nan	94.7	69.2	85.7	46.2	3.9 s	7.9 s (C++)

accuracy of the classifier. The computation times are indicated together with the type of implementation used.

Global PCA does surprisingly well for only three features, although the classes in this experiment are well differentiated by global shape. Local PCA features in a grid results in one of the worst results. The variation within each class is large, with trees and walls varying greatly in size, resulting in different alignments within the grid. The pole may be too small for the grid to form an adequate description. Keypoint PCA performs similarly, as does moment grid keypoints. This is surprising given that moment grids, with 297 dimensions describing a keypoint, give an overall classification accuracy similar to the 3 dimensional local PCA descriptor. Both PCA and moment grid keypoint approaches fail completely in car and pole classification. This indicates that for both these feature spaces, car and pole keypoints are not distinct, occurring too close to the more numerous tree and wall keypoints.

In contrast, the Spin Image does the best, with a similar keypoint matching working effectively. Trees and walls are accurately classified despite having a wide variation in shape. The global Spherical Harmonic Descriptor does worse on trees and walls for this reason, but gets comparable results on the more consistent cars and poles. These results will be compared in the next section to a feature-less classification technique.

4 Feature-Less 3D Classification

This section presents a feature-less approach to 3D classification. A key component is the exploitation of the context of the ground model provided by The Segmentation Pipeline. The previous section described an approach to classification which consists of computing local features to encode the local shape of objects. Simply due to the scale at which these features are computed, they are not easily interpretable. For instance, the harmonic coefficients forming the spherical harmonic descriptor do not allow a straightforward interpretation which limits potential improvements to the description. With the aim of building a classifier whose behaviour is more intuitive, a feature-less approach to classification based on the direct matching of known shapes is proposed. As can be seen in Fig. 4, the result of a match allows a clear interpretation of cases of misclassification. This aspect is needed for robust 3D classifying systems in the long run.

4.1 ICP for 3D Shape Alignment

The ICP (Iterative Closest Point) algorithm is a technique for performing geometric alignment of 3D models. Many variants of ICP have been proposed, varying aspects from point selection and matching to the minimisation strategy. As generalised by [16], most ICP implementations can be characterised by six aspects. Our design involves the following choices: (1) *selection*:

uniform random sampling at each iteration of ICP (based on speed considerations); (2) *matching*: L_2 norm; (3) *weighting*: none; (4) *rejection*: based on a measure of overlap between the two 3D surfaces; (5) *error metric*: the sum square distances between matched points (detailed in sec. 4.2); (6) *minimisation*: trust-region-reflective non-linear least square optimisation (based on Matlab’s lsqnonlin implementation).

In addition, as noted in [9], large numbers of 3D models are available on the Internet and can be used for the training and testing of 3D classifiers. Processing 3D models avoids the need for scanning real-world objects during the development phase of a classifier while allowing large-scale evaluations.

ICP optimisation based on distance means that while ICP is run on 3D point clouds, it corresponds to an optimisation which is 2D from a geometric point of view. Since the segmentation mechanism described in Sec. 2 provides an explicit representation of the ground, the position of the ground underneath each segmented object is known. As a consequence, the point clouds can be shifted so that they lie on a common ground surface. A 2D alignment then encodes contextual constraints. For example, if a pole is aligned with a car, a full 3D ICP may position the pole horizontally (this was observed during testing). On the other hand, performing a 2D alignment enforces that, during matching, the objects can only be shifted and rotated. The contextual information brought by the identification of the ground is essential to the classification process proposed here.

4.2 Template Based Classification

Once alignment of the two 3D shapes has been performed, a measure of similarity is computed for classification. The error metric used to evaluate the quality of the fit between the template and test shapes is as follows:

$$err = \sum_{i=1}^{N_{Test}} \|\mathbf{P}_i^{Test} - \mathbf{P}_{closest}^{Template}\| + \sum_{i=1}^{N_{Template}} \|\mathbf{P}_i^{Template} - \mathbf{P}_{closest}^{Test}\|, \quad (1)$$

where N_{Test} is the number of 3D points in the test point cloud $\{\mathbf{P}_i^{Test}\}$ used to compute the error. Here $\{\mathbf{P}_i^{Test}\}$ is a subset of the full test point cloud; it is obtained by voxelisation of the point cloud and by retaining only one return per occupied voxel (selected at random). Using a sub-sampled set accelerates the computations of the error metric. $\mathbf{P}_{closest}^{Template}$ is the 3D point in the template point cloud the closest to point \mathbf{P}_i^{Test} . The same notations are used in the second term of the equation with the difference that the superscripts *Template* and *Test* are exchanged. This second term makes the error symmetric, which is necessary if the template point cloud has a larger extent than the test case. In this case, the first term in Eq. 1 will be rather small since the whole test cloud is “covered” by the template set; however, for the same reason, the second term will be larger, allowing us to more accurately

capture the difference between the two shapes. Note that the error computed by the expression above is in metres which makes it easily interpretable.

