Fast Sampling Plane Filtering, Polygon Construction and Merging from Depth Images

Joydeep Biswas
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
joydeepb@ri.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
mmv@cs.cmu.edu

Abstract*—Recently, the availability of low cost depth cameras has provided 3D sensing capabilities for mobile robots in the form of dense 3D point clouds, usable for applications like 3D mapping and reconstruction, shape analysis, pose tracking and object recognition. For all the aforementioned applications, processing the raw 3D point cloud in real time and at full frame rates may be infeasible due to the sheer volume of data. Hence, a natural choice is to extract geometric features from the point cloud, and process these features instead of the raw point clouds. The task of geometric feature extraction itself is challenging due to noisy sensing, geometric outliers and real-time constraints. We introduce the Fast Sampling Plane Filtering (FSPF) algorithm to reduce the volume of the 3D point cloud by sampling points from the depth image, and classifying local grouped sets of points using Random Sample Consensus (RANSAC) as belonging to planes in 3D (called the "plane filtered" points) or points that do not correspond to planes (the "outlier" points). The plane filtered points are then converted to a set of convex polygons in 3D which represent the planes detected in the scene. The detected convex polygons are then merged across successive depth images to generate a set of scene polygons. The FSPF and polygon merging algorithms run in real time at full camera frame rates with low CPU requirements: In a real world indoor environment scene, FSPF takes on average 1.39 ms to process a single 640 $\times$ 480 depth image, producing 2004 plane filtered 3D points and 70 polygons(average). The polygon merging algorithm takes 1.12 ms to merge the latest frame with the history of obervations. Thus, the algorithms are capable of processing depth images at about 400 frames per second on average. We provide experimental results demonstrating the computational efficiency of the FSPF algorithm, and include results from different indoor scenes.

I. Introduction

With the advent of low-cost depth imaging cameras, recent work has focused on exploiting this sensor modality for the purpose of mobile robot autonomy. These depth cameras made dense 3D point clouds available, which was previously only possible with much more expensive sensors like time-of-flight cameras or scanning 3D laser rangefinders. Applications like 3D mapping and reconstruction, shape analysis, pose tracking and object recognition can potentially benefit from this sensor modality. However, given that indoor mobile robots have limited onboard computational power it is infeasible to process the complete 3D point clouds in real time and at full frame rates (*e.g.* the Microsoft Kinect sensor produces 9.2M 3D pts/sec). Feature extraction, and in particular, geometric feature extraction is therefore the natural choice for abstracting the sensor data. However, noisy sensing and the presence

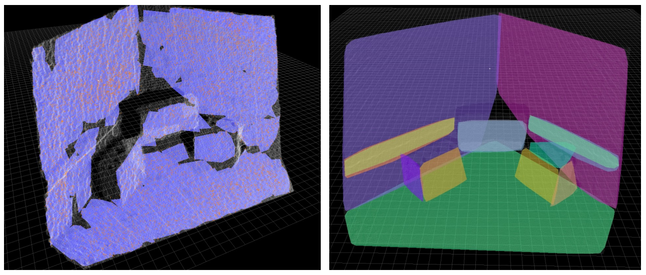

(a) (b)

Fig. 1: Results from a test scene: (a) Plane Filtered points shown as orange points, corresponding convex polygons shown in blue. The complete 3D point cloud is overlaid as translucent grey for reference. (b) Scene polygon set generated by merging polygons from 15 consecutive depth image frames

of geometric outliers (objects amidst the geometric features that do not match the geometric model of the features) provide additional challenges to the task of geometric feature extraction.

We introduce the Fast Sampling Plane Filtering (FSPF) algorithm that samples the depth image to produce a set of "plane filtered" points corresponding to planes, the corresponding plane parameters (normals and offsets), and the convex polygons in 3D to fit these plane filtered points. The FSPF algorithm meets the following goals:

1) Reduce the volume of the 3D point cloud by generating a smaller set of "plane filtered" 3D points
2) Compute convex polygons to fit the plane filtered points
3) Iteratively merge convex plane polygons without maintaining a history of all observed plane filtered points
4) Perform all of the above in real time and at full frame rates

We provide experimental results to show the effectiveness of plane estimation and the computational efficiency of FSPF. To illustrate the key processed results, Fig. 1 shows the plane filtered points and polygons from a single frame, and the scene polygon set obtained by merging 15 depth image frames from a sample scene.

