Proceedings of the OAGM&ARW Joint Workshop 2017 DOI: 10.3217/978-3-85125-524-9-36

Bilateral Filters for quick 2.5 D Plane Segmentation

Simon Schreiberhuber[1], Thomas Mörwald[2] and Markus Vincze[1]

Abstract— **We present a simple and practicable approach to segment organized point clouds gathered with RGBD sensors into planar elements. The algorithm proves to execute extremely fast while delivering all the dominant planes of a scene. As an integral part of our segmentation algorithm we examined two off the shelf and one heavily modified filtering algorithms to increase the quality of the point cloud before the actual segmentation process. The results of two of these algorithms were promising. One provides a favorable tradeoff between speed and quality while the other delivers superior quality at high computational cost.**

I. Introduction

In mobile robotics many tasks have to be fulfilled in indoor environments. More specifically one task could e.g. include the search or classification of objects lying on the floor. Instead of processing all the points captured by the RGBD sensor it would be beneficial to early on discard some of the points that can not be part of the task. Removing the dominant planes from the scene is one common measure to achieve this. This becomes obvious when we observe that indoor environments are dominated by planar surfaces.

While other plane segmentation algorithms operate on unfiltered depth data, our algorithm utilizes a filtering step. Data as it is captured by an RGBD sensor tends to have multiple sources of noise, all of which tend to make the fitting of planes difficult. Reducing the noise upfront therefore is a prerequisite to a fast and simple plane segmentation approach.

To create ideal conditions for our plane segmentation algorithm we discuss three filter approaches. With these filters we aim to refine planar regions while keeping the geometric details where they are needed. We show the results generated by the standard Bilateral Filter [7], the Sigma Adaptive Bilateral Filter [2] and the adapted Bilateral Mesh Denoising algorithm [4]. A discussion shows how these filters relate to each other and how they behave in specific situations. We describe the modifications necessary to apply the Bilateral Mesh Denoising algorithm to depth data and demonstrate its effectiveness.

Regarding the core of our plane segmentation, we offer a comparison to two other algorithms: The comparably slow approach shown by Holz [5] which uses RANSAC to refine a rough normal based plane segmentation and an approach shown by Wang [8] where a rough segmentation is improved on a point-wise basis. Both algorithms start with clustering the points into a 3 D voxel grid. By doing this they are replacing the inherent neighborhood information with a costly spacial relation. Finding the nearest neighbors to a specific point no longer is a simple access to the neighboring depth pixels but a search of all points in the adjacent voxel blocks. For our segmentation we follow a similar two-step approach as in [8] but make use of the neighborhood information contained in the organized point cloud.

[1]Simon Schreiberhuber and Markus Vincze are with the Vision4Robotics group (ACIN - TU Wien), Austria {schreiberhuber, vincze}@acin.tuwien.ac.at

[2]Thomas Mörwald was a member of the Vision4Robotics group.

This work is supported by the European Comission through the Horizon 2020 Programme (H2020-ICT-2014-1, Grant agreement no: 645376), FLOBOT.

II. Related Work

Most plane segmentation approaches can be assigned to two categories. A direct approach, where planes are directly matched with the existing points, and indirect approaches where the scene is transformed into another representation.

RANSAC [3] is a direct approach that iteratively tests randomly generated plane hypothesis against a point cloud and is often used to find the ground plane of a scene. To extract multiple planes from a scene RANSAC has to be used repeatedly to assign points to different planes. The outcome of this approach is highly dependent on the order in which the RANSAC algorithm finds the planes. Thus the affiliation of points to planes is ambiguous.

The approach shown in [5] therefore does not use RANSAC for the segmentation itself but uses it to refine already existing plane hypothesis. These hypothesis are generated by clustering normal vectors in normal space or spherical coordinates. This delivers clusters of points, each of which is assembled by multiple planes facing the same direction. Averaging the normals within each of these clusters leads to a plane hypothesis which allows to separate the points into their according planes. Calculating the distance of the points to these plane hypothesis directly allows to cluster these points into their according planes.

A more direct approach was chosen in [8] is based on roughly clustering plane patches within a 3 D voxel grid. Some of these blocks within the voxel grid are containing enough points to approximate planes. In the following step it is possible to connect neighboring grid blocks to bigger surfaces wherever these planes are facing in roughly the same direction. The approach chosen by Zhang [10] is to find lines along the horizontal scan-lines which are cuts trough planes. In a second step the normals get estimated along these line segments to find corresponding segments between scan-lines. Fitting line segments can then be connected to a planar region.

