Scene Representations for Autonomous Driving: An Approach Based on Polygonal Primitives

Miguel Oliveira, Vítor Santos, Angel D. Sappa and Paulo Dias

Abstract In this paper, we present a novel methodology to compute a 3D scene representation. The algorithm uses macro scale polygonal primitives to model the scene. This means that the representation of the scene is given as a list of large scale polygons that describe the geometric structure of the environment. Results show that the approach is capable of producing accurate descriptions of the scene. In addition, the algorithm is very efficient when compared to other techniques.

Keywords Scene reconstruction · Point cloud · Autonomous vehicles

1 Introduction

Recent research in the fields of pattern recognition suggest that the usage of 3D sensors improves the effectiveness of perception, "since it supports good situation aware-

M. Oliveira(✉)
Instituto de Engenharia de Sistemas e Computadores,
Tecnologia e Ciência, R. Dr. Roberto Frias, 465, 4200 Porto, Portugal
e-mail: miguel.r.oliveira@inesctec.pt

V. Santos · P. Dias
Institute of Electronics and Telematics Engineering of Aveiro,
Campus Universitario de Santiago, 3800 Aveiro, Portugal

V. Santos
Department of Mechanical Engineering, University of Aveiro,
Campus Universitario de Santiago, 3800 Aveiro, Portugal

A.D. Sappa
Computer Vision Center, Edificio O, Campus UAB, Bellaterra, 08193 Barcelona, Spain

A.D. Sappa
Facultad de Ingeniería Eléctrica y Computación (FIEC), Escuela Superior Politécnica del Litoral (ESPOL), Campus Gustavo Galindo, Km 30.5 Vía Perimetral, 09-01-5863, Guayaquil, Ecuador

L.P. Reis et al. (eds.), *Robot 2015: Second Iberian Robotics Conference*,
Advances in Intelligent Systems and Computing 417,
DOI: 10.1007/978-3-319-27146-0_39

ness for motion level tele operation as well as higher level intelligent autonomous functions" [3]. Nowadays autonomous robotic systems have at their disposal a new generation of 3D sensors, which provide 3D data of unprecedented quality [16]. In robotic systems, 3D data is used to compute some form of internal representation of the environment. In this paper, we refer to this as 3D scene representation or simply 3D representation. The improvement of 3D data available to robotic systems should pave the road for more comprehensive 3D representations. In turn, advanced 3D representations of the scenes are expected to play a major role in future robotic applications since they support a wide variety of tasks, including navigation, localization, and perception [4].

In summary, the improvement in the quality of 3D data clearly opens the possibility of building more complex scene representations. In turn, more advanced scene representations will surely have a positive impact on the overall performance of robotic systems. Despite this, complex scene representations have not yet been substantiated into robotic applications. The problem is how to process the large amounts of 3D data. In this context, classical computer graphics algorithms are not optimized to operate in real time, which is an non-negotiable requirement of the majority of robotic applications. Unless novel, efficient methodologies are introduced, which produce compact yet elaborate scene representations, robotic systems are limited to mapping the scenes in classical 2D or 2.5D representations or are restricted to off-line applications.

Very frequently, the scenarios where autonomous systems operate are urban locations or buildings. Such scenes are often characterized for having a large number of well defined geometric structures. In outdoor scenarios, these geometric structures could be road surfaces or buildings, while in indoor scenarios they may be furniture, walls, stairs, etc. We refer to the scale of these structures as a macro scale, meaning that 3D sensor may collect thousands of measurements of those structures in a single scan. A scene representations is defined by the surface primitive that is employed. For example, triangulation approaches make use of triangle primitives, while other approaches such as Poisson resort to implicit surfaces. Triangulation approaches generate surface primitives that are considered to have a micro scale, since a geometric structure of the scene could contain hundreds or thousands of triangles. Micro scale primitives are inadequate to model large scale environments because they are not compact enough.

In this paper, we present a novel methodology to compute a 3D scene representation. The algorithm uses macro scale polygonal primitives to model the scene. This means that the representation of the scene is given as a list of large scale polygons that describe the geometric structure of the environment. The proposed representation addresses the problems that were raised in previous lines: the representation is compact and can be computed much faster than most others, while at the same time providing a sufficiently accurate geometric representation of the scene from the point of view of the tasks required by an autonomous system.

Scene reconstruction is defined as the computation of a geometric 3D model from multiple measurements. These measurements could be obtained from stereo systems, range sensors, etc. It could also include the texture mapping of the generated model.