For a given set of templates, the error metric defined above can be minimised by a certain template, but the corresponding minimum value may still be large. For instance, a test shape might belong to a class which is not represented in the set of templates. Thus, a rejection mechanism is needed. This is equivalent to the gating process in the context of data association. It is proposed to evaluate the amount of overlap between the two shapes compared (after alignment). The largest of the two sets is first identified (based on the sum of the eigenvalues of the point cloud). If, for instance, the largest point cloud is the test point cloud, the following operations are carried out for each of P_i^{Test} point. A plane is fitted to the points belonging to the same voxel as P_i^{Test} . Two additional planes normal to the first and orthogonal between themselves are built. If the (3D) quadrants defined by the two additional planes all contain at least one data point, then point P_i^{Test} is identified as belonging to the zone of overlap between the two surfaces. If more than $minOverlap$ of the $\{P_i^{Test}\}$ points are identified as belonging to the zone of overlap, the pair of surfaces is accepted for matching, otherwise, it is rejected. In our implementation $minOverlap$ is set to $1/2$.

Note that a hard threshold on the metric in Eq. 1 is not used for rejection since such a threshold would have to be adjusted depending on the objects being matched. For larger objects for instance, the overlap may be large but so may be the error (Eq. 1), simply due to the size of the objects. Applying a threshold on the error may reject the association, while the measure of overlap may accept it. The notion of overlap is a relative quantity that avoids the issues related to applying an absolute threshold, that must be valid for all sizes of objects.

4.3 Experiments

In this experiment, a subset of the 3D segments generated by the segmentation algorithm (on the Riegl scan) are used to evaluate the classification process (see Fig. 4(a)). The test is performed based on a leave-one-out procedure: a single segment is used as test data and the remaining segments are used as training data, the evaluation being repeated for each segment. The confusion matrix is presented in Table 4. The overall evaluation takes 3.9 minutes (3GHz Intel Duo Core, Matlab implementation). The evaluation is performed in such a way that the matching between any two objects is computed only once and the value is stored for when the same evaluation is required again. ICP is run for 10 iterations before evaluating the error in Eq. 1. Also the point clouds are subsampled before applying ICP to accelerate the computations.

With respect to the classes “car”, “pole” and “tree”, the classifier displays a high accuracy. This is indicated by the off diagonal terms in the confusion

matrix, which in the case of the class car are all zero (correct car classification is illustrated in Fig. 4(b)). One of the trees is confused with a pole which is due to the lack of samples on the canopy of the tree, implying that this tree looks like a pole; this is illustrated in Fig. 4(c). With respect to the class “wall”, the accuracy is lower, due to confusion with the class “tree”. A case of wrong classification of the class wall with the class tree is illustrated in Fig. 4(d). This comes from the small number of segments in the class “wall” (7 segments) as well as the large variety of shapes (due to their extent and due to occlusions during scanning generating holes in the 3D surfaces, as can be seen in Fig. 4(a)), implying that a correct match may not be found. Fig. 4(e) illustrates a correct matching between two trees.

Overall, the accuracy is 92.1% which is similar to the best accuracy obtained with a feature based approach (see Table 3): the Spin Image based classifier. However the proposed approach is much faster to compute: the whole featureless classification process runs in 3.9 minutes while the computation of a single Spin Image takes on average 5.5 minutes (Matlab implementations).

Table 4 Confusion Matrix. Accuracy=92.1%.

Truth \ Inferred	Car	Pole	Tree	Wall
Car	8	0	0	0
Pole	0	4	0	0
Tree	0	1	18	0
Wall	1	0	1	5

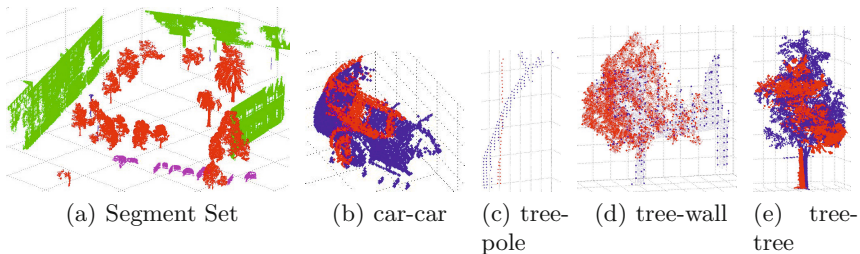


Fig. 4 (a) The subset of 3D segments used to test the 3D classifiers. The (8) segments belonging to the class “car” are indicated in cyan, the (4) segments in the class “pole” in blue, in red are the (19) segments of the class “tree” and in green the (7) segments of the class “wall”. (b)(c)(d)(e) Examples of best alignments computed during the ICP-based classification of the Riegl data. (b) The car in blue is correctly matched to the car in red. (c) Due to the lack of samples on the canopy of the tree in blue, the latter is identified with the pole in red. (d) The wall in blue is incorrectly matched to the tree in red. As developed in the text, this is due to the lack of 3D segments in the class “wall” implying that the best fit is found with the tree in red. In this plot the larger markers indicate the best fit with the set $\{\mathbf{P}_i^{Test}\}$ (see Sec. 4.2). (e) A correct match between two trees.