The V-disparity algorithm [11] transforms the 3 D data into a V-disparity map and therefore reduces the 3 D plane fit to

a 2 D line fit which greatly reduces the computational effort. While not being as straight forward as any of the presented direct algorithms it is capable of finding planes where noise is dominant or in rough outdoor environments [9].
To improve results of certain plane segmentation algorithms it is vital to reduce noise of the input data by proper filtering. While a Gaussian Blur might be sufficient to remove noise from some intensity images it is not fit to be applied to depth maps. Besides not being able to handle areas where the sensor was unable to capture data this filter would destroy any information on discontinuities. Bilateral filtering [7] therefore is more selective and reduces over-smoothing along discontinuities. It is therefore a possible candidate for point clouds but has serious issues regarding our task since this filter introduces a bending along edges of tilted planes. This so called ski effect can be tackled by restraining the filter from working on edges [2], or by applying a bilateral filter, that is specifically designed for 3D geometry [4].
Approaches as the Total Variation [6] and the Total Generalized Variation [1] based algorithms do not inherit their principle from convolution. Instead they minimize a cost function to fulfill a tradeoff of being close to the input and minimizing a smoothness measure. The Total Variation image denoising algorithms work well for intensity images but do have downsides as a tendency to frontoparalell planes when applied to depth-maps. This tendency in particular can be countered by using the Total Generalized Variation algorithm which allows for more refined regularization with higher order derivatives but at the cost of increased computational complexity.

III. Filter

The quality of depth images obtained from the Kinect is moderate, especially in distances bigger than 3 meter (see Figure 2). To reduce the noise and other artifacts like quantization, the raw data has to be filtered.

A. Bilateral Filter

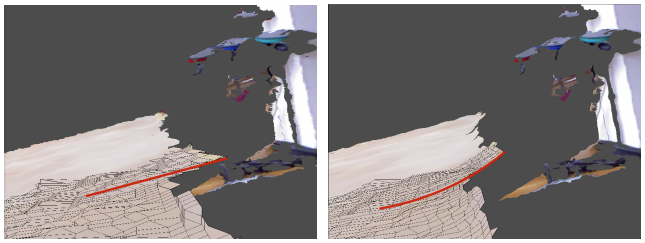
The bilateral filter is a suitable candidate. It preserves discontinuities and smooths out noise.

$$D_p^* = \frac{1}{W_p} \sum_{q \in S_p} G_{\sigma_s}(\|p-q\|) G_{\sigma_c}(|D_p - D_q|) D_q \tag{1}$$

$$W_p = \sum_{q \in S_p} G_{\sigma_s}(\|p-q\|) G_{\sigma_c}(|D_p - D_q|) \tag{2}$$

p: the coordinate of the resulting pixel.
q: the coordinate of a surrounding pixel.
G_σ: Gauss function.
$D_{p,q}$: depth values of p or q.
S_p: the neighborhood of p where $|p-q| < r_{Sth}$.
W_p: a normalization term.
σ_s: standard deviation for difference in depth.
σ_c: standard deviation for pixel distance.

This filter unfortunately introduces the unpleasant ski effect as shown in Figure 1.



(a) Unfiltered kinect image of a desk.

(b) After filtering the edge of the plane is bent upwards.

Fig. 1: The ski effect (red) is introduced by Bilateral Filtering.

B. Sigma Adaptive Bilateral Filter

Andreas Deutschmann [2] introduced the Sigma Adaptive Bilateral Filter which got rid of the ski effect and is containing edges, by reducing sigma around corners and edges.

$$D_p^* = \frac{1}{W_p} \sum_{q \in S_p} G_{\sigma_{s,p}}(\|p-q\|) G_{\sigma_{c,p}}(|D_p - D_q|) D_q \tag{3}$$

$$W_p = \sum_{q \in S_p} G_{\sigma_{s,p}}(\|p-q\|) G_{\sigma_{c,p}}(|D_p - D_q|) \tag{4}$$

