2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 7-12, 2012. Vilamoura, Algarve, Portugal

Fast Incremental Clustering and Representation of a 3D Point Cloud Sequence with Planar Regions

Francesco Donnarumma, Vincenzo Lippiello, Matteo Saveriano

Abstract— **An incremental clustering technique to partition 3D point clouds into planar regions is presented in this paper. The algorithm works in real-time on unknown and noisy data, without any initial assumption. An iterative cluster growing technique is proposed in order to correctly classify a flow of 3D points and to merge close regions. The computational efficiency of the approach is achieved by using an Incremental Principal Component Analysis (IPCA) technique, and with the adoption of a compact geometrical representation based on the concave-hull computation of each cluster. This solution adds a more realistic representation of the observed environment and reduces the number of points needed to identify the cluster shape. The effectiveness of the proposed algorithm has been validated with both synthetic and real data sets.**

I. INTRODUCTION

The construction of an environment map is a key aspect for the autonomous navigation of mobile and flying robots, especially in indoor or cluttered environment. The ability to recover the geometrical structure of visible surfaces is critical for scene understanding.

Many robot mapping algorithms have focused on semantic maps based on point clouds or grid-based representations. However, both these representations are extremely dense, i.e. the number of points or grid cells required to represent an environment generally scales with the *volume* of the environment and not with its *complexity* (e.g. the number of objects detected). Hence, those approaches would rely on many thousands of variables, and thus they could make a reliable *fast* reconstruction (possibly in real-time) intractable already for environments with a limited extension. On the other hand, an alternative solution relies on the adoption of a compact geometric representation of the environmental structures, which can be done using few geometrical primitives together with a small number of parameters.

Our challenge is to develop a fast approach to be processed by low-cost and low-resources hardware of a Micro Unmanned Aerial Vehicle (MUAV), and at the same time to be employed by an autonomous navigation control system. Thus, a suitable environmental representation should be abstract at such a level that one can discard the storing of thousands of points, which are acquired during the execution, and should be adapted to sparse and noisy points (and with drift), which are collected by a 3D stereo vision systems. To achieve this goal, we combine a RANSAC plane growing together with and Incremental Principal Component Analysis (IPCA) algorithm, and a shape retrieval by a Concave Hull algorithm. Our contribution is in selecting and combining different approaches in a novel procedure. Simulation and tests on real data will prove the robustness of the proposed approach with respect to noise and outliers.

A. Related Work

In the last decade, several approaches have been proposed in order to address this problem. PCA can be used achieving an efficient representation of the point clouds. In [1] this approach is employed in a neighborhood of data points, and then used incrementally. A combination of region growing and plane fitting is proposed in [2], [3]. In [4] a computationally expensive 3D Hough transform is used for plane detection, while in [5] the Radon transform is deployed to detect planes in volume data.

Other plane extraction algorithms are highly specialized for a specific application and are not in widespread use. An Expectation Minimization (EM) algorithm to fit planes, which are initially randomly generated, is used in [6]. In [7] lines and in [8] planes are detected relying on the specific properties of a laser scanner.

A combination of PCA and *G*-means (a modified *K*-means technique) is proposed in [9] to automatically detect the number of planes. Other approaches, such as *K*-planes [10] and its generalization *K*-subspaces [11], which are both a modification of the *K*-means algorithm, try to fit plane-subspaces resembling PCA error function.

Other solutions employ triangle meshes for the preprocessing of the original point data [12]. In [13] geometric primitives are fit into triangle meshes. The proposed prototype works for planes, cylinders and spheres and is easily extensible to other primitives, but it is computationally expensive. The Matrix Factorization (MF), which is an extension of PCA from one to multiple subspaces, is proposed in [14] in the case of independent and linear subspaces.

The Sparse Subspace Clustering (SSC) [15] approach is based on the idea of writing a data point as a linear combination of a *sparse* linear combination of all other data

Francesco Donnarumma is with Istituto di Scienze e Tecnologie della Cognizione, CNR, via S. Martino della Battaglia, 00185, Rome, Italy francesco.donnarumma@istc.cnr.it.

