Robotics and Autonomous Systems 83 (2016) 312–325

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Incremental scenario representations for autonomous driving using geometric polygonal primitives

Miguel Oliveira [a,b,*], Vitor Santos [b], Angel D. Sappa [c,d], Paulo Dias [b], A. Paulo Moreira [a,e]

[a] INESC TEC - INESC Technology and Science, R. Dr. Roberto Frias s/n, 4200-465 Porto, Portugal

[b] IEETA - Institute of Electronics and Informatics Engineering of Aveiro, Universidade de Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal

[c] Facultad de Ingeniería en Electricidad y Computación, Escuela Superior Politécnica del Litoral, ESPOL, Campus Gustavo Galindo, Km 30.5 vía Perimetral, P.O. Box 09-01-5863, Guayaquil, Ecuador

[d] Computer Vision Center, Campus UAB, 08193 Bellaterra, Barcelona, Spain

[e] FEUP - Faculty of Engineering, University of Porto, R. Dr. Roberto Frias s/n, 4200-465 Porto, Portugal

ARTICLE INFO

Article history:
Available online 7 June 2016

Keywords:
Incremental scene reconstruction
Point clouds
Autonomous vehicles
Polygonal primitives

ABSTRACT

When an autonomous vehicle is traveling through some scenario it receives a continuous stream of sensor data. This sensor data arrives in an asynchronous fashion and often contains overlapping or redundant information. Thus, it is not trivial how a representation of the environment observed by the vehicle can be created and updated over time. This paper presents a novel methodology to compute an incremental 3D representation of a scenario from 3D range measurements. We propose to use macro scale polygonal primitives to model the scenario. 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. Furthermore, we propose mechanisms designed to update the geometric polygonal primitives over time whenever fresh sensor data is collected. Results show that the approach is capable of producing accurate descriptions of the scene, and that it is computationally very efficient when compared to other reconstruction techniques.

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 awareness for motion level teleoperation as well as higher level intelligent autonomous functions" [1]. Nowadays, autonomous robotic systems have at their disposal a new generation of 3D sensors, which provide 3D data of unprecedented quality [2]. 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 [3].

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 a non-negotiable requirement of the majority of robotic applications. Unless novel and efficient methodologies that produce compact, yet elaborate scene representations, are introduced by the research community, 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 representation is defined by the surface primitive that

* Corresponding author at: INESC TEC - INESC Technology and Science, R. Dr. Roberto Frias s/n, 4200-465 Porto, Portugal.

E-mail addresses: m.riem.oliveira@gmail.com (M. Oliveira), vitor@ua.pt (V. Santos), asappa@cvc.uab.es (A.D. Sappa), paulo.dias@ua.pt (P. Dias), amoreira@fe.up.pt (A.P. Moreira).

http://dx.doi.org/10.1016/j.robot.2016.05.011
0921-8890/

Table 1
LIDAR sensor systems mounted on some autonomous vehicle systems of recent years.

Institution	Vehicle	Ref.	3D sensor type		Total[a]
			2D laser	3D laser	
Stanford U.	Stanley[b]	[11]	5 × Sick LMS 291	–	67.5
CMU	Sandstorm[b]	[12]	3 × Sick LMS 291	–	50.5
	Highlander[b]		Riegl Q140i	–	
CMU	Boss[b]	[13]	6 × Sick LMS 291	Vel. HDL-64	2305.0
			2 × Continental ISF 172	–	
			2 × IBEO Alasca XT	–	
Stanford U.	Junior[c]	[14]	4 × SICK LMS 291	Vel. HDL-64	2278.0
			2 × IBEO Alasca XT	–	
Virginia Tech	Odin[c]	[15]	4 × Sick LMS 291	–	90.0
			2 × IBEO Alasca XT	–	
			IBEO Alasca AO	–	
MIT	Talos[c]	[6]	12 × Sick LMS 291	Vel. HDL-64	2361.2
U. Munich	MuCar-3[d]	[16]	–	Vel. HDL-64	2200.0
Google	Driverless Car	[17]	–	Vel. HDL-64	2200.0

[a] Estimation of total 3D data throughput of all LIDAR sensors mounted on the vehicle, given as points $\times 10^3$/s.
[b] These vehicles participated in the *Defense Advanced Research Projects Agency* (DARPA) Grand Challenge 2006.
[c] These vehicles participated in the DARPA Urban Challenge 2007.
[d] This vehicle participated in the *Civilian European Land Robot Trial* (ELROB) Trial 2009.

is employed. For example, triangulation approaches make use of triangle primitives, while other approaches such as Poisson surface reconstruction 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 the 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.

The second problem addressed in this paper is the reconstruction of large scale scenarios from a continuous throughput of massive amounts of 3D data. We will use the distinction between the terms scene and scenario. Let scenario refer to a particular location that should be reconstructed, e.g., a city, a road or a building. By scene, we refer to the portion of the scenario that is viewed by the vehicle at a particular time. Thus, the scenario is an integration of scenes over time. In the case of large scale scenarios, the compactness of a given scene representation is even more important. In this paper, we focus also on how the representation may evolve by integrating 3D data from multiple measurements over time.

This is an extended version of [4]. The new material covers mostly the incremental part of the geometric reconstruction. There is also the possibility of adding texture to the geometric scene description. For further details on this see [5].

For testing and evaluation purposes, we use a data-set from the *Massachusetts Institute of Technology* (MIT) Team, taken from their participation in the DARPA Urban Challenge [6]. From this data-set we have extracted a 40 s sequence which will be used to assess the proposed algorithms. For the remainder of the paper, this sequence is referred to as MIT sequence. Using this data-set, we aim at reconstructing large portions of the urban environment in which the competition took place.

The remainder of this paper is organized as follows: Section 2 reviews the state of the art, Section 3 presents the proposed approach. Results are given in Section 4 and conclusions in Section 5.

2. Related work

At first glance, it would seem plain to translate the improvement on the quality of the 3D data into an enhancement of the 3D representations. However, the fact is that the majority of the robotic systems, namely autonomous vehicles, continue to rely on classic 2D or 2.5D scene representations [7], such as occupancy grids [8] or elevation maps [9], or use discretized grid-like approaches as in octrees [10]. The reason for that is that autonomous vehicles commonly require a large array of sensors installed onboard and, as a consequence, collect large amounts of range measurements every second. Table 1 shows an estimate of the amount of 3D data (measured by LIDAR systems alone) generated by several autonomous vehicles. Simplified 2D or 2.5D representations are used so that they can be computed in real time using large amounts of data. More advanced 3D representations have not been introduced in robotics because they fail to abide to the requirements of real time processing using the 3D data produced by new generation LIDAR sensors. One example of this is the methodologies used in the computer graphics research field: traditional algorithms such as building of triangular meshes are unable to operate in real time with the throughput of data provided by new generation 3D sensors. Some authors have tried to optimize triangulation algorithms (e.g., [2,7]), and they report near real time performances. Note that these results were obtained using point clouds from a Microsoft Kinect 3D camera.[1] On the other hand, the results provided towards the end of this paper are obtained using point clouds from a Velodyne HDL-64E Lidar,[2] and therefore results are not directly comparable.

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. Scene reconstruction may also include the texturing of the generated 3D 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 cells of a grid, and are then labeled free or occupied. Traditional surface based representations include several 3D triangulations methodologies, such as 3D Delaunay triangulation [18], or *Ball Pivoting*

[1] http://en.wikipedia.org/wiki/Kinect.
[2] http://velodynelidar.com/lidar/lidar.aspx.

Algorithm (BPA) [19]. There are also some alternative higher order surface representation methods such as Poisson surface reconstruction [20], Orientation Inference Framework [21] or learning approaches [22]. 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 [7], but authors report they have only achieved near real time. Volumetric occupancy representations include occupancy grids [8], elevation maps [9], multi-level surface maps [23] or octrees [10]. While these representations are easier to compute, they do not provide accurate information about the geometry of the scene.

The BPA triangulation was proposed in [24]. The BPA computes a triangle mesh interpolating a given point cloud. The principle of the BPA is very simple: Three points form a triangle if a ball of a user-specified radius touches them without containing any other point. Starting with a seed triangle, the ball pivots around an edge (i.e., it revolves around the edge while keeping in contact with the edge's endpoints) until it touches another point, forming another triangle. The process continues until all reachable edges have been tried, and then starts from another seed triangle, until all points have been considered. Although all range points are considered in the computation of the mesh, which accounts for the accuracy of the methodology, this fact also hampers the computational performance of the algorithm.

The *Greedy Triangulation* (GT) is an approach designed for fast surface reconstruction from large noisy data sets [7]. Given an unorganized 3D point cloud, the algorithm recreates the underlying surface's geometrical properties using data resampling and a robust triangulation algorithm, the authors claim to achieve near real time. For resulting smooth surfaces, the data is resampled with variable densities according to previously estimated surface curvatures. One of the advantages of this method is that, since a greedy search is executed, it is expected to be faster than other standard triangulation approaches.