This paper is organized as follows: Section II discusses relevant related work. Section III introduces the Fast Sampling Plane Filtering algorithm that samples the depth image to produce a set of points corresponding to planes, and the convex plane polygons to best fit the points. Section IV introduces the polygon merging procedure which iteratively merges convex plane polygons at every time step based on the complete history of detected plane filtered points, without explicitly mainting such history. The paper concludes with Experimental results (Section V) showing the effectiveness and speed of execution of the algorithms.

II. Related Work

Approaches that operate on raw 3D point clouds for plane (and general geometric shape) detection (*e.g.* [5, 9]) are ill-suited to running in real-time due to their high computational requirements, and because they ignore the fact that depth cameras make observations in "2.5D": the depth values are observed on a (virtual) 2D image plane originating from a single point. Region growing [8] exploits the local correlation in the depth image and attempts to assign planes to every 3D point. The Fast Sampling Plane Filtering algorithm, which we introduce in this paper, on the other hand samples points at random and does not attempt to fit planes to every point, and instead uses local RANSAC [1].

A number of 3D mapping algorithms ([6, 2, 10]) have been developed that build maps using 3D points in space, but these methods do not reason about the geometric primitives that the 3D points approximate.

An alternative approach to mapping using the raw 3D points is to map using planar features extracted from the 3D point cloud [12, 4]. Approaches to extraction of geometric features in point clouds explored in the past include 3D Hough transform [11] and region growing [8]. In particular, 3D Plane SLAM [7] is a 6D SLAM algorithm that uses observed 3D point clouds to construct maps with 3D planes. The plane detection in their work relies on region growing for plane extraction, whereas our approach uses local RANSAC filtering of the depth image.

III. Fast Sampling Plane Filtering

Depth cameras provide, for every pixel, in addition to the usual color values, their distance from the camera. This depth information, along with the camera intrinsics (horizontal field of view f_h, vertical field of view f_v, image width w and height h in pixels) can be used to reconstruct a 3D point cloud. Let the depth image of size $w \times h$ pixels provided by the camera be I, where $I(i,j)$ is the depth of a pixel at location $d = (i,j)$. The corresponding 3D point $p = (p_x, p_y, p_z)$ is reconstructed using the depth value $I(d)$ as

$$p_x = I(d)\left(\frac{j}{w-1} - 0.5\right)\tan\left(\frac{f_h}{2}\right), \tag{1}$$

$$p_y = I(d)\left(\frac{i}{h-1} - 0.5\right)\tan\left(\frac{f_v}{2}\right), \tag{2}$$

$$p_z = I(d). \tag{3}$$

The naïve approach to reducing the volume of data would be to sub-sample the 3D point cloud. Although this reduces the number of 3D points processed by the algorithms, it discards information about the scene. An alternative is to convert the 3D point cloud into a more compact, feature-based representation, like planes in 3D. However, computing optimal planes to fit the point cloud for every observed 3D point would be extremely CPU-intensive and sensitive to occlusions by obstacles. The Fast Sampling Plane Filtering (FSPF) algorithm combines both ideas: it samples random neighborhoods in the depth image, and in each neighborhood, it performs a RANSAC based plane fitting on the 3D points. Each neighborhood of plane filtered points is then combined to form a convex polygon. Thus, FSPF reduces the volume of the 3D point cloud and extracts geometric features in the form of convex polygons in 3D while being robust to outliers.

FSPF takes the depth image I as its input, and creates a list P of n 3D points, a list R of corresponding plane normals, a list C of convex polygons, and a list O of outlier points that do not correspond to the locally evaluated planes. Algorithm 1 outlines the plane filtering procedure. The configuration parameters required by FSPF are listed in Table I.

FSPF proceeds by first sampling three locations d_0,d_1,d_2 from the depth image (lines 6-8). The first location d_0 is selected randomly from anywhere in the image, and d_1 and d_2 are selected randomly within a neighborhood of size η around d_0. The 3D coordinates for the corresponding points p_0, p_1, p_2 are then computed using eq. 1-3. A search window of width w' and height h' is computed based on the mean depth (z-coordinate) of the points p_0, p_1, p_2 (lines 11-13), and the minimum expected size S of the planes in the world. An additional $l-3$ local samples d_j are then sampled from the search window. The plane fit error for the 3D point p_j from the plane defined by the points p_1, p_2, p_3 is computed (line 21) to determine if it as an "inlier." If more than $\alpha_{in} l$ points in the search window are classified as inliers, then all the inlier points $\hat{P}$ are added to the list P, and the associated normals to the list R. A convex polygon $\hat{c}$ is constructed (line 31) from $\hat{P}$, and added to the list of convex polygons, C. The procedure for constructing the convex polygon $\hat{c}$ from 3D points $\hat{P}$ is discussed in Section IV. This algorithm is run a maximum of m_{max} times to generate a list of at most n_{max} 3D points and their corresponding plane normals.