Vincenzo Lippiello is with PRISMA Lab, Dipartimento di Informatica e Sistemistica, Università degli Studi di Napoli Federico II, via Claudio 21, 80125, Naples, Italy lippiello@unina.it

Matteo Saveriano is with CoTeSys Lab, Fakultät für Elektrotechnik und Informationstechnik, Technische Universität München, Karlstrasse, 80333, Munich, Germany matteo.saveriano@tum.de.

The research leading to these results has been supported by the European Community's Seventh Framework Programme inside the AIRobots and Goal-Leaders collaborative projects (FP7/2007-2013) under grant agreements ICT-248669 and ICT-270108, respectively. The authors are solely responsible for its content. It does not represent the opinion of the European Community and the Community is not responsible for any use that might be made of the information contained therein.

points. However this approach scales with the number of input points and thus is too slow for real-time purposes.

In [16] the Generalized Principal Component Analysis (GPCA) is proposed. In detail, it is an algebraic-geometric method for clustering data lying in subspaces by fitting a union of K subspaces with a set of polynomials of degree K, whose derivatives at a point give a vector normal to the subspace containing that point, and then proceed on the segmentation of these vectors.

In real applications, data is corrupted by the presence of noise, outliers, etc. While robust estimation techniques have been developed for the case of a single subspace, the case of multiple subspaces is still an open issue. For this purpose, the Random Sample Consensus (RANSAC) algorithm [17] is often employed. For example, in [18] the authors adapted RANSAC for plane extraction. This algorithm performs precise and fast plane extraction, but only if the clustering parameters have been fine-tuned properly. Moreover, for the optimization process a knowledge is assumed which is not readily available in point cloud data, such as normals, neighboring relations and outlier ratios. A faster RANSAC variant extending the previous work is given in [19].

II. REAL TIME INCREMENTAL CLUSTERING

Approaches that use point clouds or grid-based representation as environmental maps would rely on many thousands of variables, and often become intractable for real-time reconstruction. A solution is to rely on a compact geometric representation of the environmental structures, which can be done using few geometric primitives[1]. This solution requires that 3D points acquired by the robot should be collected (*clustered*) in homogeneous groups, and then represented with geometric parameters. The crucial point to speed up the algorithm is that representation should be used incrementally on new points, without recurring on all so far points that have been presented during execution.

Given a large number of point features corresponding to the same environmental structure, a dimensionality-reduction technique to extract a higher-order, lower-dimensional *representation*, that still captures the data, should be employed [9], [20]. Therefore, there is the need to simultaneously cluster data into multiple subspaces, and to find a low-dimensional subspace fitting each group of points (*dimensionality reduction*). In machine learning literature this is known as a *subspace clustering* problem [21]. A typical algorithm to model data by multiple subspaces would require:

- to detect how many planes are in the same scene;
- to cluster the points of the scene in different planes;
- to represent planes in a point-independent manner.

Thus, sequences of scenes are incrementally combined together with the advantage of speeding up the algorithm without decreasing the performance. The main components of the proposed fast incremental-clustering algorithm are described in the following subsections.

[1]Typical indoor environments usually comprise a large amount of planar surfaces. Hence, with the adoption of *closed and limited planar region* geometric primitives, the representation of the main structures present in the environment can be achieved.

A. Subspace clustering problem definition

The subspace clustering problem requires modeling of a collection of data points in a union of subspaces. Let $\mathbf{x}_i \in \mathbb{R}^n$, with $i = 1, \ldots, N$, be a set of points drawn from an unknown union of K subspaces S_k such that

$$S_k = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \boldsymbol{\mu}_k + U_k \mathbf{y}\} \quad \text{with } k = 1, \ldots, K,$$

where $\boldsymbol{\mu}_k \in \mathbb{R}^n$ is a point of the subspace S_k, $U_k = \{\mathbf{u}_1; \ldots; \mathbf{u}_{d_k}\} \in \mathbb{R}^{n \times d_k}$ is a basis of S_k, and $\mathbf{y} \in \mathbb{R}^{d_k}$ is a low-dimensional representation of $\mathbf{x}$. When $K = 1$ the problem consists in finding $\boldsymbol{\mu} \in \mathbb{R}^n$, the basis $U \in \mathbb{R}^{n \times d}$, and the dimension d. The optimal solution (without noise/outliers) is the Principal Component Analysis (PCA) [22]. PCA computes a set of basis functions ordered in correspondence of the variance of the data, with $\boldsymbol{\mu} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n$, and produces a *reduced* representation of the data.