Poisson surface reconstruction was initially proposed in [25]. In this approach, surface reconstruction of a point cloud with estimated normals is viewed as a spatial Poisson problem. The Poisson formulation considers all the points at once, without resorting to heuristic spatial partitioning or blending, and is therefore highly resilient to data noise. Unlike radial basis function schemes, a Poisson approach allows a hierarchy of locally supported basis functions, and therefore the solution reduces to a well conditioned sparse linear system.

The work from [26] proposes an expectation maximization based method for producing a scene representation based on planes, rather than polygons. Also, this method is not intended for real time applications, since creating a scene representation may take up to twenty minutes. Finally, this work does not discuss how new sensor measurements could be integrated in the existing representation, therefore not focusing on the incremental part of scenario reconstruction.

3. Proposed approach

In this section we will explain in detail the methodologies of our approach. First, we describe the scene reconstruction algorithm (see Section 3.1) and then, in Section 3.2, we describe how a scenario is created over time from a continuous throughput of 3D data.

3.1. One-shot scene reconstruction

This work proposes to explore the usage of *Geometric Polygonal Primitives* (GPP) to perform scene reconstruction. In other words, the idea is to describe a scene by a list of macro scale polygons. The detection of geometric polygonal 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: 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 coefficients denoted by $\mathcal{G}_p^i = \begin{bmatrix} a^i & b^i & c^i & d^i \end{bmatrix}$. The search for the support plane is done on a given input point cloud $\mathcal{P}$ using a *Random Sample Consensus* (RANSAC) procedure [27]. Note that there are other algorithms which are more efficient for detecting planes in 3D data [28]. However, an analysis of the impact of these alternative approaches is out of the scope of the current paper. 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 parameterized model which is 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.

Algorithm 1 Cascade 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 P^k, its area $\mathbf{A}(\mathrm{P}^k)$ and solidity $\mathbf{S}(\mathrm{P}^k)$
      if $\mathbf{A}(\mathrm{P}^k) > \mathbf{A}_t$ and $\mathbf{S}(\mathrm{P}^k) > \mathbf{S}_t$ **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. 1(a) shows in different colors the inlier points of the five best candidates of a RANSAC search. Wall like structures are correctly detected. Fig. 1(c) shows the inliers (signaled in green) of a RANSAC plane detection. In this case, range measurements from two separate walls have been signaled 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. By using the proposed clustering algorithm, it is possible to separate the two walls into separate polygons, as shown in Fig. 1(d). Polygons are computed using a 2D convex hull operation on the (clustered) RANSAC inliers. In this work, the implementation provided in [29] is used to compute the 2D convex hull, based on a non recursive version of [30,31].

To increase the efficiency of the algorithm, we propose to conduct the detection of polygonal primitives in a cascade like processing configuration. This should be more efficient and fast to process. The input point cloud contains a large amount of 3D points: we assume each 3D point can only belong to a single polygonal primitive. 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

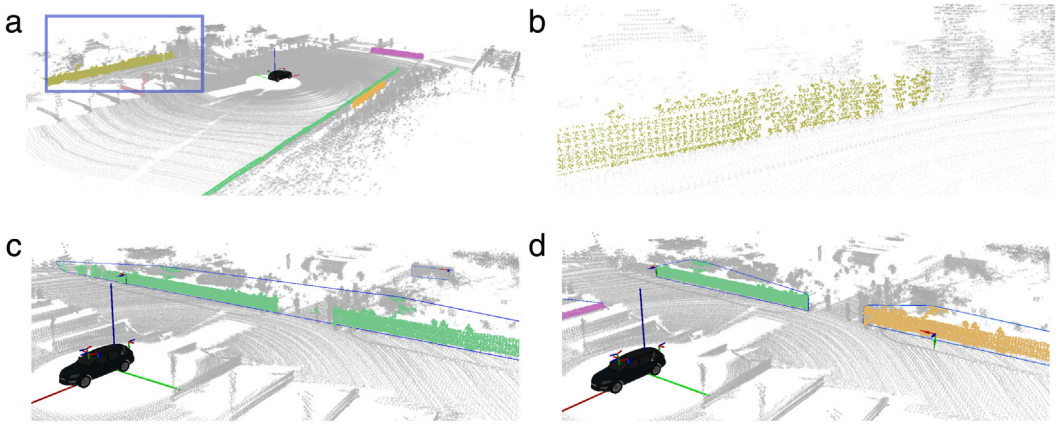

Fig. 1. Plane detection examples using RANSAC: (a) five best RANSAC candidates for the input point cloud in grey; (b) a detail of (a); (c) without using clustering; (d) using clustering. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

search of the next primitive, $\mathcal{P}^{k+1}$, is obtained by removing the support points of primitive k:

$$\mathcal{P}^{k+1} = \left\{ p \in \mathcal{P}^k \mid p \notin \mathcal{S}^k \right\}. \quad (1)$$

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.

3.2. Incremental scenario reconstruction

We use the term scenario reconstruction to designate an incremental process of reconstruction of scenes from a continuous stream of sensor data. At first sight, there are three alternatives for performing scenario refinement: (1) store raw measurements and, in the end, reconstruct using the entire data; (2) reconstruct periodically and fuse partial scene reconstructions; (3) reconstruct with the first input data then make the representation evolve as new data arrives.

Approach (1) is the most immediate, since there are well known surface reconstruction algorithms which reconstruct a surface from a single point cloud. This approach merely merges all input point clouds into a single point cloud, the accumulated point cloud, and then, standard surface reconstruction algorithms may be applied using the accumulated point cloud as input. One downside of this method is that the process of reconstruction can only begin after all point clouds have been collected. This is not suited for usage in real time applications. Furthermore, the amount of data that results from the accumulation of point clouds should be very large, which in turn might cause problems for reconstruction algorithms. Fig. 2 shows an example of a scenario reconstruction (with BPA) using as input the accumulated point clouds of three scenes. This reconstruction took over 2 h to complete.

Alternative (2) proposes a fusion of (partially) reconstructed scenes for each of the locations. Taking the example of Fig. 2: if the three scenes are reconstructed independently using BPA, the overall process takes $488.2 + 480.4 + 558.8 = 1487.4$ s (this is without considering the overhead attached to the process of fusion). Fig. 3 shows the reconstructed scenarios obtained using this alternative. The downsides of this alternative are the need to define a strategy to fuse the reconstructed scenes in order to obtain a global scenario representation. In addition to this, note that the reconstructed scenes overlap. In overlapping regions, reconstruction is carried out multiple times without originating an improved description, therefore waisting computational resources.

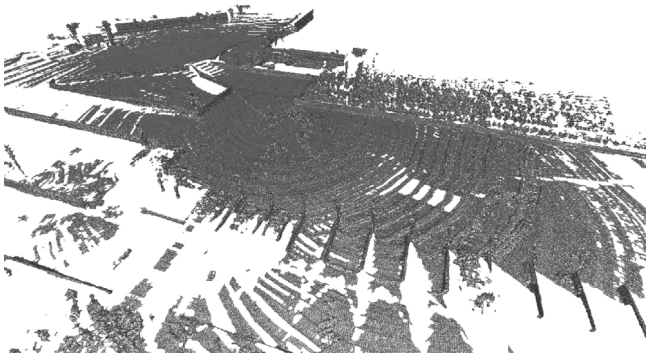

Fig. 2. Scenario reconstruction approach (1): BPA surface reconstruction over the accumulated point cloud of three scenes.

Alternative (3) proposes an incremental scenario reconstruction. Unlike in the other two approaches, in this case, the reconstruction of a given scenario receives as input not only the current sensor data (a point cloud), but also a description of the scenario as seen by preceding reconstructions. This method appears to be more interesting than the others, since a scenario representation is updated or refined when new sensor data arrives. An illustrative example: a vehicle is moving on a road and there is a long wall on the side of the road. At the beginning of the road, sensors see only a portion of the wall and the reconstructed surface corresponds only to the visible part of the wall. As the vehicle moves forward, additional areas of the wall become visible to the sensors. These additional range measurements of the wall should be used to update the already existing shape primitive that represents the wall, rather than to create a new shape primitive which represents the new visible portion of the wall, and that overlaps, to some extent, the first wall primitive.