Where

$$\sigma_{s,c,p} = \sigma_{s,c,max} + m_{sat,p} * (\sigma_{s,c,min} - \sigma_{s,c,max}) \tag{5}$$

is depending on the depth-maps curvature

$$m_{sat,p} = \begin{cases} 1 & \text{if } m > (1-k_{th}) \\ 0 & \text{if } m < k_{th} \\ m & \text{else} \end{cases} . \tag{6}$$

With

$$m_p = \frac{\tilde{m}_p - \tilde{m}_{min}}{\tilde{m}_{max} - \tilde{m}_{min}} \tag{7}$$

and

$$\tilde{m}_p = \left\| \frac{1}{|R_p|} \sum_{q \in_p R} (P_p - P_q) \right\| . \tag{8}$$

The terms are described the following:

$\tilde{m}$: is the raw curvature.
m: is the normalized curvature of the surface see Figure 5a.
m_{sat}: is a curvature that is saturated by k_{th} and $(1-k_{th})$.
$\sigma_{s,p}$: standard deviation of the Gauss filter. Weighing depending on the difference in depth.
$\sigma_{c,p}$: standard deviation of the Gauss filter. Weighing depending on the pixel distance.
$P_{p,q}$: the point at p or q, given as vector $P_{p,q} = \begin{bmatrix} x_p & y_p & z_p \end{bmatrix}^T$
R_p: like S a neighborhood around p where $\|P_p - P_q\| < r_{Rth}$.

This filter essentially is a Bilateral Filter which is suppressed in critical regions like edges (see Figure 3).

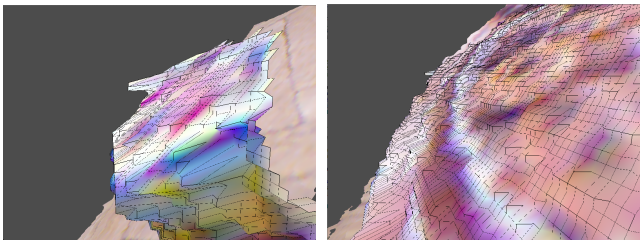
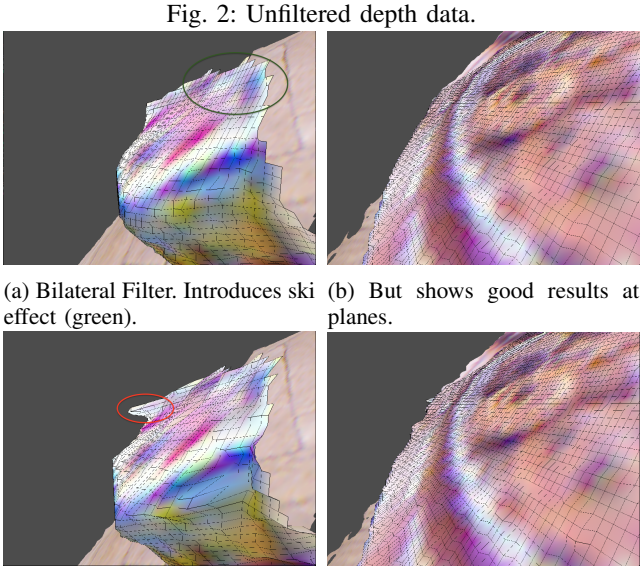


Fig. 2: Unfiltered depth data.



(a) Bilateral Filter. Introduces ski effect (green).

(b) But shows good results at planes.

(c) Sigma Adaptive Filter. Preserves details like edges, but can't filter noise at discontinuities (red).

(d) Delivers good results when applied to planar regions.

Fig. 3: Filtering results of Bilateral Filter and Sigma Adaptive Filter.

C. Bilateral Tangential Filter

The promoted filter is based on the Bilateral Mesh Denoising algorithm [4] which is used for meshes but not raw depth data. The idea behind this filter is to correct each point along its normal by a value composed by the deviation of surrounding points to its tangent plane. We adapt this principle to depth data by not correcting the points along their normal as in [4] but along the camera view rays. The filter is written as

$$C_p = \frac{1}{W_p} \sum_{q \in S_p} G_{\sigma_s}(\|p-q\|) G_{\sigma_c}(d_{p,q}) d_{p,q} \tag{9}$$

$$W_p = \sum_{q \in S_p} G_{\sigma_s}(\|p-q\|) G_{\sigma_c}(d_{p,q}) \tag{10}$$

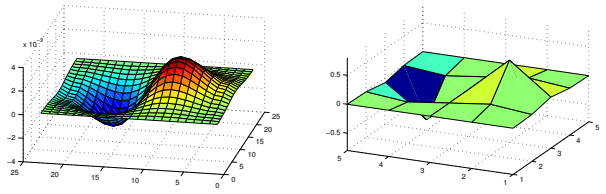
where the correction term $C_{p,k}$ is used to correct the depth values

$$D_p^* = D_p + C_p. \tag{11}$$

$d_{p,q}$ is the distance of the point q to the tangent plane of p

$$d_{p,q} = n_p \cdot (P_p - P_q). \tag{12}$$

It is not implied in this equation, but this filter is meant to be used iteratively.