Scene reconstruction methodologies are grouped into two different approaches: surface based representations or volumetric occupancy representations. In the first, the underlying surfaces of the scene that generated the range measurements are estimated, while in the second, the range measurements are grouped into grid cells which are then as labelled free or occupied. Traditional surface based representations include several 3D triangulations methodologies, such as 3D Delaunay triangulation [10], or Ball Pivoting Algorithm (BPA) [2]. The Greedy triangulation (GT) is an approach designed for fast surface reconstruction from large noisy data sets [12]. Given an unorganized 3D point cloud, the algorithm recreates the underlying surface's geometrical properties using data re-sampling and a robust triangulation algorithm, the authors claim to achieve near real time. There are also some alternative higher order surface representation methods such as Poisson surface reconstruction [11], Orientation Inference Framework [6] or learning approaches [13]. However, most of these methods do not tackle well noisy range measurements and, above all, since these methods involve a large number of nearest neighbor queries, they are very slow to compute. One attempt to accelerate the triangulation of point clouds was done in [12], but authors report they have only achieved near real time. Volumetric occupancy representations include occupancy grids [17], elevation maps [14], multi-level surface maps [15] or octrees [18]. While these representations are easier to compute, they do not provide accurate information about the geometry of the scene. The remainder of this paper is organized as follows: section 2 presents the proposed approach, results are given in section 3 and conclusions in section 4.

2 Proposed Approach

This work proposes to explore the usage of geometric polygonal primitives to perform scene reconstruction. In other words, the idea is to describe a scene by a list of polygons. The detection of polygonal geometric primitives is simple when compared to the detection of other more complex primitives. Furthermore, given that road environments are often geometrically structured, it seems feasible to represent the 3D structure with a set of planar polygons. In addition to that, polygons are compact representations, which require only the support plane and a list of points to be described.

Geometric polygonal primitives are described by a support plane and a bounding polygon. Let $\mathcal{G}^i$ represent the *ith* polygonal geometric primitive of a given scene, with the support plane Hessian form coefficients denoted by $\mathcal{G}^i_p = \left[a^i \; b^i \; c^i \; d^i \right]$. The search for the support plane is done on a given input point cloud $\mathcal{P}$ using a RANSAC procedure [7]. RANSAC is an iterative method to estimate parameters of a mathematical model from a set of observed data points. The assumption is that data consists of *inliers*, i.e., data whose distribution can be explained by some set of model parameters, and *outliers*, data that does not fit the model. The input to the RANSAC algorithm is a set of observed data values, a parametrized model which is

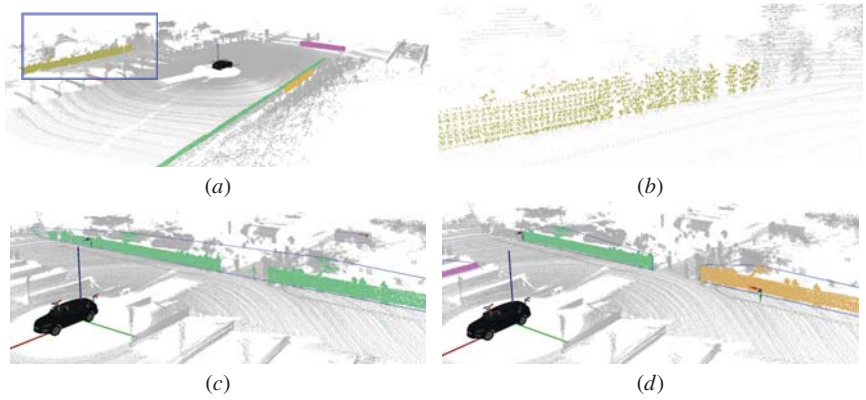


Fig. 1 Plane detection examples using RANSAC: (*a*) five best RANSAC candidates for the input point cloud in grey; (*b*) a detail of (*a*); (*c*) without using clustering; (*d*) using clustering;

fitted to the observations, and the output are the model parameters, i.e., in the case of detecting the support plane of polygonal primitives the Hessian coefficients.

Figure 1 (*a*) shows in different colors the inlier points of the five best candidates of a RANSAC search. Wall like structures are correctly detected. Figure 1 (*c*) shows the inliers (signalled in green) of a RANSAC plane detection. In this case, range measurements from two separate walls have been signalled as inliers of a single support plane. To address this issue, the set of inliers of each support plane hypothesis is used as input to a clustering procedure. Using the proposed clustering algorithm it is possible to separate the two walls into separate polygons as shown in Fig. 1 (*d*).