In the following lines, we present mechanisms that enable a scene representation based on GPPs to be incrementally refined from novel point cloud data. To update the representation, an operation called expansion is executed for each of the existing GPPs. The expansion receives as input a list of points (the 3D point cloud) as well as the definition of the polygon that is to be expanded. It is composed of two parts, a perpendicular and a longitudinal expansion, and is defined as follows (see Fig. 4): Let $\mathcal{P}$ represent an input point cloud that is received at a given time and that contains several points (triangles and diamonds in Fig. 4(a)), and the scenario representation that was previously computed, being composed of a single primitive $\mathcal{G}$ (black solid line polygon in Fig. 4(a)). The primitive has a support plane, as well as a local coordinate system represented by red–green–blue lines. The first

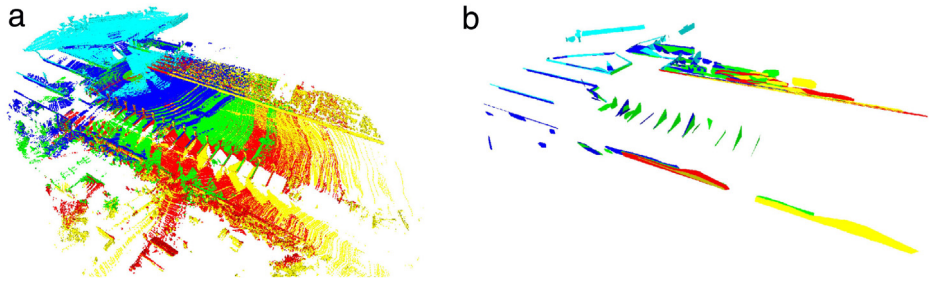

Fig. 3. Scenario reconstruction approach (2): Accumulation of scene reconstructions over multiple scenes, each marked with a different color; (a) BPA; (b) GPP 2, shown without the ground plane for an easier visualization. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

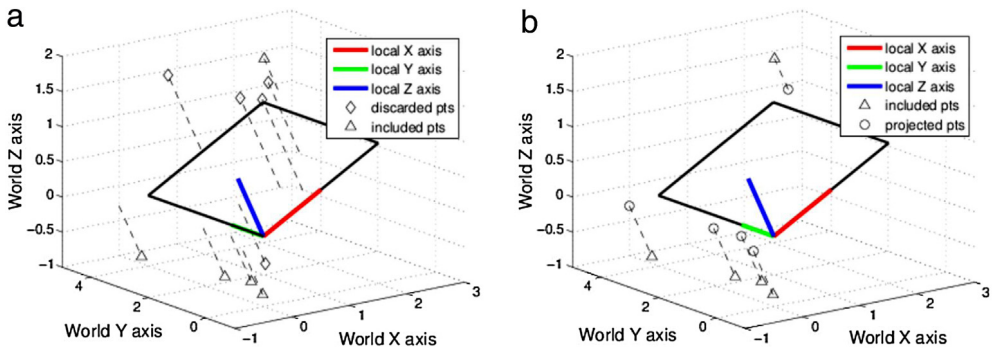

Fig. 4. Orthogonal component of the expansion operation: (a) points are tested for their orthogonal distance to the support plane; (b) included points are projected to the support plane. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

step is to compute a new point cloud $\mathcal{P}_{ort}$ that is given by the points of $\mathcal{P}$ whose distance to the plane is smaller than the perpendicular expansion threshold T_{ort}:

$$\mathcal{P}_{ort} = \{p_j \in \mathcal{P} \mid d_j < T_{ort}\}, \tag{2}$$

where d_j is the distance of point p_j to the support plane of $\mathcal{G}$. Only points that lie close to the primitives support plane are stored in $\mathcal{P}_{ort}$ and used in the next steps. In Fig. 4(a), some points are included in $\mathcal{P}_{ort}$ (triangles) and others discarded (diamonds). Then, the points in $\mathcal{P}_{ort}$ are projected into the primitives support plane and their coordinates transformed into the primitives local coordinate frame. In this local reference frame, the projected points always have z value equal to zero, which is why only the x and y coordinates are stored, i.e., points are defined in $\mathbb{R}^2$. Let $\mathcal{J}$ be the point cloud that contains the x and y coordinates of the projected points viewed from the primitive local coordinate system. Fig. 4(b) shows the projections of the triangles of Fig. 4(a) to the support plane (circles). This process is called the orthogonal part of the expansion. From here onward, all computations are performed in $\mathbb{R}^2$, which significantly speeds up the computation.

The second part of the expansion is referred to as longitudinal expansion. Fig. 5(a) shows an example point cloud $\mathcal{J}$ that contains several points. Let us consider that some of these points actually belong to the same object that the primitive represents (circles in Fig. 5(a)), and others do not (squares in Fig. 5(a)). Since these points are obtained from new data, not all of them are contained inside the bounding polygon of the corresponding primitives. Therefore, the primitive should expand to accommodate these new points. To do this, an iterative process is proposed. The first step is to offset the existing bounding polygon of the primitive. The algorithm used was introduced in [32] and the implementation is from [33]. The offsetting operation generates a new larger polygon, and in every iteration, the polygon grows as detailed next. The bounding polygon of the primitive is referred to as P, and the grown or extended polygon is referred to $\hat{P}$. Then, all points in $\mathcal{J}$ are tested to see if they are inside $\hat{P}$. The final stage is to compute a new convex hull. This new convex hull is computed from the point set that contains both the points of the previous convex hull and the points to which the polygon expanded to. The process is repeated using the newly computed convex hull as the starting hull. The iterative expansion stops when the extended polygon does not contain points inside it.

Fig. 5 shows an example of an iterative longitudinal expansion. Fig. 5(a) shows the initial situation; Fig. 5(b) shows the start of the iterative process. The expanded polygon (dashed line), is offset from the initial bounding polygon (solid blue line). Points are tested to see whether they are inside the offset polygon (annotated with crosses). The process repeats until, in the fifth iteration (Fig. 5(f)), no new points are found inside the offset polygon. This causes the iterative search to finish. The expansion operation changes the polygon from its initial state Fig. 5(a) to a new shape, solid magenta line in Fig. 5(f).

We propose to conduct the expansion of the current primitives using also a cascade configuration, based on the following reasoning. Each range measurement is obtained from a laser beam reflection of a single physical object. Thus, we can assume that each 3D point is explained by a GPP. Under this assumption, the points that have been assigned to a given primitive by an expansion operation, can only belong to that primitive and no other. Because of this, expanded points are removed from the input point cloud and are not a part of subsequent expansions (of other primitives) nor part of detections of new primitives. Since all the points that are taken by the expansion of a primitive are removed from the input point cloud, the subsequent expansions or searches are accelerated since that less points need to be analyzed. In terms of configuration, a cascade processing recommends that the faster stages are computed first. The expansion of primitives is faster than the detection. Because of this, when a new input point cloud is received, all existing primitives are first expanded and only then the remainder non expanded points are used for searching new primitives. Algorithm 2 describes the architecture of the complete algorithm for the geometric polygonal primitives representation computation, including both the detection and expansion operations.