IV. Plane Polygon Construction And Merging

A convex polygon is denoted by the tuple $c = \{\hat{P}, n, \bar{p}, r, b_1, b_2, B\}$ where $\hat{P}$ is the set of 3D points used to construct the convex polygon, n the number of points in

Parameter	Description
n_{max}	Maximum total number of 3D points
k_{max}	Maximum number of neighborhoods to sample
l	Number of local samples
η	Neighborhood for global samples (in pixels)
S	Plane size in world space for local samples
ϵ	Maximum plane offset error for inliers
α_{in}	Minimum fraction of inliers to accept local sample

TABLE I: Configuration parameters for plane filtering algorithm

Algorithm 1 Plane Filtering Algorithm

```
1:  procedure PLANEFILTERING($I$)
2:      $P, R, C, O \leftarrow \{\}$
3:      $n, k \leftarrow 0$
4:      while $n < n_{max} \wedge k < k_{max}$ do
5:          $k \leftarrow k + 1$
6:          $d_0 \leftarrow (\text{rand}(0, h-1), \text{rand}(0, w-1))$
7:          $d_1 \leftarrow d_0 + (\text{rand}(-\eta, \eta), \text{rand}(-\eta, \eta))$
8:          $d_2 \leftarrow d_0 + (\text{rand}(-\eta, \eta), \text{rand}(-\eta, \eta))$
9:          Reconstruct $p_0$, $p_1$, $p_2$ from $d_0$,$d_1$,$d_2$        ▷ eq. 1-3
10:         $r = \frac{(p_1-p_0)\times(p_2-p_0)}{||(p_1-p_0)\times(p_2-p_0)||}$        ▷ Compute plane normal
11:         $\bar{z} = \frac{p_{0z}+p_{1z}+p_{2z}}{3}$
12:         $w' = w\frac{S}{\bar{z}}\tan(f_h)$
13:         $h' = h\frac{S}{\bar{z}}\tan(f_v)$
14:         numInliers $\leftarrow 0$
15:         $\hat{P} \leftarrow \{\}$
16:         $\hat{R} \leftarrow \{\}$
17:         $\hat{c} \leftarrow \{\}$
18:         for $j \leftarrow 3, l$ do
19:             $d_j \leftarrow d_0 + (\text{rand}(-\frac{h'}{2}, \frac{h'}{2}), \text{rand}(-\frac{w'}{2}, \frac{w'}{2}))$
20:             Reconstruct 3D point $p_j$ from $d_j$        ▷ eq. 1-3
21:             $e = \text{abs}\left[r \cdot (p - p_0)\right]$
22:             if $e < \epsilon$ then
23:                 Add $p_j$ to $\hat{P}$
24:                 Add $r$ to $\hat{R}$
25:                 numInliers $\leftarrow$ numInliers + 1
26:             end if
27:         end for
28:         if numInliers $> \alpha_{in} l$ then
29:             Add $\hat{P}$ to $P$
30:             Add $\hat{R}$ to $R$
31:             Construct convex polygon $\hat{c}$ from $\hat{P}$
32:             Add $\hat{c}$ to $C$
33:             numPoints $\leftarrow$ numPoints + numInliers
34:         else
35:             Add $\hat{P}$ to $O$
36:         end if
37:     end while
38:     return $P, R, C, O$
39: end procedure
```

$\hat{P}$, $\bar{p}$ the centroid of the polygon, r the normal to the polygon plane, b_1 and b_2 the 2D basis vectors on the plane of the polygon and B the set of 3D points which define the convex boundary of the polygon. Given a set of 3D points $\hat{P}$, a convex polygon c is constructed from $\hat{P}$ as follows:

1) The polygon centroid $\bar{p}$ is computed as $\bar{p} = \frac{1}{n}\sum_{p_i \in \hat{P}} p_i$.
2) The Scatter Matrix S of the points in $\hat{P}$ is computed as $S = \sum_{p_i \in \hat{P}} (p_i - \bar{p})(p_i - \bar{p})^T$.
3) The normal r is then given by the eigenvector of S corresponding to its smallest eigenvalue.
4) The basis vectors b_1 and b_2 are the remaining two eigenvectors of S.
5) The boundary set B is found using Graham scan [3] over the points of $\hat{P}$ projected onto the plane.