For a generic K, the solution is not straightforward. Hence, the problem with $n = 3$ (i.e. 3D point) and $d_k = 2$ (i.e. only points that form planes are selected, while all points that do not satisfy this requirement are treated as outlier) is considered. In other words, all surfaces are searched as an approximation of planes under a certain threshold.

There is a strong coupling between data segmentation and the geometric primitive estimation. Specifically, if the clustering of the data is known, one could easily fit a single subspace to each group of points using PCA. Conversely, if the subspace parameters are known, the data points that best fit each subspace can be easily found. In practice, neither the segmentation of the data nor the subspace parameters are known and both problems have to be simultaneously solved. This aspect becomes dramatic especially in indoor environments with many planar surfaces to be detected.

The RANSAC algorithm is employed to fit planes by PCA in a group of unlabeled points of the scene. At each image scene, the RANSAC Plane Growing algorithm is performed, which can be summarized in the following steps:

Algorithm 1 RANSAC Plane Growing

```
1. for $i = 1 \rightarrow maxExternalIterations$ do
2.    for $j = 1 \rightarrow maxInternalIterations$ do
3.       randomly take 3 unlabeled points from the scene
4.       calculate principal components of the plane through
         the 3 points
5.       project unlabeled points onto the plane
6.       label points under a certain distance threshold from
         the plane as a new plane
7.    end for
8.    select the plane with the larger number of the inliers
9. end for
```

This procedure finds a maximum of *maxExternalIterations* new planes in the scene. A small value (less than 10) is sufficient in most cases as this procedure is meant to be executed on the new unlabeled points received. The RANSAC parameter *maxInternalIterations* allows the selection of the best plane (i.e. with the largest number of points) in the scene. This is crucial in order to cope with noise and outliers. Points that do not fit any plane are left unlabeled.

B. Incremental cluster representation by IPCA

When considering different scenes, whenever new data points are presented, they should be clustered in the most efficient way. In the proposed approach, point storing explosion is avoided by using a procedure which relies on Incremental Principal Component Analysis (IPCA) [23]. In fact, with $N_k(t)$ the number of points belonging to the cluster before updating it, ICPA recursively computes, for all the new M points $\mathbf{x}_m$, with $m = N_k(t)+1, \ldots, N_k(t)+M$, added to the k-th cluster, the new centers

$$\boldsymbol{\mu}_k(t+1) = \frac{1}{N_k(t+1)} \left(N_k(t) \cdot \boldsymbol{\mu}_k(t) + \sum_{m=1}^{M} \mathbf{x}_m \right),$$

with $N_k(t+1) = N_k(t) + M$ and the new components $\mathbf{U}_k = \{\mathbf{u}_{1,k}, \mathbf{u}_{2,k}\}$ at step t with the recursive equations

$$\mathbf{v}_{i,m} = \frac{1}{m} \left((m-1)\boldsymbol{I} + \frac{(\mathbf{x}_m - \boldsymbol{\mu}_k)(\mathbf{x}_m - \boldsymbol{\mu}_k)^{\mathrm{T}}}{\|\mathbf{v}_{i,m-1}\|} \right) \mathbf{v}_{i,m-1}$$

with $\mathbf{v}_{i,0} = \mathbf{u}_{i,k}(0)$ and $i \in \{1,2\}$, and thus

$$\mathbf{u}_{i,k}(t+1) = \frac{1}{\left\|\mathbf{v}_{i,N_k(t+1)}\right\|} \mathbf{v}_{i,N_k(t+1)}$$

In practice, the first time that the cluster is detected a standard PCA is executed to compute $\mathbf{u}_{i,k}(0)$. Then, for all new points detected, IPCA is performed. We stress that with this approach the principal components of the clusters are iteratively estimated without requiring the storage of all 3D points. Thus, the finer bounding box of the clusters is computed on a fixed limited number of stored points selected on the basis of the following Concave Hull procedure.

C. Shape representation: concave hull estimation

In order to preserve the cluster shapes, the concave envelope, or *concave hull* of the clusters is computed. To this purpose, the shape retrieval step is based on the Edelsbrunner et al. [24] algorithm, which computes the concave hull of N 2D points with a complexity of $O(N \log N)$.