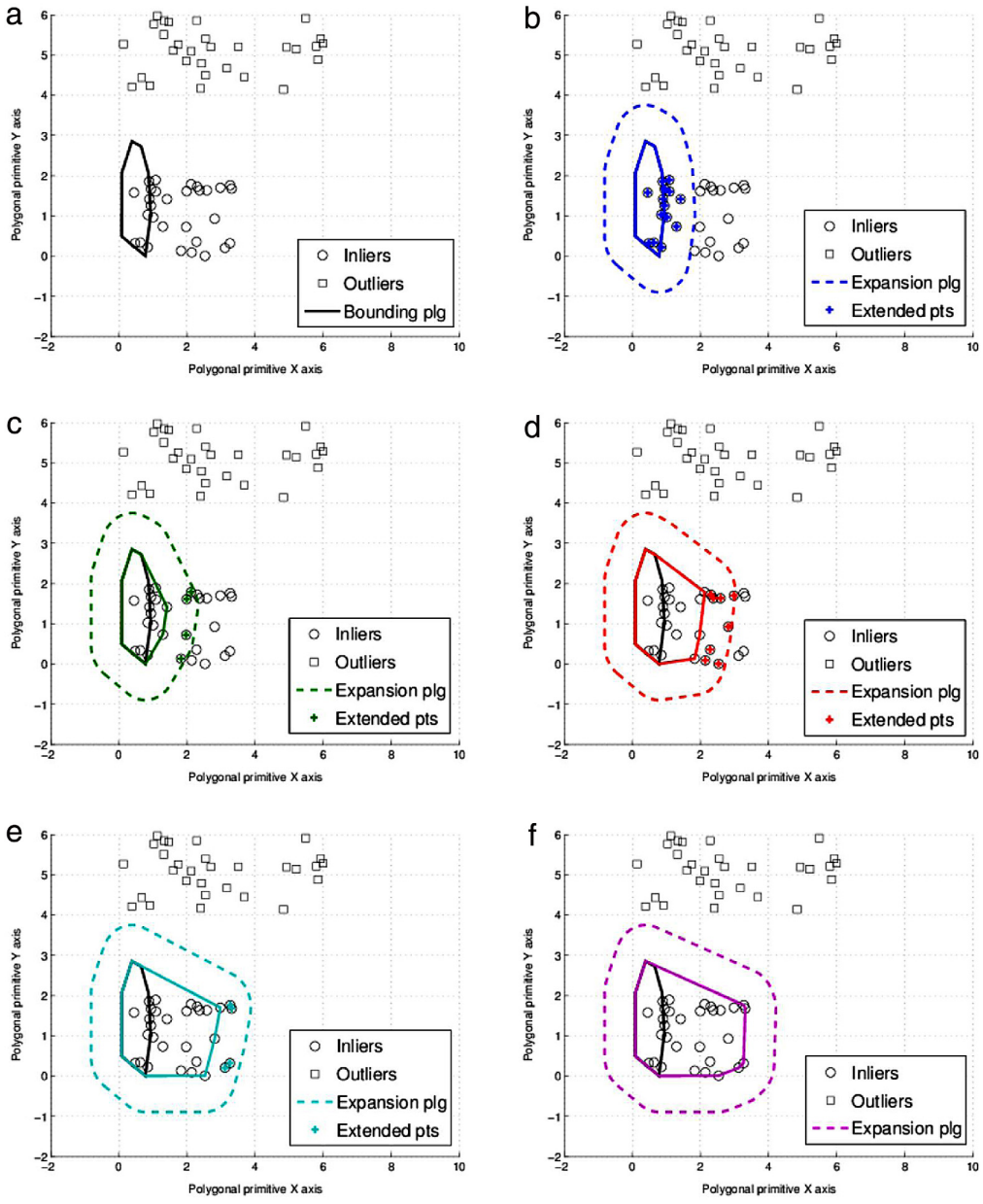

Fig. 5. Successive iterations of the longitudinal expansion of GPPs. Solid lines in color represent the convex hull at the start of a given iteration, dashed lines the expanded polygon. Crosses over points mean they were added to the polygon in a given expansion: (a) initial situation; (b), (c), (d), (e) and (f) iterations 0–4, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4. Results

In order to evaluate the proposed 3D processing techniques, a complete data-set with 3D laser data, cameras and accurate ego-motion 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 [6]. In total, the MIT logs sum up to 315 GB of data. We have cropped a small sequence of 40 s (200 m of vehicle movement) at the start of the race (see Fig. 6). 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 2. Fig. 7 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 analyzing how the scenario contained in this MIT sequence is reconstructed.

Fig. 6. MIT sequence.

Table 2
Information on each of the locations defined in the MIT sequence. Columns description: *pt*, number of points; size, memory size in mega bytes; *t*, mission time in seconds; *d*, traveled distance in meters.

Location name	Location snapshot		Sequence accumulated			
	pt ($\times 10^6$)	Size (MB)[a]	*pt* ($\times 10^6$)	Size (MB)[a]	*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

[a] 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 additional data is present in the message, such as the time stamp, the coordinate frame identification, etc.

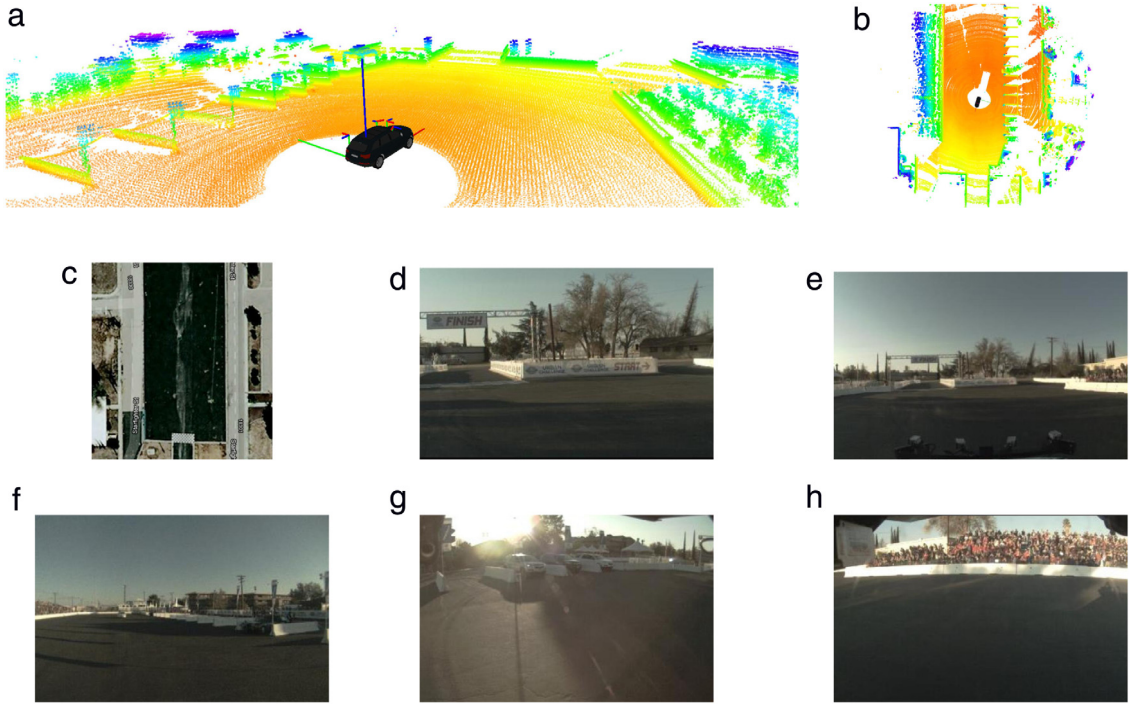

Fig. 7. Location *C* of the MIT sequence: (a) 3d view; (b) top view; (c) satellite view of the location; (d) front 6 mm camera; (e) front; (f) rear; (g) left (h) right.

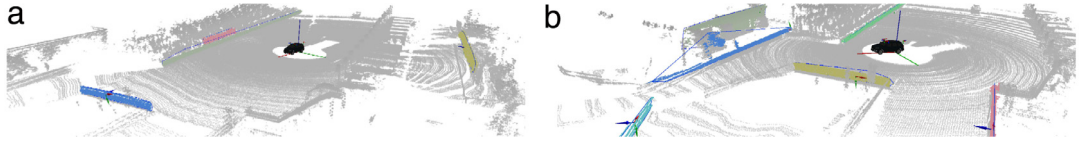

Fig. 8. Detection of geometric polygonal primitives in the data-sets of the MIT sequence: (a) location *C*; (b) location *D*.

Fig. 8 shows the polygonal primitives detected at locations *C* and *D*. It is possible to observe that most of the relevant planes are picked up by the algorithm.

4.1. One-shot reconstruction

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 as the primitives are extracted. In Fig. 9 an analysis of the computation time of each primitive is displayed. Primitives with higher numbers are detected in posterior phases.

Fig. 9(a) shows the number of points remaining in the input point cloud as a function of the polygon number. Results are shown for all locations in the MIT sequence. 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 larger for primitives detected earlier in the process. 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 behavior 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 accelerates the subsequent

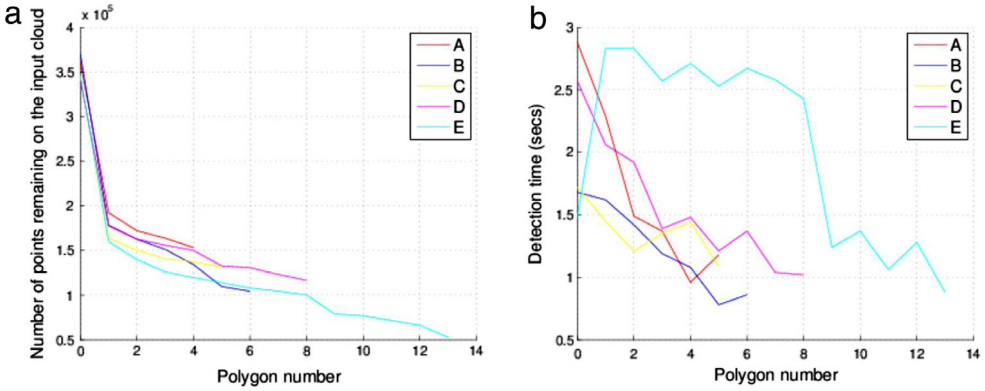

Fig. 9. Cascade processing analysis for MIT sequence locations A through E: (a) the number of points left to process for a given input point cloud, as a function of the index of the detected primitive; (b) the time it takes to perform the detection of each of the geometric polygonal primitives as a function of the primitives index.

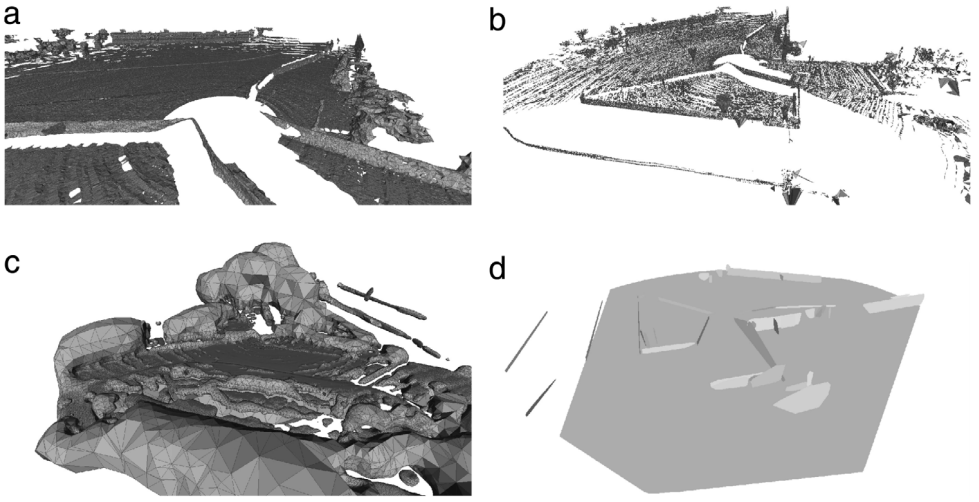

Fig. 10. Reconstruction of location *E* of MIT sequence: (a) BPA; (b) GT; (c) POIS; (d) GPP2. The observation of Fig. 16(e) may help the reader better understand the viewpoints in these images.

detection stages of the cascade. The detection time per primitive is shown in Fig. 9(b). The detection time tends to decrease with the increase in polygon number, for the reasons that were previously reported.