The computation of the bounding polygon is done after the detection of the support plane. The bounding polygon $\mathtt{P}^i$ is defined by a list of 2D points p:

$$\mathtt{P}^i = [p_0, p_1 ... p_n] = \begin{bmatrix} x_0 & x_1 & \dots & x_n \\ y_0 & y_1 & \dots & y_n \end{bmatrix} \tag{1}$$

where n is the number of points in the polygon. In order to define the polygon points in $\mathbb{R}^2$ (which saves memory), a local reference system for each polygon primitive is defined, with Z axis normal to the support plane, and the orientation of the remaining axes defined arbitrarily. Polygons are computed using a 2D convex hull operation. In this work the implementation provided in [8] is used to compute the 2D convex hull, based on a non recursive version of [5], presented in [1].

The proposed detection of polygonal primitives is designed in a cascade like processing architecture, which is efficient and fast to process. The input point cloud contains a large amount of 3D points. We assume 3D points can only belong to a single polygonal primitive, which makes sense since polygonal primitive represent objects

Algorithm 1. Cascade configuration for the detection of geometric polygonal primitives

Require: $\mathcal{P}^{it=0}$, the input point cloud at iteration 0
Ensure: A list of geometric polygonal primitives $\mathbf{G} = \{\mathcal{G}^0, \mathcal{G}^1, ..., \mathcal{G}^n\}$
  Initialize number of primitives, $k \leftarrow 0$
  Initialize number of iterations, $it \leftarrow 0$
  Initialize primitives list, $\mathbf{G} \leftarrow \{\}$
  Initialize cycle break flag, $cycle\_break \leftarrow$ `false`
  while $cycle\_break =$ `false` **do**
    RANSAC search over $\mathcal{P}^k$, returns estimated plane $\hat{\mathcal{G}}_p^k$ (first guess) and inliers $\mathcal{I}^k$
    if RANSAC found a candidate **then**
      Cluster inliers point cloud $\mathcal{I}^k$ to cluster list $\mathbf{C}$=$\{C^0, C^1, ..., C^n\}$
      Find largest cluster, $max\_cluster = argmax_i(\mathbf{size}(C^i))$
      Set the primitive support points $\mathcal{S}^k$ to the largest cluster, $\mathcal{S}^k = C^{max\_cluster}$
      Compute accurate plane coefficients from support points, $\mathcal{G}_p^k \leftarrow$ PCA over $\mathcal{S}^k$
      Compute bounding polygon $\mathtt{P}^k$, its area $\mathbf{A}(\mathtt{P}^k)$ and solidity $\mathbf{S}(\mathtt{P}^k)$
      if $\mathbf{A}(\mathtt{P}^k) > \mathbf{A}_t$ and $\mathbf{S}(\mathtt{P}^k) > \mathbf{S}_t$ **then**
        Add to primitive list, $\mathbf{G} \leftarrow \{\mathbf{G}, \mathcal{G}^k\}$
        increment number of primitives, $k \leftarrow k + 1$
      end if
      Remove support points $\mathcal{S}^k$ from $\mathcal{P}^{it}$, compute $\mathcal{P}^{it+1}$
    else
      Finish search for primitives, $cycle\_break =$ `true`
    end if
    increment number of iterations, $it \leftarrow it + 1$
  end while

Fig. 2 Sequence collected from the Massachusetts Institute of Technology (MIT) dataset.

in the scene. Let $\mathcal{S}^k$ be the point cloud containing the support points of primitive k, and $\mathcal{P}^k$ be the input point cloud in which the primitive was searched. The input point cloud for the search of the next primitive, $\mathcal{P}^{k+1}$, is obtained by removing the support points of primitive k:

$$\mathcal{P}^{k+1} = \left\{ \mathrm{p} \in \mathcal{P}^k \mid \mathrm{p} \notin \mathcal{I}^k \right\}. \tag{2}$$

Since every iteration of primitive detection will conduct a search on a smaller point cloud, it is expected that the cascade configuration is capable of reducing the processing time. Algorithm 1 details the complete procedure for the detection of a set of polygonal primitives given a point cloud.