By projecting the cluster points onto the fitting plane, a set of 2D points is achieved and the 2D concave hull can be straightforwardly computed with known methods. In this way, a compact but realistic representation of the cluster, i.e. of the environment, with a reduced number of points needed to identify its shape, is achieved.

For example, for a representation of a planar region by means of the smallest *bounding box* surrounding the points, a single rectangle is sufficient to approximate the cluster shape. However, this solution is not adequate in many practical cases. Let us consider the case of the floor of a corridor with a turn, as shown in Fig. 1. By comparing the bounding-box representation (the green line) with the concave hull (blue line), it is clear that the latter allows the representation of the exact shape of the region with a limited number of points.

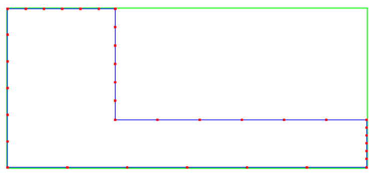


Fig. 1. Simulation of a turn in a corridor. The blue line is the concave hull, the green one is the bounding box.

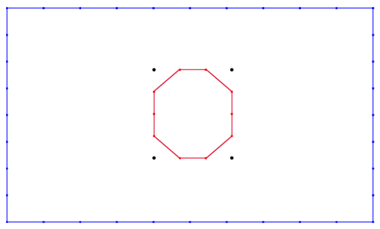


Fig. 2. Rectangle with a rectangular hole inside. The blue line is the *external* perimeter, the red one is the approximated cavity hull. The black points also belong to the cavity hull but the algorithm doesn't include them.

However, in large data sets it could be necessary to limit the number of the points employed to represent each cluster, in order to increase the computational performances. To this purpose, the area of the concave hull can be employed as a parameter to saturate the maximum number of points used to represent it. To compute the area of the concave hull the *Shoelace Theorem* can be used:

$$A = \frac{1}{2} \left| \sum_{i=1}^{N} (x_i y_{i+1} - x_{i+1} y_i) \right|,$$

with $\mathbf{x}_{N+1} = \mathbf{x}_1$, where $\mathbf{x}_i = (x_i, y_i)$ are the vertices of the concave hull.

An additional advantage of this approach is that, given a point cloud representing an indoor environment, it is possible, by setting the parameter α, to identify the escapes (e.g. opened doors and windows). In fact, these types of structures correspond to a *hole* in the cloud points. Hence, by computing the α-*shape*, it is possible to estimate the contour of any hole in the cloud with a desired accuracy. Figure 2 shows an example of a cloud with a rectangular hole, where the blue line represents the estimated rectangle perimeter, while the red one the estimated contour of the hole. Notice that four points (the black points in Fig. 2) of the real hole contour have not been included in the estimated one. This is due to the threshold used to compute the α-*shapes*. By increasing the threshold all points are collected. A good trade-off between the number of points to be stored and the desired accuracy has to be chosen.

D. Incremental matching of views

The joining of these partial views into a map is referred to as *scan registration*. Scan registration is subject to matching errors, sensor noise and systematic errors in the scans. Also this phase is done in an incremental way in order to reduce the execution time. In particular, new points are tested if belonging to older clusters. This is done by controlling if they belong to the same geometrical plane, and at the same time if they are close enough to previous points. After each scene acquisition, if possible, unlabeled new points are added to older clusters, and then new clusters are searched by means of the RANSAC Plane Growing algorithm only on the unlabeled points. Then, the cluster representations are recomputed by means of IPCA and Concave Hull algorithm. Thus, the resulting algorithm can be written as:

Algorithm 2 Incremental Matching

```
1. perform RANSAC Plane Growing on the first scene and
   find representation $U_k(0)$ for each cluster
2. while $next\_scene$ do
3.   $scene = actual\_scene(t)$
4.   project unlabeled points and label points under a
     certain distance threshold onto the planes $U_k(t)$
5.   ambiguous points (labeled to more than one plane) are
     assigned to the nearest plane in terms of:
       a) distance from the plane $\|\boldsymbol{\mu}_k(t)+U_k(t)(U_k(t))^T(\mathbf{x}_n-\boldsymbol{\mu}_k(t))-\mathbf{x}_n\|$
       b) distance from the centroids $\boldsymbol{\mu}_k(t)$ of the clusters (intra-
          cluster distance) $\sum_{h=1}^{H_k}\|\bar{\mathbf{x}}_{h_k}-\boldsymbol{\mu}_k(t)\|^2$
6.   perform RANSAC Plane Growing on the remaining
     unlabeled points searching for new Principal Compo-
     nents bases $U_{k+1}(t)$
7.   update representation:
       a) add new Principal Components bases $U_{k+1}(t)$ found,
          $k = k+1$
       b) perform IPCA computing new centers $\boldsymbol{\mu}_k(t+1)$ and
          Principal Components $U_k(t+1)$
       c) perform ConcaveHull for each cluster $k$ to save cluster
          shape in a collection of points $\{\hat{\mathbf{x}}_{h_k}\}_1^{H_k}$
8. end while
```