We compare the proposed approach with three surface reconstruction methodologies: *Ball Pivoting Algorithm* (BPA) [24], *Greedy Triangulation* (GT) [7] and *Poisson Surface Reconstruction* (POIS) [25]. Two different parameter configurations for the proposed approach are used. In the first 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, the accuracy or completeness of the scene representation is not very good. The second alternative, GPP 2, is configured so that even small polygons are detected, which should provide a more accurate scene description at the cost of a higher computation time.

Fig. 10(a) shows that the BPA method Fig. 10(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 3 shows the computation times that 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 analyzes 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.

Table 3
Comparison of the computation time of several approaches for surface reconstruction on the MIT sequence. Results were obtained with an i7-860 2.8 GHz quad core processor, Ubuntu operating system.

Sequence/Location	Processing time (s)				
	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
Average	585.9	151.8	58.3	25.5	35.5

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:

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

where sup and inf are the supremum and infimum, respectively. In this particular case, a variation of the Hausdorff distance, called the one sided Hausdorff distance is used where only the $\sup_{x\in X}\left(\inf_{y\in Y} d(x, y)\right)$ 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.

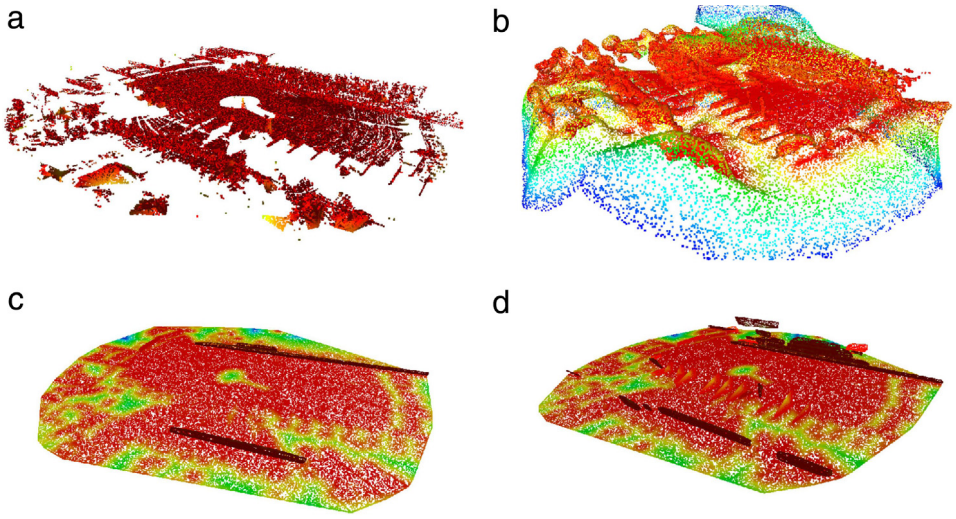

Fig. 11. Qualitative analysis of the one sided Hausdorff distance in location *C* sequence 1: (a) GT; (b) POIS; (c) GPP 1; (d) GPP 2; A Red–Green–Blue color map is used to code the distance. Red represents zero distance and blue maximum distance. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

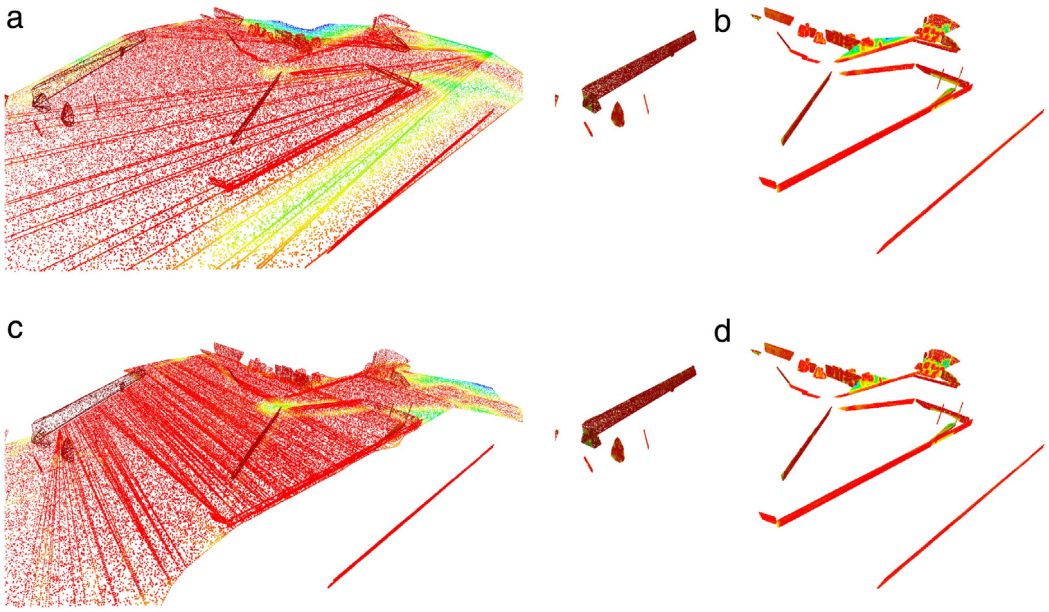

Fig. 12. Results from the Hausdorff distance obtained when using alternatives for the GPP 2 method for location *E*, sequence 1: (a) the standard GPP 2, with ground plane and convex hull; (b) discarded ground plane, convex hull; (c) with ground plane, concave hull; (d) discarded ground plane, concave hull. A Red–Green–Blue color map is used to code the distance. Red represents zero distance and blue maximum distance. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4 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 mean results is GT with an average error of 0.14 m. The accuracy exhibited by the GPP 1 and GPP 2 approaches is about 0.99 and 0.83 m, respectively. Fig. 11 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. 11(a), corresponding to the GT approach, almost all points have red color, resulting in low mean error. The POIS approach, represented in Fig. 11(b), shows many 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, 11(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; that occurs because 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 5. We can observe that, with the option of ground plane suppression and concave hull, the mean accuracy of GPP 2 improves to 0.1 m.

Fig. 12 shows a visual analysis of the Hausdorff distance errors for these variations of GPP 2. It is possible to observe that regions

Table 4
Comparison of the accuracy of the several approaches using BPA results as ground truth and Hausdorff distance as metric.

Location	Hausdorff distance (m)							
	GT		POIS		GPP 1		GPP 2	
	max	mean	max	mean	max	mean	max	mean
A	11.7	0.15	14.0	1.39	7.6	1.02	7.6	0.87
B	11.8	0.12	14.1	1.39	12.7	0.94	12.6	0.81
C	12.7	0.18	13.9	1.06	8.9	0.87	8.9	0.69
D	13.8	0.10	13.9	1.90	7.6	0.86	7.6	0.69
E	12.5	0.14	14.0	1.42	14.0	1.25	14.0	1.11
Average	12.5	0.14	13.9	1.43	10.2	0.99	10.1	0.83

Table 5
Comparison of the Hausdorff distance accuracy of the GPP 2 approach using: the standard approach, convex hull and ground plane included (also in Table 4); 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 (m)							
Hull	Convex		Convex		Concave		Concave	
Ground plane	Included		Not included		Included		Not included	
Location	max	mean	max	mean	max	mean	max	mean
A	7.6	0.87	1.8	0.15	6.8	0.71	1.2	0.13
B	12.6	0.81	1.5	0.11	12.6	0.53	1.1	0.08
C	8.9	0.69	1.9	0.16	6.6	0.52	1.9	0.12
D	7.6	0.69	2.2	0.14	7.3	0.59	2.1	0.11
E	14.0	1.11	1.7	0.10	8.8	0.32	1.4	0.08
Average	10.1	0.83	1.8	0.13	8.4	0.53	1.5	0.10

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.

4.2. Incremental reconstruction

This section presents several results and analysis of the expansion mechanism of the geometric polygonal primitives. The polygonal primitives algorithm without the expansion mechanism is compared against the same algorithm using the expansion mechanism. These two algorithms will be referred to as "with expansion" and "without expansion". By using this evaluation, it is possible to assess what are the benefits or disadvantages of the expansion mechanism. All five locations of sequence 1 are used to obtain results. Parameters used in the detection are similar to the GPP 1 set.