3 Results

In order to evaluate the proposed 3D processing techniques a complete dataset both with 3D laser data, cameras and accurate egomotion is required. The MIT autonomous vehicle *Talos* competed in the Darpa Urban Challenge and achieved fourth overall place. The data logged by the robot is publicly available [9]. In total, the MIT logs sum up to 315GB of data. We have collected a small sequence of 40 seconds (200 meters of vehicle movement) at the start of the race (see Fig. 2). The sequence contains a continuous stream of sensor data, but in addition we have marked five locations (*A* through *E*) which are used to facilitate the analysis of the results. Additional information on each location is given in Table 1. Figure 3 shows images from all cameras, isometric and top views of the 3D data, and a satellite photograph of location *C*. The proposed approach is evaluated by analysing how the scenario contained in the sequence is reconstructed.

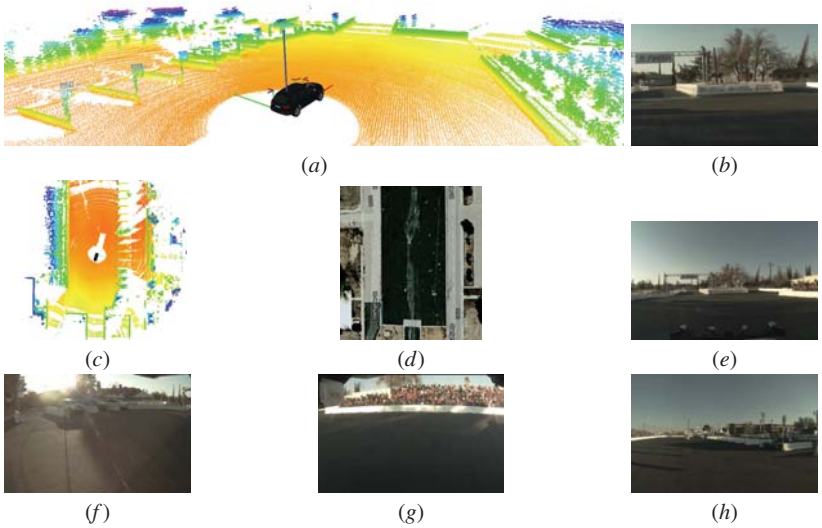
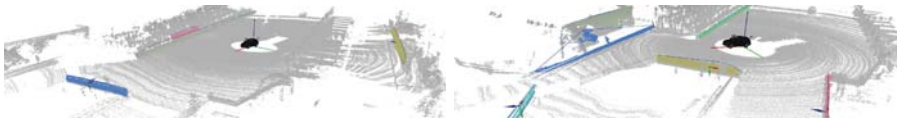


Fig. 3 Location *C* of the sequence: (*a*) isometric view; (*c*) top view; (*b*) front 6mm camera; (*e*) front (*h*) rear; (*f*) left (*g*) right; (*d*) satellite view of the location.



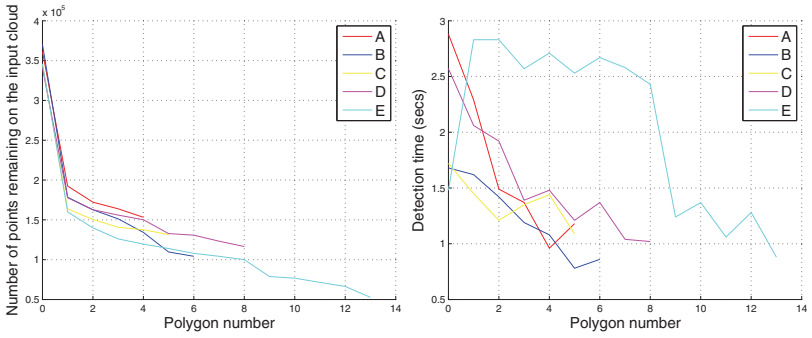
(*a*) (*b*)

Fig. 4 Detection of geometric polygonal primitives in the data sets of sequence 1: (*a*) location *C*; (*b*) location *D*.

Table 1 Information on each of the locations in this sequence. Columns description: pt, number of points; size, memory size in mega bytes; t, mission time in seconds; d, traveled distance in meters.

Location	Location Snapshot		Sequence accumulated			
Name	pt ($\times 10^6$)	size (MB) [(1)]	pt ($\times 10^6$)	size (MB) [(1)]	t (s)	d (m)
A	1.3	15.6	1.3	15.6	1	0
B	1.3	15.6	13.0	156.0	11	75
C	1.3	15.6	26.0	312.0	21	125
D	1.3	15.6	39.0	468.0	31	140
E	1.3	15.6	52.0	624.0	41	190

[(1)] Computed from the number of points times the three xyz dimensions times the four bytes for each dimension (type *float32*). It is an approximate value since there are other informations on the message, such as the time stamp, the coordinate frame identification, etc.