This further incremental phase allows the algorithm to rely only on the PCA representation, the principal components for each plane U_k and the bounding concave hull points.

III. EXPERIMENTAL RESULTS

A. Implementation details

The Incremental RANSAC PCA algorithm has been implemented in *C++* and tested on an *AMD Phenom II x4 945* processor. This implementation has been tested on a large database consisting in the point cloud flow provided by a stereo camera system presented in [25], which performs the visual odometry and the 3D points estimation at 10 Hz, during the navigation of an unmanned aerial vehicle within a real building (see Fig. 3(a)). The point cloud flow has been stored in a ".BAG" *ROS* (*Robot Operating System*[2]) file.

The proposed clustering algorithm has been encapsulated in a ROS node, that reads the point cloud from a topic linked with the BAG file. Moreover, another ROS node reads clustered point clouds and computes the concave hull of each cluster using the α*-shape* algorithm implementation of the *Point Cloud Library* (*PCL*)[3]. The concave hull estimation is performed only for the new planes or for those to which new points are added. This software architecture, consisting of three separated ROS nodes, allows the execution of each node in a separate thread with different scheduling priority. In this way, the Incremental RANSAC PCA algorithm is executed without losing any frame from the vision system.

B. Synthetic dataset: Reliability with noise and outliers

Several tests have been performed to compare the proposed approach with other plane clusterization methods. The other methods considered for the comparison are:

- *K*-subspaces [11] enforced with RANSAC;
- the classic K-means clusterization [26];
- Generalized Principal Component Analysis (GPCA) alone [16] and enforced with RANSAC;
- Matrix Factorization (MF) [14].

To evaluate the best performing algorithm in terms of execution time, a dataset with synthetic 3D data has been generated. For each $K \in \{2,\ldots,6\}$ planes, $M = 10$ synthetic scenes have been generated, for a total of 50 synthetic scenes per experiment. As for each plane, about 100 points are employed, hence in each scene there are from 200 to 1200 points. For each experiment we have produced different noise cases, Gaussian Noise and Outliers. In order to simulate Gaussian noise we considered for each point generated $\mathbf{p}$ a noisy point generated as $\mathbf{p}_n = \mathbf{p} + \sigma\mathcal{N}(0,1)$, with $\mathcal{N}(0,1)$ a Gaussian distribution centered in 0 with standard deviation 1. The parameter σ was chosen as a percentage of the standard deviation of the original non noisy data. Outliers are points uniformly distributed in the considered space that do not belong to any generated plane. We prepare corresponding datasets with 5% or 10% of Gaussian noise together with 0% or 8% of outliers. The performance of the algorithm has been estimated with an *accuracy* measure, i.e.

$$a = \frac{Number\ of\ correct\ guesses}{Number\ of\ samples} \quad \in [0,1],$$

with $a = 1$ in the case of a perfect clusterization.

As shown in Tab. I, in the absence of noise, the best performing approach is the GPCA, which is capable of reaching an accuracy close to 1. However, the execution time is slower than other approaches, especially while enforcing it with the RANSAC algorithm. The fastest algorithm remains the *K*-means, even if it must be stressed that for the tests of this algorithm the knowledge of *K* (the number of planes) is required. However, the problem of finding the right *K* is not trivial and requires a non-neglecting execution time that here has not been considered.

On the other hand, in presence of noise and outliers the performance of the algorithms decreases. The proposed RANSAC plane growing by IPCA, thanks to its incremental part, is more robust to noise and outliers and worsens less than other approaches. In general, while the other algorithms try to model each point, the incremental step of this algorithm is capable of finding the most reliable planes and of discarding ambiguous points. Moreover, by suitably tuning the threshold parameters and the iteration of the RANSAC, it remains very competitive also in terms of execution time.