While the size of the current dataset is limited, these preliminary results demonstrate the potential of a feature-less approach to classification in terms of its interpretation and performance.

5 Conclusion

The paper introduces two methods as part of a robust 3D point cloud processing pipeline. The first is a segmentation method that jointly determines the ground surface and individual objects. The second is a feature-less classification that directly compares subsequent objects to templates via the ICP algorithm.

These methods are tested against field datasets and shown to be fast and accurate. For example, a 1.7 million point Riegl scan is segmented in 30 seconds and has four times less error than if processed by the next alternative approach. Similarly, 50 Velodyne scans are modelled with 7% less occupied voxels because the segmentation focuses the alignment on key sections. The feature-less classification achieves accuracy (92.1%) on par with the best feature based classifier (Spin Image KNN) while being 30% faster. Future work is considering the use of the ICP based classifier to explicitly remove cars, pedestrians and other dynamic objects in 3D scans before performing ICP registration. This should make the resulting alignment more accurate and useful.

In the end, an integrated processing approach based on robust segmentation is advocated as a means of going from 3D data to semantically rich classifications on which informed decisions can be made.

Acknowledgements. This work was supported by the Centre for Intelligent Mobile Systems (CIMS), funded by BAE Systems as part of an ongoing partnership with the University of Sydney, the Rio Tinto Centre for Mine Automation and the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government.

References

1. Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., Ng, A.: Discriminative learning of Markov random fields for segmentation of 3D scan data. In: Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR (2005)
2. Bosse, M., Zlot, R.: Place recognition using regional point descriptors for 3d mapping. In: Proc. of the International Conference on Field and Service Robotics, FSR (2009)
3. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Cambridge (2005)

4. MIT Urban Challenge datasets,
<http://grandchallenge.mit.edu/wiki/index.php/PublicData>
5. Johnson, A.: Spin-Images: A Representation for 3-D Surface Matching. PhD thesis, Carnegie Mellon University (1997)
6. Johnson, A., Hebert, M.: Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 25(1), 433–449 (1999)
7. Kazhdan, M., Funkhouser, T., Rusinkiewicz, S.: Rotation invariant spherical harmonic representation of 3d shape descriptors. In: *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2003)
8. Kelly, A., Stentz, A., Amidi, O., Bode, M., Bradley, D., Diaz-Calderon, A., Hap-pold, M., Herman, H., Pilarski, T., Rander, P., Thayer, S., Vallidis, N., Warner, R.: Toward reliable off road autonomous vehicles operating in challenging environments. *International Journal of Robotics Research (IJRR)* 25(5-6), 449–483 (2006)
9. Lai, K., Fox, D.: 3D laser scan classification using web data and domain adaptation. In: *Proceedings of Robotics: Science and Systems, Seattle, USA* (June 2009)
10. Lalonde, J., Vandapel, N., Huber, D., Hebert, M.: Natural terrain classification using three-dimensional lidar data for ground robot mobility. *Journal of Field Robotics* 23(10), 839–861 (2006)
11. Malisiewicz, T., Efros, A.: Improving spatial support for objects via multiple segmentations. In: *British Machine Vision Conference*, pp. 282–289 (2007)
12. Melkumyan, N.: Surface-based Synthesis of 3D Maps for Outdoor Unstructured Environments. PhD thesis, University of Sydney, Australian Centre for Field Robotics (2008)
13. Moosmann, F., Pink, O., Stiller, C.: Segmentation of 3D Lidar Data in non-flat Urban Environments using a Local Convexity Criterion. In: *Intl. Conf. Information Visualisation* (2009)
14. Munoz, D., Vandapel, N., Hebert, M.: Onboard contextual classification of 3-d point clouds with learned high-order markov random fields. In: *Proc. of the IEEE International Conference on Robotics & Automation, ICRA* (2009)
15. Pfaff, P., Burgard, W.: An efficient extension to elevation maps for outdoor terrain mapping and loop closing. *International Journal of Robotics Research (IJRR)* 26(2), 217–230 (2007)
16. Rusinkiewicz, S., Levoy, M.: Efficient variants of the ICP algorithm. In: *Proc. 3DIM*, pp. 145–152 (2001)
17. Siciliano, B., Khatib, O.: *Springer handbook of robotics*. Springer (2008)
18. Simmons, R., Henriksen, L., Chrisman, L., Whelan, G.: Obstacle avoidance and safeguarding for a lunar rover. In: *AIAA Forum on Advanced Developments in Space Robotics* (1996)
19. Thrun, S., et al.: Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics* 23(9), 661–692 (2006)
20. Triebel, R., Kersting, K., Burgard, W.: Robust 3D scan point classification using associative Markov networks. In: *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, pp. 2603–2608 (2006)