(*a*) (*b*)

Fig. 5 Cascade processing analysis: (*a*) the number of points left to process for a given input point cloud, as a function of the index of the detected primitive; (*b*) the time it takes to perform the detection of each of the geometric polygonal primitives as a function of the primitives index.

Figure 4 shows the polygonal primitives detected at locations *C* and *D*. It is possible to observe that the majority of the relevant planes are picked up by the algorithm.

3.1 Computational Efficiency

The detection of polygonal primitives is operated in a cascade-like configuration. In other words, the algorithm will search for polygonal primitives on a given input point cloud. After the first primitive is found, all the range measurements that are explained by that primitive are removed from the input point cloud. The second primitive is then searched in a smaller point cloud and so on. Since the search for a primitive is done over a decreasing size point cloud, it is expected that the search becomes faster with the number of detected primitives. In Fig. 5 an analysis of the computation time of each primitive is displayed. Primitives with higher numbers are detected in posterior phases.

Figure 5 (*a*) shows the number of points remaining in the input point cloud as a function of the primitive number. Results are shown for all locations in sequence 1. The number of detected primitives varies from location to location. It is also possible to observe that, as expected, the number of remaining points decreases with the increase in the number of detected primitives. Also, the reduction in the number of points is higher for early detected primitives. Hence, since the algorithm tends to remove the largest portion of points at the early stages of the cascade processing, this means that the latter stages will also be more efficient to compute. The reason for this behaviour is the RANSAC algorithm. Because RANSAC will search for the larger consensus, it will most likely select planes that are supported by a greater number of points. In this way, RANSAC tends to select first polygons with the largest amount of primitive support points. As a consequence, the largest decreases in the input point cloud occur early in the cascade, which in turn fastens subsequent detection stages of the cascade. The detection time per primitive is shown in Fig. 5 (*b*). The detection time tends to decrease with the increase in polygon number, for the reasons that where previously reported.

3.2 Comparison With Other Approaches

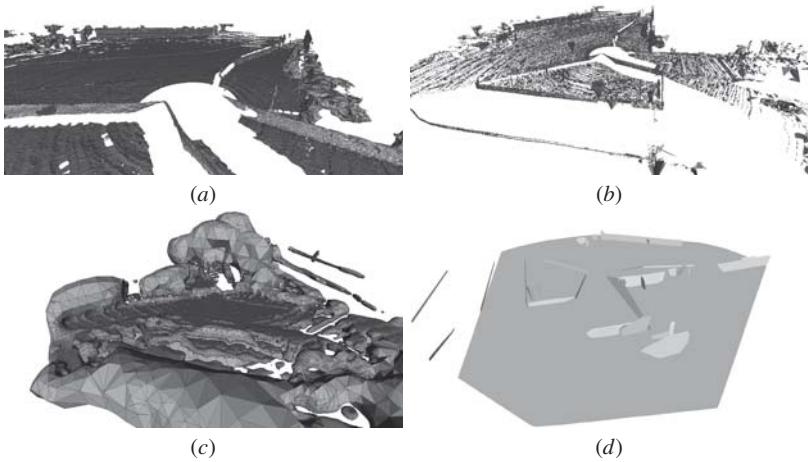
In this section we will compare the proposed approach with three surface reconstruction methodologies: Ball Pivoting Algorithm (BPA) [2], Greedy triangulation (GT) [12] and Poisson Surface Reconstruction (POIS) [11]. Two different parameter configurations the proposed approach are used. In the first Geometric Polygonal Primitives (GPP)1, parameters are set so that only very large polygons are detected. Processing time is faster, since a lot of polygons are discarded but, on the other hand, accuracy or completeness of the scene representation should be degraded. The second alternative, GPP 2, is configured so that even small polygons are detected, which should provides a more accurate scene description at the cost of a higher computation time.

Figure 6 (*a*) shows that the BPA method Figure 6 (*d*) shows results from the GPP. Since our approach uses primitives to define macro size structures, the number of

polygons used to represent the scene is small. Even though, it can be said that the most relevant polygons are part of the representation

Table 2 shows the computation times each algorithm took to reconstruct each of the locations in the sequence. The GPP methodology is the fastest one. This efficiency is related to the simplicity of the computed representation, and to the fact that RANSAC analyses only a small sample of points in the input point cloud, which means that not all input points are visited in order to reconstruct the scene, as is the case with the slower triangulation approaches.