[2]See `http://www.ros.org/wiki/` for further details.

[3]See `http://www.pointclouds.org/` for further details

TABLE I

COMPARISON OF SIX METHODS FOR PLANE CLUSTERING ON THE FIVE SYNTHETIC DATASETS

Case 1: no noise and no outliers

Method	TMF	$\bar{a}$	std(a)
Ransac K-subspaces	67.69	0.86	0.20
K-means	1.03	0.87	0.18
RansacPlane Growing by IPCA	1.70	0.95	0.04
GPCA	8.74	0.99	0.03
Ransac GPCA	80.39	1.00	0.18
Matrix Factorization (MF)	100	0.79	0.25

Case 2: 5% noise and no outliers

Method	TMF	$\bar{a}$	std(a)
Ransac K-subspaces	58.90	0.81	0.19
K-means	0.74	0.78	0.15
RansacPlane Growing by IPCA	2.25	0.91	0.10
GPCA	9.30	0.83	0.15
Ransac GPCA	85.29	0.95	0.06
Matrix Factorization (MF)	100	0.71	0.11

Case 3: 5% noise and 8% outliers

Method	TMF	$\bar{a}$	std(a)
Ransac K-subspaces	60.02	0.61	0.13
K-means	1.25	0.74	0.15
RansacPlane Growing by IPCA	3.55	0.87	0.17
GPCA	13.64	0.65	0.09
Ransac GPCA	138.19	0.84	0.13
Matrix Factorization (MF)	100	0.62	0.15

Case 4: 10% noise and no outliers

Method	TMF	$\bar{a}$	std(a)
Ransac K-subspaces	65.43	0.58	0.18
K-means	0.60	0.61	0.13
RansacPlane Growing by IPCA	3.57	0.66	0.12
GPCA	10.21	0.58	0.21
Ransac GPCA	90.50	0.66	0.17
Matrix Factorization (MF)	100	0.53	0.20

Case 5: 10% noise and 8% outliers

Method	TMF	$\bar{a}$	std(a)
Ransac K-subspaces	63.01	0.57	0.15
K-means	1.67	0.54	0.13
RansacPlane Growing by IPCA	5.42	0.63	0.15
GPCA	13.19	0.51	0.13
Ransac GPCA	139.56	0.60	0.19
Matrix Factorization (MF)	100	0.56	0.14

Legend: TMF are the means of execution times in % of MF; $\bar{a} \in [0, 1]$ is the mean accuracy; std(a) is the standard deviation of the mean accuracy. The mean time of execution of MF is $\bar{M}F = 3.45s$.

C. A Building 3D points dataset

The proposed Incremental RANSAC PCA algorithm is tested on a large dataset representing a real building. The dataset consists of 6000 frames, that produce about 30 new 3D points for each frame, resulting in more than 15000 points. At the end of the process the algorithm have clustered 48 different planes, as shown in Fig. 3. Figure 3(a) represents the point cloud obtained by performing the stereo vision algorithm, while Fig. 3(b) represents the results of the proposed algorithm (each plane is represented by its bounding box).

In the red circled box the concave hull on one of this planar regions is highlighted. In particular, thanks to the adoption of the concave hull representation, the real shape of a corridor is rightly estimated. The number of points requested to store this contour is very small compared with respect to the original point set corresponding to this surface.

A choice of $maxInternalIterations = 150$ was sufficient to cope with noise and outliers. The parameter α of the concave hull algorithm has been chosen equal to the size of the aerial vehicle employed for the building inspection. This is a reasonable choice in order to find escapes (windows, doors) useful for navigation. Moreover, a similar strategy has been followed for setting the threshold of the plane noise. This choice leads to a unique cluster for each entire staircase and it is exactly what we were looking for: in our case of a map-reconstruction for the autonomous navigation of a flying robot, we are not interested in the reconstruction of each step but in the slope of the stairs and in the distance from it for a safe navigation. In this regard, notice that all the boxes in Fig. 3(b) have three dimensions. In fact, the height of each bounding box represents the *standard deviation* of the point set with respect to the fitting plane. This information can be fully exploited to endure a safe navigation of an autonomous robot.