Fig. 13 shows a reconstruction of the scene using the expansion mechanism. The state of the reconstructed scenario at each location is shown. One clear advantage of this representation is that there are no overlapping primitives. A qualitative analysis of the results present in Fig. 13 shows that the most important features of the scenario are contained in the representation, especially if the task in mind is navigation.

To evaluate the computational complexity of the proposed algorithms, we measure the amount of detail of the representation that they produce when given a fixed amount of time to reconstruct a scene. Fig. 14(a) shows the number of polygons created in both algorithms. Although at the beginning of the sequence both algorithms generate a similar number of primitives, after some locations the algorithm without expansion shows a greater number of primitives. The explanation is that since the algorithm without expansion does not compare the stored primitives with the new data, it ends up duplicating primitives. On the contrary, when using the expansion mechanism, the duplication of primitives is avoided which leads to a smaller number of primitives. Note that just because a representation has more primitives it is not necessarily better. In fact, if there are duplicated primitives, the representation may be worse than another with a smaller number of non duplicated primitives. Fig. 14(b) shows the number of points that are given as input for the detection mechanism. In the case of the algorithm without expansion, all of the input points are passed on to the detection component, i.e., approximately 400 000 points per scan. When the expansion is active, a large number of points are explained by the expansion component and are not fed into the detection. In conclusion, the expansion mechanism takes a small amount of time when compared to the detection and other processes, and is able to quickly explain a large portion of the input points, thus filtering out many points which are not sent to other slower components. Finally, in Fig. 14(c), the total accumulated area of the primitives is shown. There is a clear difference between the with and without expansion approaches. This difference is due to the duplication of primitives in the case of the without expansion approach.

4.3. Comparison of approaches with and without expansion

Next, we focus on characterizing how the polygonal primitives evolve. Only the algorithm with expansion is portrayed, since in the without expansion approach primitives are static. Fig. 15(a) shows the number of support points assigned to each primitive. Only primitives of even index are shown. We can see that primitives are initialized in different locations in the sequence: at location *A* primitive 0 is created, while primitives 2, 4 and 6 are created at location *B*. These results show that most primitives significantly increase their number of support points throughout the sequence: Primitive 0 was detected at location *A* with $0.4 \times 50 \times 10^4 = 200$ *k*points, and at location *E* it already supported $1.4 \times 50 \times 10^4 = 700$ *k*points. In other words, it increased the number of support points by 350%. Another example, primitive 10, detected at location *D* with 3 *k*points, has at location *E* around 7 *k*points. A 230% increase between consecutive locations. The same analysis holds when considering the area of the primitives (see Fig. 15(b)): significant increases in the area of the primitives bounding polygons are also observed. All these observations, both in number of support points as well as in terms of area, show that polygons grow considerably after being detected. If these primitives were not expanded, the additional support points and area would have to be handled by a detection mechanism.

Fig. 16 shows how primitive 0, i.e., the ground plane primitive, evolves over sequence 1. The primitive expands at every iteration to accommodate newly observed data points that belong to the ground plane.

Algorithm 2 Cascade configuration for the expansion of existing geometric polygonal primitives and detection of new ones.

Require: Current list of geometric polygonal primitives $\mathbf{G} = \{\mathcal{G}^0, \mathcal{G}^1, ..., \mathcal{G}^n\}$
Require: $\mathcal{P}$, the input point cloud, containing new range measurements
Ensure: The updated list of geometric polygonal primitives $\mathbf{G} = \{\mathcal{G}^0, \mathcal{G}^1, ..., \mathcal{G}^n, \mathcal{G}^{n+1}..., \mathcal{G}^{n+m}\}$, where n is the number of primitives contained in the current representation and m is the number of additional primitives that are found in this iteration
for $i = 0 \rightarrow n$ **do** ▷ Expansion of primitive $\mathcal{G}^i$
Find the orthogonal distances d from all points in $\mathcal{P}$ to the support plane of primitive $\mathcal{G}^i$
Compute a new point cloud $\mathcal{P}_{ort}$ containing all points of $\mathcal{P}$ whose distance is smaller than the perpendicular expansion threshold (T_{ort}), $\mathcal{P}_{ort} = \{p_j \in \mathcal{P} \mid d_j < T_{ort}\}$
Project all points in $\mathcal{P}_{ort}$ to support plane of primitive $\mathcal{G}^i$, obtain projected point cloud $\mathcal{J}$ defined in $\mathbb{R}^2$
Initialize cycle break flag, *cycle_break* $\leftarrow$ `false`
Initialize number of expansion iterations, $it \leftarrow 0$
while *cycle_break* = `false` **do** ▷ Iterative longitudinal expansion
Expand bounding polygon, P^i_{it}, and obtain expanded polygon $\hat{P}^i_{it}$
Compute $\mathcal{J}^{in}_{it}$, the points from $\mathcal{J}$ that are inside $\hat{P}^i_{it}$
if $\mathcal{J}^{in}_{it} = \{\}$ **then** ▷ no expansion occurred, break cycle
break_cycle $\leftarrow$ `true`
Update primitive $\mathcal{G}^i$, by defining the updated bounding polygon given by P^i_{it}, recomputing the support plane using *Principal Component Analysis*(PCA) over the old and the new inliers, e.g, the points whose projections are points contained in the list $\{\mathcal{J}^{in}_0, ..., \mathcal{J}^{in}_{it}\}$
else
Remove all points in $\mathcal{J}^{in}_{it}$ from $\mathcal{J}$
Remove all points from $\mathcal{P}$ whose projections are points in $\mathcal{J}_{in}$
Compute the bounding polygon of the next iteration, P^i_{it+1}, from the convex hull of $\{P^i_{it}, \mathcal{J}_{in}\}$
end if
increment number of iterations, $it \leftarrow it + 1$
end while
end for
From the input point cloud $\mathcal{P}$ continue to perform detection of new polygonal primitives $\mathcal{G}^{n+1}..., \mathcal{G}^{n+m}$, adding them to list of primitives **G**,
Execute algorithm 1

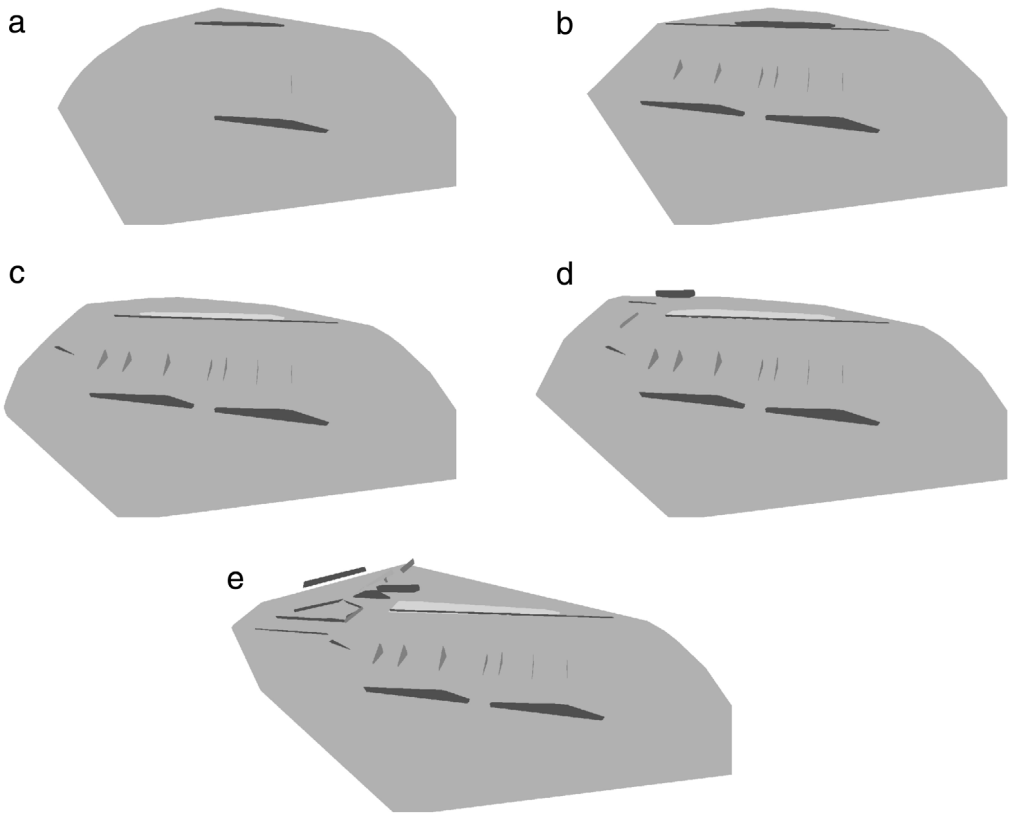

Fig. 13. GPP reconstruction for sequence 1: (a) The reconstructed scenario after the input point cloud of location *A* is received; (b) after location *B*; (c) after location *C*; (d) after location *D*; (e) after location *E*.