In order to construct a meaningful geometric representation of a scene, it is neccesary to merge the observed polygons from multiple scenes to form a single, "global scene". In general, this is done in three steps:

1) Pose Registration: The pose of the camera is extracted relative to the global coordinates of the scene.
2) Polygon Correspondence Matching: Polygons in the latest frame are matched to polygons in the global scene.
3) Polygon Merging: Matched polygons are merged to update the plane parameters and the convex hull of the polygons in the scene.

In this Section, we discuss methods for performing the third step, polygon merging.

The problem of polygon merging is thus: given two convex polygons c^1, c^2, we wish to construct a merged polygon c^m using the 3D points from both planes c^1 and c^2.

The naïve approach to merging polygons $c^1 = \{\hat{P}^1, n^1, \bar{p}^1, r^1, b_1^1, b_2^1, B^1\}$ and $c^2 = \{\hat{P}^2, n^2, \bar{p}^2, r^2, b_2^2, b_2^2, B^2\}$ to form merged polygon $c^m = \{\hat{P}^m, n^m, \bar{p}^m, r^m, b_m^m, b_2^m, B^m\}$ is to combine the 3D points as $\hat{P}^m = \hat{P}^1 \bigcup \hat{P}^2$ and then compute all other polygon parameters from $\hat{P}^m$. In practice, this is infeasible since it requires a complete history of all 3D points ever observed to be maintained. It is, however, possible to perform polygon merging without maintaining a complete history of all observed 3D points.

The scatter matrix S^1 for polygon 1, constructed from points $p_i^1 = \{x_i^1, y_i^1, z_i^1\}, p_i^1 \in \hat{P}^1, i \in [1, n^1]$ has the form

$$S^1 = \begin{bmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{33} \end{bmatrix}, \tag{4}$$

where

$$S_{11}^1 = \sum(x_i^1 - \bar{x}^1)(x_i^1 - \bar{x}^1), \tag{5}$$
$$S_{12}^1 = S_{21}^1 = \sum(x_i^1 - \bar{x}^1)(y_i^1 - \bar{y}^1), \tag{6}$$
$$S_{13}^1 = S_{31}^1 = \sum(x_i^1 - \bar{x}^1)(z_i^1 - \bar{z}^1), \tag{7}$$
$$S_{22}^1 = \sum(y_i^1 - \bar{y}^1)(y_i^1 - \bar{y}^1), \tag{8}$$
$$S_{23}^1 = S_{32}^1 = \sum(y_i^1 - \bar{y}^1)(z_i^1 - \bar{z}^1), \tag{9}$$
$$S_{33}^1 = \sum(z_i^1 - \bar{z}^1)(z_i^1 - \bar{z}^1). \tag{10}$$

Scatter matrices from two different polygons cannot be directly combined due to the coupling in S between the points p_i and the centroid. However, S^1 may be decoupled as $S^1 = S_1^1 + n^1 S_2^1$ where S_1^1 depends only on the 3D points in $\hat{P}^1$, and S_2^1 on the centroid $\bar{p}^1$. S_1^1 and S_2^1 are then given by,

$$S_1^1 = \begin{bmatrix} \sum x_i^1 x_i^1 & \sum x_i^1 y_i^1 & \sum x_i^1 z_i^1 \\ \sum y_i^1 x_i^1 & \sum y_i^1 y_i^1 & \sum y_i^1 z_i^1 \\ \sum z_i^1 x_i^1 & \sum z_i^1 y_i^1 & \sum z_i^1 z_i^1 \end{bmatrix}, \tag{11}$$

$$S_2^1 = \begin{bmatrix} \bar{x}^1\bar{x}^1 & \bar{x}^1\bar{y}^1 & \bar{x}^1\bar{z}^1 \\ \bar{y}^1\bar{x}^1 & \bar{y}^1\bar{y}^1 & \bar{y}^1\bar{z}^1 \\ \bar{z}^1\bar{x}^1 & \bar{z}^1\bar{y}^1 & \bar{z}^1\bar{z}^1 \end{bmatrix}. \tag{12}$$