(a) $K_{q,big}$.

(b) $K_{q,small}$.

Fig. 4: Filtering kernels, to calculate horizontal and vertical derivation of x, y and z. Sizes for these kernels are 23x23 and 5x5.

The quality of this filter strongly depends on the normal vectors n_p which tend to be difficult to obtain, especially along discontinuities and in noisy data. Incorrect normal values can make the algorithm locally unstable. Figure 6 shows a good example for how normal vectors affect the result. The normal vector is calculated by the vertical and horizontal derivation of x, y and z coordinates by the image coordinates u and v.

$$n_p = \frac{\tilde{n}_p}{\|\tilde{n}_p\|} \qquad \tilde{n}_p = \begin{bmatrix} \frac{dx_p}{du} \\ 0 \\ \frac{dz_p}{du} \end{bmatrix} \times \begin{bmatrix} 0 \\ \frac{dy_p}{dv} \\ \frac{dz_p}{dv} \end{bmatrix} \tag{13}$$

To obtain the needed derivatives we can not rely on a Canny Edge detection like approach since this would lead to wrong normals along discontinuities. We therefore have to mix the Canny Edge detection with the idea of the Bilateral Filter. To reduce the impact of discontinuities on the normals, points which are further away from the center point contribute less or not at all. This is achieved by an other Gaussian term G_{σ_n}.

$$\frac{d(x,y,z)_p}{du,v} = \sum_{q \in S_p} K_{q,p} G_{\sigma_n}(D_p - D_q)((x,y,z)_p - (x,y,z)_q) \tag{14}$$

Since the depth data along edges of objects is often distorted, it is necessary to compensate for that by locally extending the kernel:

$$K_{q,p} = \begin{cases} K_{q,big} & \text{if } c_p > c_{th} \\ K_{q,small} & \text{else} \end{cases} \tag{15}$$

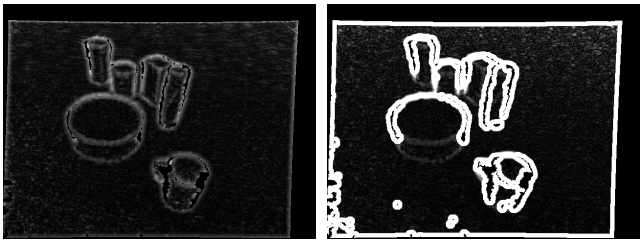
The filter kernels itself are shown in Figure 4. As basis to decide we are using a measure for how erratic the image is (Figure 5b).

$$c_p = \sum_{q \in R'_p} |D_p - D_q| \tag{16}$$

One example for proper filtering kernels are shown in Figure 4. Note that R'_p is in this case the neighborhood of p where $|p-q| < r_{R'th}$.

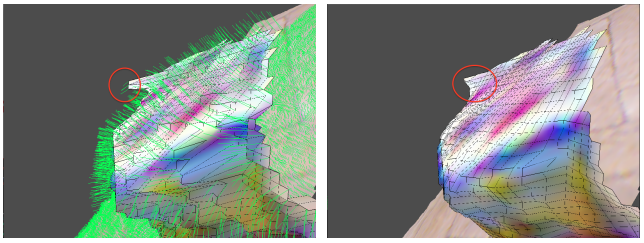
IV. Results of Filtering

The standard Bilateral Filter introduces the unpleasant ski effect [2] and therefore does not preserve information on edges (see Figure 3). On said edges the ski effect refers to a