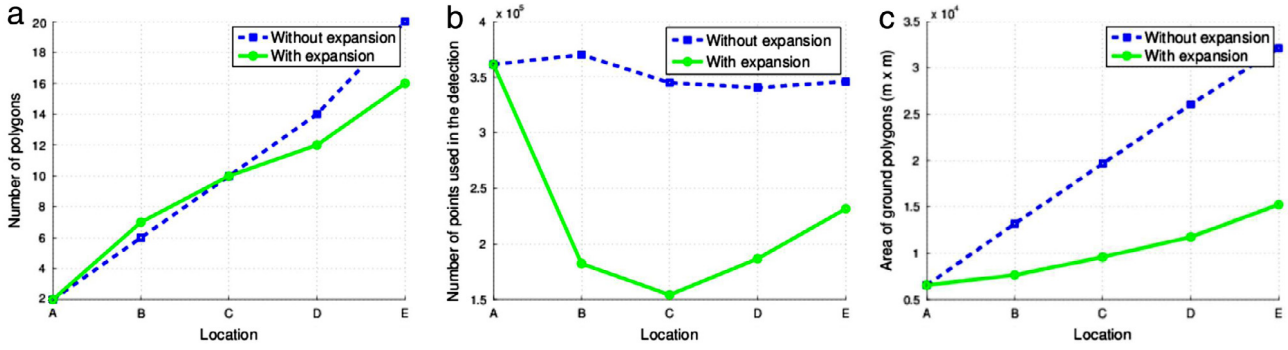

Fig. 14. (a) Comparison between the number of polygons generated by the algorithm using expansion and not using expansion through sequence 1; (b) number of points to use as input to the detection in both cases; (c) Total area of the detected polygons.

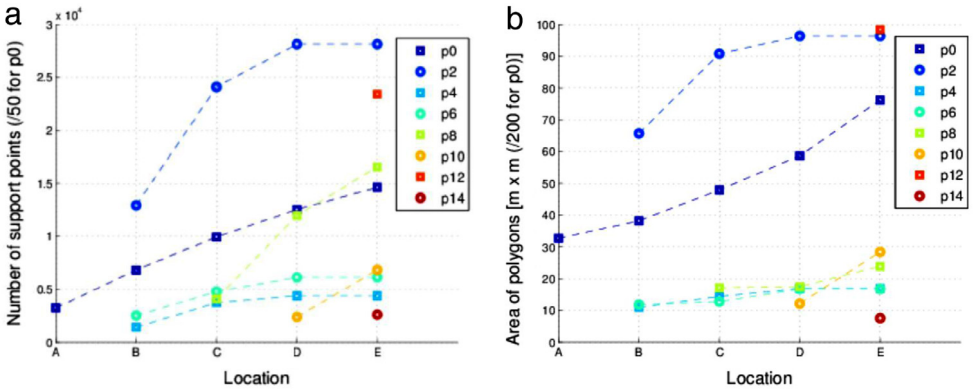

Fig. 15. Analysis of the evolution of each of the polygons through sequence 1. Only pair index polygons are shown to simplify the graphs: (a) number of support points per polygon; (b) total area of the primitives.

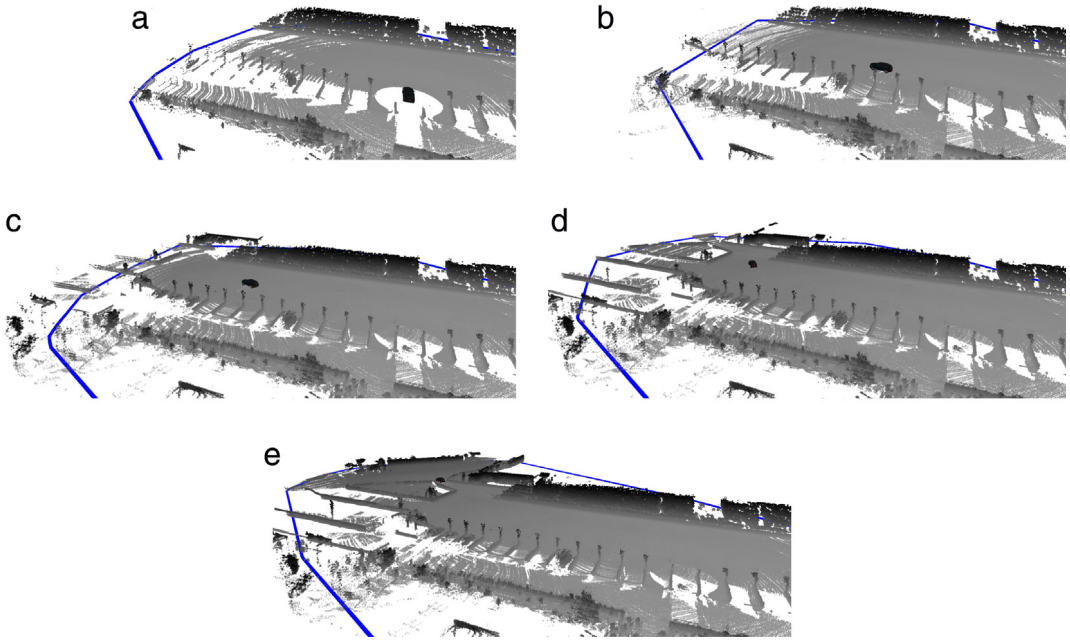

Fig. 16. Evolution of polygonal primitive 0 (the ground plane) through sequence 1: (a) location *A*; (b) location *B*; (c) location *C*; (d) location *D*; (e) location *E*.

Table 6 provides the links for some videos that show how the system processes the data stream from the MIT sequence. It is possible to see the difference between an incremental versus a non incremental (without expansion) approach.

5. Conclusions

This paper proposes a novel approach to produce scene representations using the array of sensors on-board autonomous

Table 6
Online videos showing the system running for the MIT sequence. The non incremental approach corresponds to the continuous processing without using the expansion mechanism. It is possible to observe that because there is no expansion, the primitives are duplicated.

URL	Description
https://youtu.be/hI_tiCYEprk	Input data from MIT sequence
https://youtu.be/a1mGGbAiNsk	Incremental approach
https://youtu.be/_HPLh3lPz7M	Non incremental approach

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. Furthermore, we propose mechanisms designed to update the polygonal primitives as new sensor data is collected. 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. The proposed expansion mechanism updates previous descriptions of the scene, therefore not creating duplicate representations of the same objects. In addition to this, the expansion mechanism is capable of efficiently filtering out data points that otherwise would be handled by (slower) detection mechanisms.

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 endeavors such as recognizing patterns in the scene.

Acknowledgments

This work has been supported by the Portuguese Foundation for Science and Technology "Fundação para a Ciência e Tecnologia" (FTC), under grant agreements SFRH/BD/43203/2008 and SFRH/BPD/109651/2015 and projects POCI-01-0145-FEDER-006961 and UID/CEC/00127/2013. This work was also financed by the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020. 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", reference CEB-02502014.