Given the centroids $\bar{p}^1$ and $\bar{p}^2$ of the two polygons, the centroid $\bar{p}^m$ of the merged polygon is computed as

$$\bar{p}^m = \frac{1}{n^m}(n^1\bar{p}^1 + n^2\bar{p}^2), \quad (13)$$

$$n^m = n^1 + n^2. \quad (14)$$

Thus, the matrix S_2^m can be computed from $\bar{p}^m$, and the scatter matrix of the combined polygon is given by,

$$S^m = S_1^1 + S_1^2 + n^m S_2^m. \quad (15)$$

Therefore, the matrix S^1 along with centroid $\hat{p}$ and the number of points n are sufficient for merging polygons based on the complete history of observed 3D points, and the individual 3D points no longer need to be maintained for each polygon. To ensure numerical stability over time, the eigenvector decomposition is performed on the normalized matrix $\frac{1}{n}S$, and normalized matrices $\frac{1}{n}S_1$ are maintained instead of S_1. The convex boundary point set B^m of the merged polygon is computed by running Graham scan on the union of the points from the boundary point sets B^1 and B^2 of the constituent polygons.

The complete set of steps to merge two polygons are thus,

1) Compute centroid $\bar{p}^m$ using Eq. 14
2) Compute scatter matrix S^m using Eq. 15
3) Compute normal r^m and basis vectors b_1^m, b_2^m by eigenvector decompostion of S^m
4) Compute boundary point set B^m using Graham scan on plane projected points from $B^1 \bigcup B^2$

V. Experimental Results

We performed two sets of experiments to demonstrate the computational efficiency and effectiveness of FSPF and the polygon merging algorithms. First, we evaluated the effect of varying the parameters of FSPF on its run time using logged data from the Kinect Sensor mounted on our mobile robot while driving around in an indoor environment. Next, we tested the FSPF and polygon merging algorithms on a set of real indoor scenes.

A. Computational Requirements

Depth image data was recorded for a 153 second long run of our mobile robot navigating down a series of hallways and open areas. This data was processed offline by varying the parameters of the FSPF algorithm to see their impact on computational requirements in terms of the mean CPU time required to generate the plane filtered points. This does not include the time taken for polygonization, since the polygonization time is approximately constant (dominated by the time taken for the eigenvector decomposition of the scatter matrix) for a fixed number of polygons. These tests were run on a single core of an Intel Core i7 950 processor @ 3.07GHz.

Fig. 2 shows the variation in the processing time for the FSPF algorithm for different values of n_{max}, η, f, and l.The processing time scales roughly linearly with the maximum number of 3D points, n_{max}. For smaller values of η, there are more rejected samples, corresponding to longer processing times. For larger values of η, plane estimates with larger offset errors are accepted, leading to shorter processing times. Larger values of f correspond to larger consensus sets, so the trend in processing times is the opposite to that for different values of η. The processing times decrease with larger values of l.

B. Real World Scene Tests

The FSPF and polygon merge algorithms were tested on data collected from 5 scenes. At each scene, the Kinect sensor was kept static, so no pose update was neccesary. For each scene, the polygons from 15 consecutive frames were merged to generate a set of polygons which represent the entire scene. Fig. 3 shows the scenes and the processed output from each scene, while Table II summarizes the output and processing times for the algorithms for each scene. The sampling efficiency for each scene is computed as the ratio of the mean number of plane filtered points per frame to the mean sampled locations per frame. The FSPF run time is per frame, and the polygon merge processing time is for all the polygons per frame.

Scene 1 has three boxes in the corner of a room. The merged scene polygons correctly include all the planes in the scene except two (the tops of the boxes on the left and center). Scene 2 is set in a corridor, and lacks clutter. The merged scene polygons correctly include all the polygons in the immediate vicinity of the Kinect. Scene 3 is set in an open area next to a staircase. The merged scene polygons include the ground polygon, the side of the staircase, and the overhanging staircase. Scenes 4 and 5 show work areas with chairs and tables. Despite the clutter, all table surfaces are correctly identified among the merged scene polygons. Scene 5 had a toy chair on top of the table, and the merged scene includes its corresponding polygons as well.