The required computational time is about 45 ms for each frame of this dataset, which is less than the period of 100 ms of the stereo acquisition vision system.

IV. CONCLUSION AND FUTURE WORK

In this paper a fast algorithm for the incremental clustering and representation with planar regions of a point cloud flow has been presented. The shape representation of each cluster is based on a reduced and bounded number of points stored by means of the concave hull evaluation. New representations of the planes have been computed by means of IPCA, which allows discarding of the old points already computed, without losing the past information on clusters. The performance of the proposed approach has been tested by stressing it both on a synthetic dataset and on a large 3D dataset representing a real building. It has been demonstrated that this approach is fast enough to perform a reconstruction at the same frame rate of a typical stereo vision system.

Our next research will focus on fast ways to introduce and recall *semantic labeling* in order to identify and distinguish different common properties, such as, walls, stairs, floors and ceilings, in order to construct a *high-level* semantic map.

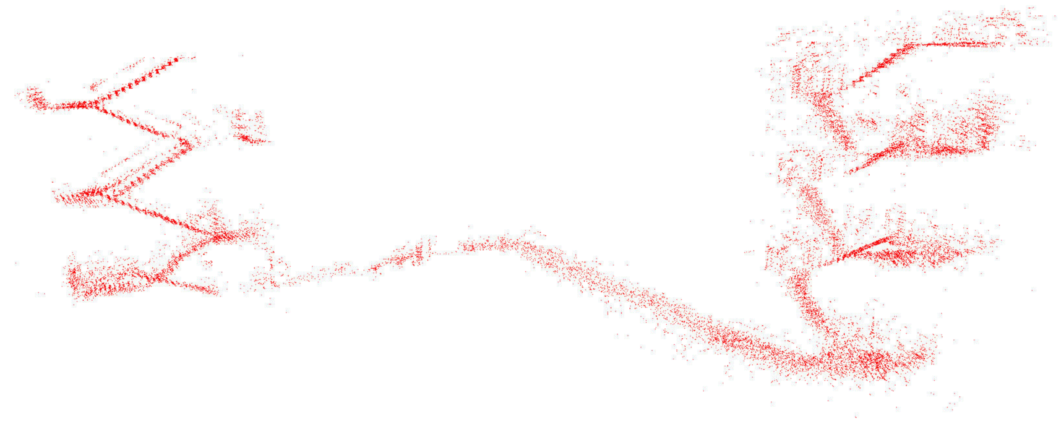
REFERENCES

[1] A.-L. Chauve, P. Labatut, and J.-P. Pons, "Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1261–1268, 2010.

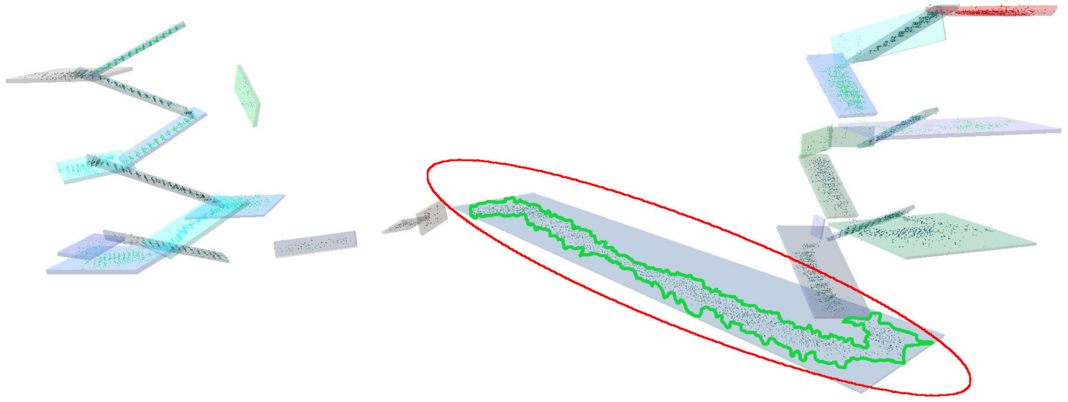
[2] D. Hähnel, W. Burgard, and S. Thrun, "Learning compact 3D models of indoor and outdoor environments with a mobile robot," *Robotics and Autonomous Systems*, vol. 44, pp. 15–27, 2003.