References

[1] A. Birk, N. Vaskevicius, K. Pathak, S. Schwertfeger, J. Poppinga, H. Buelow, 3-d perception and modeling, IEEE Robot. Autom. Mag. 16 (4) (2009) 53–60.
[2] R.B. Rusu, S. Cousins, 3D is here: Point Cloud Library (PCL), in: IEEE International Conference on Robotics and Automation, ICRA, Shanghai, China, 2011.
[3] W. Burgard, P. Pfaff, Editorial: Three-dimensional mapping, part 1, J. Field Robot. 26 (10) (2009) 757–758.
[4] M. Oliveira, V. Santos, A.D. Sappa, P. Dias, Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, Volume 1, Springer International Publishing, Cham, 2016, Ch. Scene Representations for Autonomous Driving: An Approach Based on Polygonal Primitives, pp. 503–515.
[5] M. Oliveira, V. Santos, A. Sappa, P. Dias, A.P. Moreira, Incremental texture mapping for autonomous driving, Robot. Auton. Syst. (2016) (submitted January 2016).
[6] A.S. Huang, M. Antone, E. Olson, L. Fletcher, D. Moore, S. Teller, J. Leonard, A High-rate, Heterogeneous Data Set from the DARPA Urban Challenge, Int. J. Robot. Res. 29 (13) (2011) 1595–1601.
[7] Z.C. Marton, R.B. Rusu, M. Beetz, On fast surface reconstruction methods for large and noisy datasets, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, Kobe, Japan, 2009.
[8] T. Weiss, B. Schiele, K. Dietmayer, Robust driving path detection in urban and highway scenarios using a laser scanner and online occupancy grids, in: Intelligent Vehicles Symposium, 2007 IEEE, 2007, pp. 184–189.
[9] F. Oniga, S. Nedevschi, Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection, IEEE Trans. Veh. Technol. 59 (3) (2010) 1172–1182.
[10] K. Zhou, M. Gong, X. Huang, B. Guo, Data-parallel octrees for surface reconstruction, IEEE Trans. Vis. Comput. Graphics 17 (5) (2011) 669–681.
[11] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C.M. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G.R. Bradski, B. Davies, S. Ettinger, A. Kaehler, A.V. Nefian, P. Mahoney, Stanley: The robot that won the darpa grand challenge, J. Field Robot. 23 (9) (2006) 661–692.
[12] C. Urmson, J. Anhalt, D. Bartz, M. Clark, T. Galatali, A. Gutierrez, S. Harbaugh, J. Johnston, H. Kato, P.L. Koon, W. Messner, N. Miller, A. Mosher, K. Peterson, C. Ragusa, D. Ray, B.K. Smith, J.M. Snider, S. Spiker, J.C. Struble, J. Ziglar, W.R.L. Whittaker, A robust approach to high-speed navigation for unrehearsed desert terrain, J. Field Robot. 23 (8) (2006) 467–508.
[13] C. Urmson, J. Anhalt, H. Bae, J.A.D. Bagnell, C.R. Baker, R.E. Bittner, T. Brown, M.N. Clark, M. Darms, D. Demitrish, J.M. Dolan, D. Duggins, D. Ferguson, T. Galatali, C.M. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, S. Kolski, M. Likhachev, B. Litkouhi, A. Kelly, M. McNaughton, N. Miller, J. Nickolaou, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, V. Sadekar, B. Salesky, Y.-W. Seo, S. Singh, J.M. Snider, J.C. Struble, A.T. Stentz, M. Taylor, W.R.L. Whittaker, Z. Wolkowicki, W. Zhang, J. Ziglar, Autonomous driving in urban environments: Boss and the urban challenge, J. Field Robot. 25 (8) (2008) 425–466. Special Issue on the 2007 DARPA Urban Challenge, Part I.
[14] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, S. Thrun, Junior: The stanford entry in the urban challenge, J. Field Robotics (2008).
[15] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, S. Cacciola, P. Currier, A. Dalton, J. Farmer, J. Hurdus, S. Kimmel, P. King, A. Taylor, D.V. Covern, M. Webster, Odin: Team victortango's entry in the darpa urban challenge, J. Field Robot. 25 (8) (2008) 467–492.
[16] T. Luettel, M. Himmelsbach, F. von Hundelshausen, M. Manz, A. Mueller, H.-J. Wuensche, Autonomous Offroad Navigation Under Poor GPS Conditions, in: Proceedings of 3rd Workshop On Planning, Perception and Navigation for Intelligent Vehicles, PPNIV, IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 2009.
[17] Wikipedia, Google driverless car—Wikipedia, the free encyclopedia, [Online; accessed November 2015], 2015.
[18] R. Jovanovic, R. Lorentz, Compression of volumetric data using 3d delaunay triangulation, in: 2011 4th International Conference on Modeling, Simulation and Applied Optimization, ICMSAO, 2011, pp. 1–5.
[19] A. Specht, M. Devy, Surface segmentation using a modified ball-pivoting algorithm, in: 2004 International Conference on Image Processing, 2004. ICIP'04. Vol. 3, 2004, pp. 1931–1934.
[20] C. Yin, D. Gang, C. Zhi-quan, L. Hong-hua, L. Jun, J. Shi-yao, An algorithm of cuda-based poisson surface reconstruction, in: 2010 International Conference on Audio Language and Image Processing, ICALIP, 2010, pp. 203–207.
[21] Y.-L. Chen, S.-H. Lai, An orientation inference framework for surface reconstruction from unorganized point clouds, IEEE Trans. Image Process. 20 (3) (2011) 762–775.
[22] A. de Medeiros Brito, A. Doria Neto, J. Dantas de Melo, L. Garcia Goncalves, An adaptive learning approach for 3-d surface reconstruction from point clouds, IEEE Trans. Neural Netw. 19 (6) (2008) 1130–1140.
[23] C. Rivadeneyra, I. Miller, J. Schoenberg, M. Campbell, Probabilistic estimation of multi-level terrain maps, in: IEEE International Conference on Robotics and Automation, 2009. ICRA'09. 2009, pp. 1643–1648.
[24] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taubin, The ball-pivoting algorithm for surface reconstruction, IEEE Trans. Vis. Comput. Graphics 5 (4) (1999) 349–359.
[25] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: SGP'06: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, Eurographics Association, 2006, pp. 61–70.
[26] R. Triebel, W. Burgard, F. Dellaert, Using hierarchical em to extract planes from 3d range scans, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005, pp. 4437–4442. http://dx.doi.org/10.1109/ROBOT.2005.1570803.
[27] M.A. Fischler, R.C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, in: ACM, Los Angeles, California, 1981.
[28] A. Nurunnabi, D. Belton, G. West, Robust segmentation for multiple planar surface extraction in laser scanning 3d point cloud data, in: 2012 21st International Conference on Pattern Recognition, ICPR, 2012, pp. 1367–1370.
[29] S. Hert, S. Schirra, 2D convex hulls and extreme points, in: CGAL User and Reference Manual, 4.0 Edition, CGAL Editorial Board, 2012.
[30] A. Bykat, Convex hull of a finite set of points in two dimensions, Inform. Process. Lett. 7 (1978) 296–298.
[31] C.B. Barber, D.P. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, ACM Trans. Math. Software 22 (4) (1996) 469–483.

[32] O. Aichholzer, F. Aurenhammer, D. Alberts, B. Gartner, A novel type of skeleton for polygons, J.UCS 1 (12) (1995) 752–761.
[33] F. Cacciola, 2D straight skeleton and polygon offsetting, in: CGAL User and Reference Manual, 4.0 Edition, CGAL Editorial Board, 2012.

Miguel Oliveira received the Mechanical Engineering and M.Sc. in Mechanical Engineering degrees from the University of Aveiro, Portugal, in 2004 and 2007, where later in 2013 he obtained the Ph.D. in Mechanical Engineering specialization in Robotics, on the topic of autonomous driving systems. Currently he is a researcher at the Institute of Electronics and Telematics Engineering of Aveiro, Portugal, where he works on visual object recognition in open-ended domains. His research interests include multimodal sensor fusion, computer vision and robotics.

Vítor Santos obtained a 5 year degree in Electronics Engineering and Telecommunications in 1989, at the University of Aveiro, Portugal, where he later obtained a Ph.D. in Electrical Engineering in 1995. He was awarded fellowships to pursue research in mobile robotics during 1990–1994 at the Joint Research Center, Italy. He his currently Associate Professor at the University of Aveiro and lectures courses related to advanced perception and robotics, and has managed research activity on mobile robotics, advanced perception and humanoid robotics, with the supervision or cosupervision of more than 100 graduate and undergraduate students, and more that 120 publications in conferences, books and journals. At the University of Aveiro he has coordinated the ATLAS project for mobile robot competition that achieved 6 first prizes in the annual Autonomous Driving competition and has coordinated the development of ATLASCAR, the first real car with autonomous navigation capabilities in Portugal. He is one of the founders of Portuguese Robotics Open in 2001 where he has kept active participation ever since. He his also cofounder of the Portuguese Society of Robotics, and participated several times in its management since its foundation in 2006. His current interests extend to humanoid robotics and the application of techniques from perception and mobile robotics to autonomy and safety in ADAS contexts.

Angel Domingo Sappa (S'94-M'00-SM'12) received the Electromechanical Engineering degree from National University of La Pampa, General Pico, Argentina, in 1995, and the Ph.D. degree in Industrial Engineering from the Polytechnic University of Catalonia, Barcelona, Spain, in 1999. In 2003, after holding research positions in France, the UK, and Greece, he joined the Computer Vision Center, Barcelona, where he is currently a Senior Researcher. He is a member of the Advanced Driver Assistance Systems Group. His research interests span a broad spectrum within the 2D and 3D image processing. His current research focuses on stereoimage processing and analysis, 3D modeling, and dense optical flow estimation.

Paulo Dias graduated from the University of Aveiro Portugal in 1998 and started working in 3D reconstruction at the European Joint research Centre in Italy. In September 2003, he concluded his Ph.D. with the thesis ''3D Reconstruction of real World Scenes Using Laser and Intensity Data''. He is currently an assistant professor within the Department of Electronics Telecommunications and Informatics (DETI) and is involved in several works and projects within the Institute of Electronics and Informatics Engineering of Aveiro (IEETA) related to 3D Reconstruction, Virtual Reality, Computer Vision, Computer Graphics, Visualization and Combination and Fusion of data from multiple sensors.

António Paulo Moreira graduated with a degree in electrical engineering at the University of Oporto, in 1986. He then pursued graduate studies at University of Porto, obtaining an M.Sc. degree in electrical engineering-systems in 1991 and a Ph.D. degree in electrical engineering in 1998. Presently, he is an Associate Professor at the Faculty of Engineering of the University of Porto and researcher and manager of the Robotics and Intelligent Systems Centre at INESC TEC. His main research interests are process control and robotics.