VI. Conclusion And Future Work

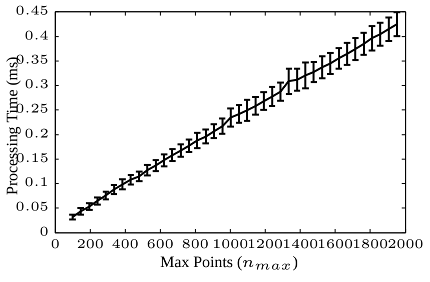
In this paper, we introduced the Fast Sampling Plane Filtering algorithm to filter depth images into plane filtered points and polygons, and subsequently the polygon merge algorithm which merged polygons based on the complete history of observations. We experimentally showed the computational efficiency of FSPF. We further evaluated the effectiveness of FSPF and the polygon merge algorithms by testing on a number of indoor scenes which included clutter.

Currently the polygon merging algorithm assumes that merged polygons are convex, but merging to complex polygons would be the natural next step. The camera pose registration and polygon matching algorithms have not been covered in this work, but are areas of interest to the authors.

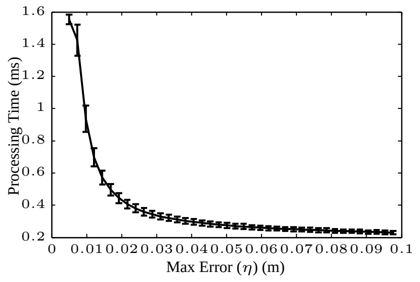
References

[1] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

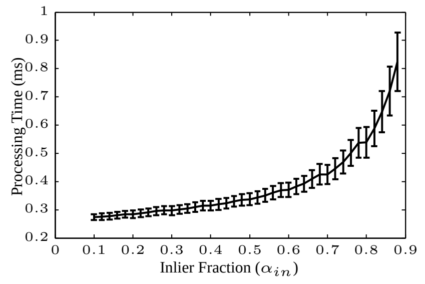
[2] M. Gary, N. Fairfield, W.C. Stone, D. Wettergreen, GA Kantor, and J.M. Sharp Jr. 3D mapping and characterization of sistema Zacatón from DEPTHX



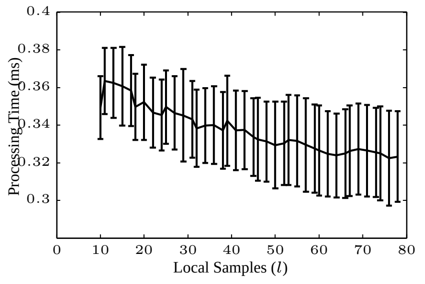
(a)



(b)



(c)



(d)

Fig. 2: Effect of varying the parameters of FSPF on its run time: (a) Max Points n_{max}, (b) Max Error η, (c) Inlier Fraction f, and (d) Local Samples l

Scene Scene	Merged Polygons	Plane Filtered Points Per Frame	Sampled Points Per Frame	Sampling Efficiency (%)	Polygons Per Frame	FSPF Run Time (ms)	Polygon Merge Time (ms)
1	14	2002	5434	36.84	71	1.86	0.37
2	5	2006	8620	23.28	81	1.66	0.71
3	3	2001	11009	18.18	82	2.06	0.38
4	14	2004	8260	24.27	70	1.39	1.12
5	7	2001	6861	19.17	67	1.66	0.95

TABLE II: Results from test scenes

(DEep Phreatic THermal eXplorer). In *Proceedings of KARST08: 11th Sinkhole Conference ASCE*, 2008.

[3] R. L. Graham. An efficient algorith for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132 – 133, 1972.

[4] P. Kohlhepp, P. Pozzo, M. Walther, and R. Dillmann. Sequential 3D-SLAM for mobile action planning. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 722–729. IEEE, 2004.

[5] N.J. Mitra and A. Nguyen. Estimating surface normals in noisy point cloud data. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 322–328. ACM, 2003.

[6] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *ICAR'05*, pages 242–249. IEEE, 2005.

[7] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga. Online three-dimensional SLAM by registration of large planar surface segments and closed-form pose-graph relaxation. *Journal of Field Robotics*, 27(1):52–84, 2010.

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

[9] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for Point-Cloud Shape Detection. In *Computer Graphics Forum*, volume 26. Wiley Online Library, 2007.

[10] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 321–328. IEEE, 2000.

[11] G. Vosselman, B.G.H. Gorte, G. Sithole, and T. Rabbani. Recognising structure in laser scanner point clouds. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 46(8):33–38, 2004.