[3] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, "Fast plane detection and polygonalization in noisy 3D range images," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3378–3383, 2008.

[4] D. Borrmann, J. Elseberg, K. Lingemann, and A. Nüchter, "The 3D hough transform for plane detection in point clouds: A review and a new accumulator design," *3D Research*, vol. 2, pp. 1–13, 2011.



(a) Point cloud



(b) Clustered planes

Fig. 3. Clustering of a building using Incremental RANSAC PCA (courtesy of ETH Zurich, which provided the flow of 3D points).

[5] U. Bauer and K. Polthier, “Detection of planar regions in volume data for topology optimization,” *Advances in Geometric Modeling and Processing*, pp. 119–126, 2008.

[6] R. Lakaemper and L. J. Latecki, “Extended EM for planar approximation of 3D data,” in *IEEE International Conference on Robotics and Automation*, 2006.

[7] O. Wulf, K. O. Arras, H. I. Christensen, and B. Wagner, “2d mapping of cluttered indoor environments by means of 3D perception,” in *2004 IEEE International Conference on Robotics and Automation*, vol. 4, pp. 4204–4209, 2004.

[8] H. Surmann, K. Lingemann, A. Nchter, and J. Hertzberg, “A 3d laser range finder for autonomous mobile robots,” in *International Symposium on Robotics*, 2001.

[9] E. Brunskill and N. Roy, “SLAM using incremental probabilistic PCA and dimensionality reduction,” in *2005 IEEE International Conference on Robotics and Automation*, pp. 342–347, 2005.

[10] P. S. Bradley and O. L. Mangasarian, “k-plane clustering,” *Journal of Global Optimization*, vol. 16, pp. 23–32, 2000.

[11] P. Tseng, “Nearest q-flat to m points,” *Journal of Optimization Theory and Applications*, vol. 105, pp. 249–252, 2000.

[12] D. Viejo and M. Cazorla, “Geometric primitive extraction for 3D model reconstruction,” *Recent Advances In Artificial Intelligence Research And Development*, p. 267, 2004.

[13] M. Attene, B. Falcidieno, and M. Spagnuolo, “Hierarchical mesh segmentation based on fitting primitives,” *The Visual Computer*, vol. 22, pp. 181–193, 2006.

[14] J. Costeira and T. Kanade, “A multi-body factorization method for independently moving objects,” *International Journal of Computer Vision*, vol. 29, pp. 159–179, 1997.

[15] E. Elhamifar and R. Vidal, “Sparse subspace clustering,” in *International Conference on Computer Vision and Pattern Recognition*, pp. 2790–2797, 2009.

[16] R. E. Vidal, Y. Ma, and S. Sastry, “Generalized principal component analysis (gpca),” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1945–1959, 2005.

[17] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, pp. 381–395, 1981.

[18] R. Schnabel, R. Wahl, and R. Klein, “Efficient RANSAC for Point-Cloud Shape Detection,” *Computer Graphics Forum*, vol. 26, pp. 214–226, 2007.

[19] J. Elseberg, D. Borrmann, and A. Nüchter, “Efficient processing of large 3d point clouds,” in *International Symposium on Information, Communication and Automation Technologies*, 2011.

[20] M. Artač, M. Jogan, and A. Leonardis, “Mobile robot localization using an incremental eigenspace model,” in *IEEE Conference of Robotics and Automation*, pp. 1025–1030, 2002.

[21] R. E. Vidal, “Subspace clustering,” *IEEE Signal Processing Magazine*, 2011.

[22] I. T. Jolliffe, *Principal Component Analysis*. Springer, 2th ed., 2002.

[23] J. Weng, Y. Zhang, and W. S. Hwang, “Candid covariance-free incremental principal component analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1034–1040, 2003.

[24] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, “On the shape of a set of points in the plane,” *IEEE Transactions on Information Theory*, vol. 29, pp. 551–559, 1983.

[25] R. Voigt, J. Nikolic, C. Hürzeler, S. Weiss, L. Kneip, and R. Siegwart, “Robust embedded egomotion estimation,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2694–2699, 2011.

[26] S. P. Lloyd, “Least squares quantization in pcm,” *IEEE Transactions on Information Theory*, vol. 28, pp. 129–136, 1982.