To measure the accuracy of each reconstruction approach, the results obtained by BPA (the most accurate algorithm) are used as reference. Let X and Y be two meshes. The Hausdorff distance between those meshes $d_{\mathrm{H}}(X, Y)$ is computed as:



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

Fig. 6 Reconstruction of location *E* of MIT sequence: (*a*) BPA; (*b*) GT; (*c*) POIS; (*d*) GPP2.

Table 2 Comparison of the computation time of several approaches for surface reconstruction on the MIT data sets.

Sequence/ Location	Processing time (secs)				
	BPA	GT	POIS	GPP 1	GPP 2
A	659.0	154.0	63.2	16.3	27.3
B	752.9	157.5	61.6	25.3	17.4
C	488.2	156.3	56.3	13.5	49.4
D	480.4	142.4	52.6	25.2	25.2
E	558.8	149.0	57.9	47.4	58.1
μ	585.9	151.8	58.3	25.5	35.5

$$d_{\mathrm{H}}(X, Y) = \max\{ \sup_{x \in X} \inf_{y \in Y} d(x, y),\ \sup_{y \in Y} \inf_{x \in X} d(x, y) \}, \tag{3}$$

where sup and inf are the supremum and infinus, respectively. Since the Hausdorff distance is computed over a set of points (sampled using a Montecarlo strategy). In this particular case a variation of the Hausdorff distance, called the one sided Hausdorff distance is used where only the $\sup_{x \in X} \inf_{y \in Y} d(x, y)$ part is computed, because we only wish to measure how distant is each approach to the ground truth and not the other way around. In this case, the X meshes are given by each of the algorithms and the Y mesh is supplied by the ground truth mesh BPA. Table 3 shows the Hausdorff distance values obtained by GT, POIS, GPP 1 and GPP 2 using BPA meshes as ground truth.

The algorithm that obtains the best results is GT. The accuracy presented by the GPP 1 and GPP 2 approaches are about 0.8 and 0.65 meters, respectively. Figure 7 shows a graphical representation of the error for all of the approaches. For each approach, the output mesh has been sampled and the points are shown with color associated to the computed one sided Hausdorff distance of each point. A Red-Green-Blue colormap is used to code the distance. Red represents zero distance and blue maximum distance. In Fig. 7 (*a*), corresponding to the GT approach, almost all points have red color, resulting in low mean error. The POIS approach, represented in Fig. 7 (*b*), shows a lot of points in blue and green color, e.g., points whose minimum distance to the ground truth sampled points was very large. This is why POIS shows low accuracy values. In the case of the GPP approaches, 7 (*c*) and (*d*), some regions of the sampled points are more prone to have large error distances, while those in red seem to perfectly fit the ground truth mesh. The reason for this is that the BPA methodology, that was selected to serve as ground truth, does not perform interpolation over occluded areas, as the GPP approaches do. Most of the errors appear in the polygonal primitive that represents the ground plane, since this is the one that suffers more from occlusion from other planes. Errors may also result from the usage of convex hulls to compute the boundary polygons. We have investigated this by using alternatives to the proposed approach where the ground plane polygon is suppressed, and where concave hulls are used. Results are shown in Table 3. We

Table 3 Comparison of the accuracy of the several approaches using BPA results as ground truth and Hausdorff distance as metric.

Location	Hausdorff distance (meters)											
	GT			POIS			GPP 1			GPP 2		
	max	mean	RMS	max	mean	RMS	max	mean	RMS	max	mean	RMS
A	11.7	0.15	0.41	14.0	1.39	2.98	7.6	1.02	1.71	7.6	0.87	1.55
B	11.8	0.12	0.37	14.1	1.39	2.99	12.7	0.94	1.77	12.6	0.81	1.62
C	12.7	0.18	0.44	13.9	1.06	2.59	8.9	0.87	1.54	8.9	0.69	1.32
D	13.8	0.10	0.40	13.9	1.90	4.00	7.6	0.86	1.47	7.6	0.69	1.28
E	12.5	0.14	0.49	14.0	1.42	3.03	14.0	1.25	2.56	14.0	1.11	2.39
μ	12.5	0.14	0.42	13.9	1.43	3.12	10.2	0.99	1.81	10.1	0.83	1.63

can observe that, with the option of ground plane suppression and concave hull, the mean accuracy of GPP 2 increases to 0.1 meters, when compared to the previous 1.63.