[12] J. Weingarten and R. Siegwart. 3D SLAM using planar segments. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3062–3067. IEEE, 2006.

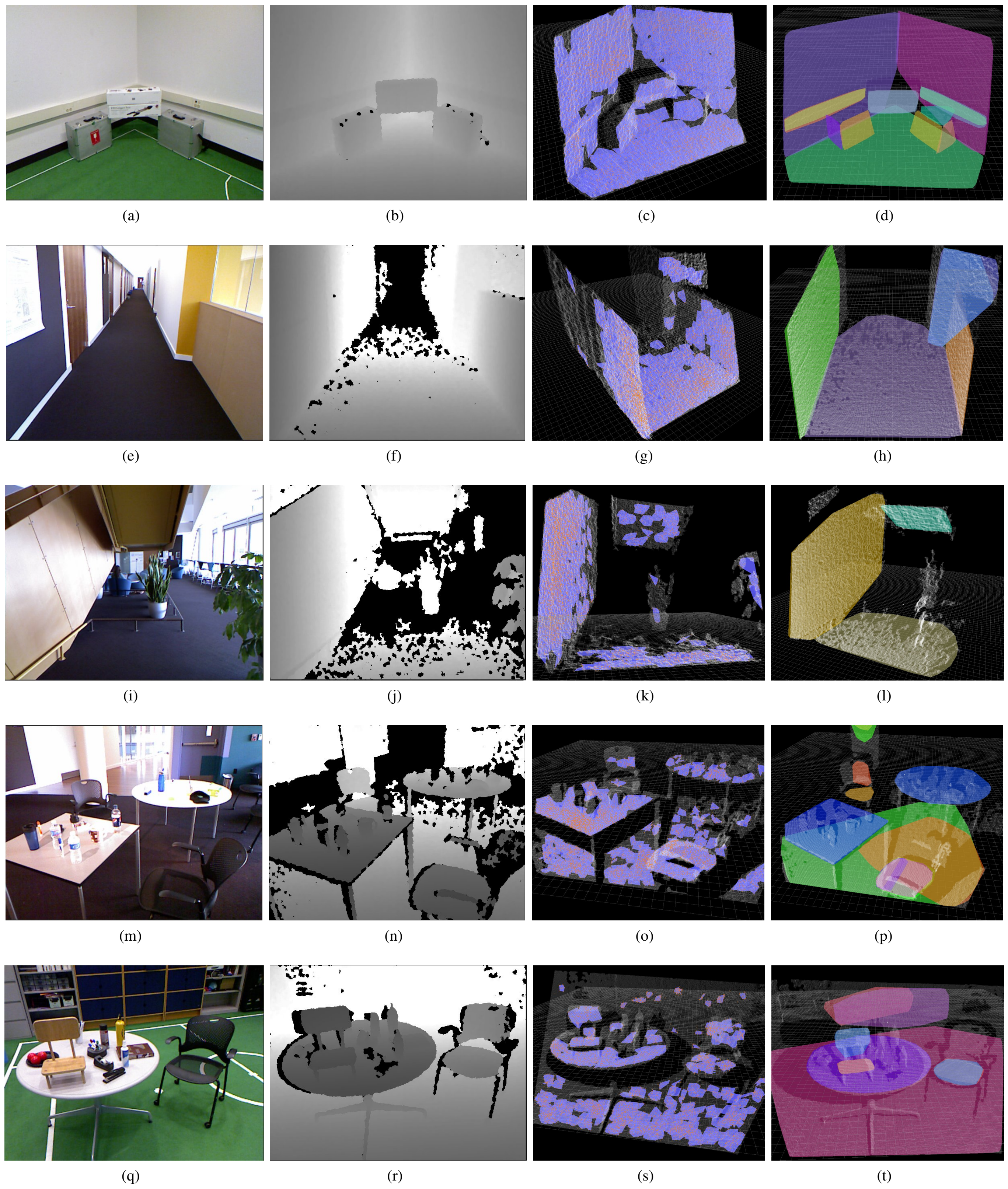


Fig. 3: Test scenes 1-5 (top to bottom). First column shows the RGB image of the scene, second column the depth image, third column the plane filtered points (orange points) and polygons (lilac) for a single depth image frame, fourth column the polygons generated by merging 15 consecutive processed depth image frames. Columns 2 and 3 include overlays of the raw 3D point cloud (translucent white) for reference.