Figure 8 shows a visual analysis of the Hausdorff distance errors for these variations of GPP 2. It is possible to observe that regions with error, e.g., in blue and green, decrease considerably when the concave hull is used, but in particular when the ground plane polygon is discarded.

Table 4 Comparison of the Hausdorff distance accuracy of the GPP 2 approach using: the standard approach, convex hull and ground plane included (also in Table 3); the convex hull with no ground plane included; the concave hull with ground plane; and the concave hull without ground plane.

	GPP 2 Hausdorff distance (meters)											
Hull	Convex			Convex			Concave			Concave		
Ground plane	Included			Not included			Included			Not included		
Location	max	mean	RMS	max	mean	RMS	max	mean	RMS	max	mean	RMS
A	7.6	0.87	1.55	1.8	0.15	0.26	6.8	0.71	1.25	1.2	0.13	0.19
B	12.6	0.81	1.62	1.5	0.11	0.19	12.6	0.53	1.09	1.1	0.08	0.14
C	8.9	0.69	1.32	1.9	0.16	0.29	6.6	0.52	0.99	1.9	0.12	0.22
D	7.6	0.69	1.28	2.2	0.14	0.26	7.3	0.59	1.13	2.1	0.11	0.21
E	14.0	1.11	2.39	1.7	0.10	0.19	8.8	0.32	0.81	1.4	0.08	0.14
μ	10.1	0.83	1.63	1.8	0.13	0.24	8.4	0.53	1.05	1.5	0.10	0.18

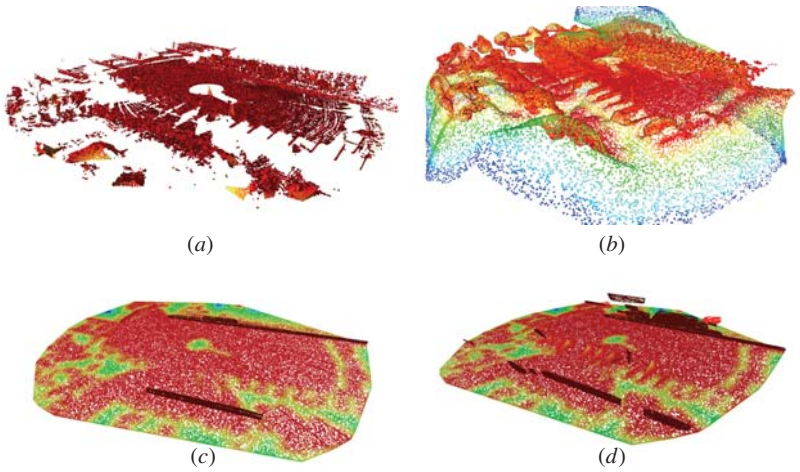


Fig. 7 Qualitative analysis of the one sided Hausdorff distance in location *C* sequence 1: (*a*) GT; (*b*) POIS; (*c*) GPP 1; (*d*) GPP 2; A Red-Green-Blue color map is used to code the distance. Red represents zero distance and blue maximum distance.

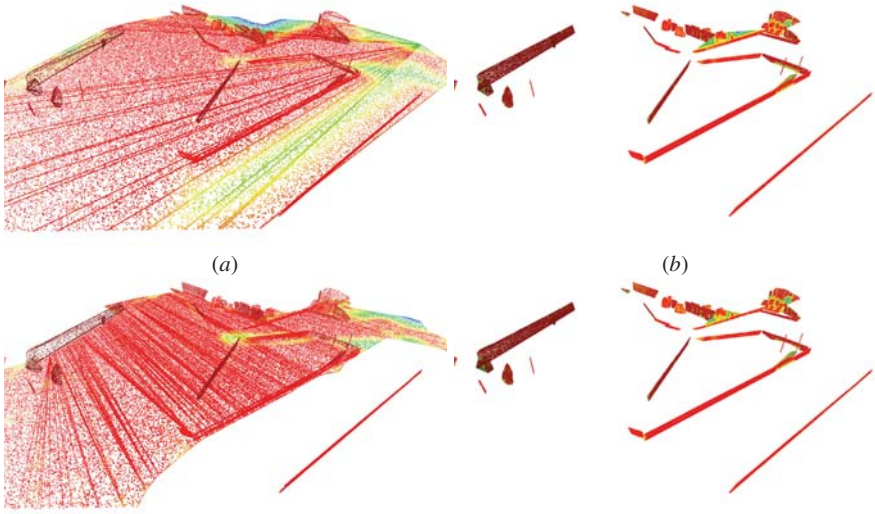


Fig. 8 Results from the Hausdorff distance obtained when using alternatives for the GPP 2 method for location *E*, sequence 1: (*a*) the standard GPP 2, with ground plane and convex hull; (*b*) discarded ground plane, convex hull; (*c*) with ground plane, concave hull; (*d*) discarded ground plane, concave hull. A Red-Green-Blue color map is used to code the distance. Red represents zero distance and blue maximum distance.

4 Conclusions

This paper proposes a novel approach to produce scene representations using the array of sensors on-board autonomous vehicles. Since roads are semi structured environments with a great deal of macro size geometric structures, we argue that the use of polygonal primitives is well suited to describe these scenes. Results have shown that the proposed approach is capable of producing accurate descriptions of the scene, and that it is considerably faster than all the approaches used in this evaluation. Future work will include the addition of texture on the polygons generated by the proposed algorithm. In this way, we expect to have the means to produce scene representations that can be used not only for standard task such as obstacle detection and motion planning, but also for more complex endeavours such as recognizing patterns in the scene.

Acknowledgments This work has been supported by the "Fundação para a Ciência e Tecnologia" under grant agreement SFRH/BD/43203/2008. A. Sappa has been partially supported by: the Spanish Government under Project TIN2014-56919-C3-2-R and the PROMETEO Project of the "Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación de la República del Ecuador.

References

1. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software **22**(4), 469–483 (1996)
2. Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G.: The ball-pivoting algorithm for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics **5**(4), 349–359 (1999)
3. Birk, A., Vaskevicius, N., Pathak, K., Schwertfeger, S., Poppinga, J., Buelow, H.: 3-d perception and modeling. IEEE Robotics Automation Magazine **16**(4), 53–60 (2009)
4. Burgard, W., Pfaff, P.: Editorial: Three-dimensional mapping, part 1. Journal of Field Robotics **26**(10), 757–758 (2009)
5. Bykat, A.: Convex hull of a finite set of points in two dimensions. Information Processing Letters **7**, 296–298 (1978)
6. Chen, Y.L., Lai, S.H.: An orientation inference framework for surface reconstruction from unorganized point clouds. IEEE Transactions on Image Processing **20**(3), 762–775 (2011)
7. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: ACM. Los Angeles, California, June 1981
8. Hert, S., Schirra, S.: 2D convex hulls and extreme points. In: CGAL User and Reference Manual. CGAL Editorial Board, 4th edn. (2012)
9. Huang, A.S., Antone, M., Olson, E., Fletcher, L., Moore, D., Teller, S., Leonard, J.: A High-rate, Heterogeneous Data Set from the DARPA Urban Challenge. International Journal of Robotics Research **29**(13), 1595–1601 (2011)
10. Jovanovic, R., Lorentz, R.: Compression of volumetric data using 3D delaunay triangulation. In: 2011 4th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO), pp. 1–5, April 2011
11. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the fourth Eurographics Symposium on Geometry Processing SGP 2006, pp. 61–70. Eurographics Association (2006)
12. Marton, Z.C., Rusu, R.B., Beetz, M.: On fast surface reconstruction methods for large and noisy datasets. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, May 2009
13. de Medeiros Brito, A., Doria Neto, A., Dantas de Melo, J., Garcia Goncalves, L.: An adaptive learning approach for 3-d surface reconstruction from point clouds. IEEE Transactions on Neural Networks **19**(6), 1130–1140 (2008)
14. Oniga, F., Nedevschi, S.: Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection. IEEE Transactions on Vehicular Technology **59**(3), 1172–1182 (2010)
15. Rivadeneyra, C., Miller, I., Schoenberg, J., Campbell, M.: Probabilistic estimation of multi-level terrain maps. In: IEEE International Conference on Robotics and Automation, ICRA 2009, pp. 1643–1648, May 2009
16. Rusu, R.B., Cousins, S.: 3D is here: point cloud library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 2011
17. Weiss, T., Schiele, B., Dietmayer, K.: Robust driving path detection in urban and highway scenarios using a laser scanner and online occupancy grids. In: 2007 IEEE Intelligent Vehicles Symposium, pp. 184–189, June 2007
18. Zhou, K., Gong, M., Huang, X., Guo, B.: Data-parallel octrees for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics **17**(5), 669–681 (2011)