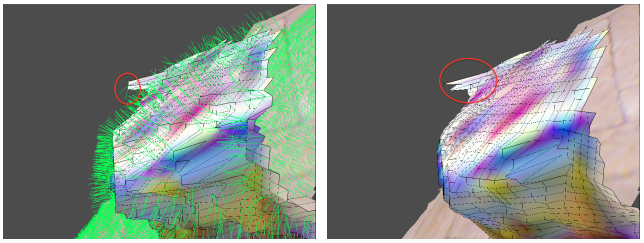


(a) m_p. (b) c_p

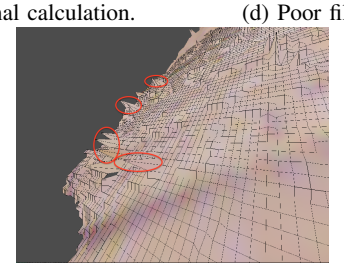
Fig. 5: Curvature m_p and unsteadyness c_p. These measures are used to guide σ in the adaptive filter and the normal estimation in the tangential filter.



(a) Correct normals. (b) Reasonable result.



(c) Poor normal calculation. (d) Poor filtering.



(e) Even along plane surfaces, poorly calculated normals are leading to artifacts or instabilities.

Fig. 6: Filtering results of the proposed Bilateral Tangential filter.

slight bending towards fronto-parallelity. Note that this effect gets stronger as the planes get tilted.
The Sigma Adaptive Bilateral Filter gets rid of this effect by not filtering in these critical regions. As seen in Figures 3c and 3d the results are comparable to the standard Bilateral Filter but without creating the ski defects. Spikes, as they often appear at sharp edges, will unfortunately not undergo any smoothing. Since we selected this filter to support our segmentation we created an GPU (AMD Radeon HD 6750M) implementation that computes within 25 ms.
For the Bilateral Mesh Denoising algorithm the results are different (see Figure 6). While being equally as suitable for the planar regions as the Bilateral and Sigma Adaptive Bilateral Filter, this algorithm shows the best results along discontinuities. In terms of computational complexity this algorithm unfortunately is way more demanding than the other two. This is mainly due to the complex normal estimation but also because it needs two to three iterations the other algorithms compute within one.

V. Segmentation

Two examples for state of the art algorithms coming close to a $30\,Hz$ segmentation rate are [5] and [8]. The algorithm shown in [5] utilizes a segmentation in normal space but is slower than the other. We therefore follow the approach shown in [8] where the points are split into a 3D voxel grid. A coarse pre-segmentationin on these then segments a majority of points with a relatively small amount of computations. Although the this approach is the faster one, it still is overly complicated for our needs. Seperating the organized pointcloud into equally sized cubes (voxels) only creates the need to compare these voxels to their 26 neighbouring voxels.

A. Hierarchical Plane Segmentation

The proposed algorithm follows the idea of pre-segmentation and splits the depth image into smaller patches similar to [8] but does it in image space. This reduces the number of neighbors for each patch to 8 and therefore saves computation time. The main steps of the algorithms are:

1) Patch generation: The depth data is grouped into equally sized section with sizes like e.g. $10 \times 10\,pixel$. It is then tried to fit a plane into these points. If there are enough points within a threshold of this plane the patch is retained and the points will be assigned to this patch. When this criteria is not met the patch is discarded and the according points stay unassigned.
2) Patch Segmentation: The initially unassigned patches get grouped together to assemble planes. This happens according to their normal vector and position.
3) Post filtering: During this phase, no new patches will be added, but every pixel, which is bordering onto a plane and meets certain conditions, will be assigned to this plane.

1) Patches: As already mentioned. Patches are small equally sized fragments of the depth image and described by their plane equation:

$$ax + by + cz - 1 = 0 \tag{17}$$

The parameters can be acquired by principal component analysis of all points. After getting the parameters, it is necessary to test if they provide a good description of the plane. To ensure this, at least a certain percentage (e.g. 90 %) of the points considered for this patch should be inside the

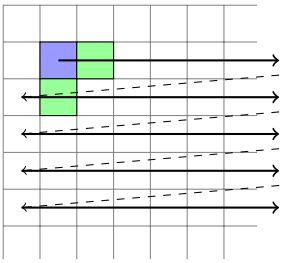


Fig. 7: Segmentation strategy for patches. Every valid patch (blue) is a cluster of points e.g. $10 \times 10\,pixel$ and will be connected to an neighboring (green) existing collection of patches if it fits to one of the existing plane hypothesis. If it can not be added to an existing hypothesis it will become the starting point for a new hypothesis.

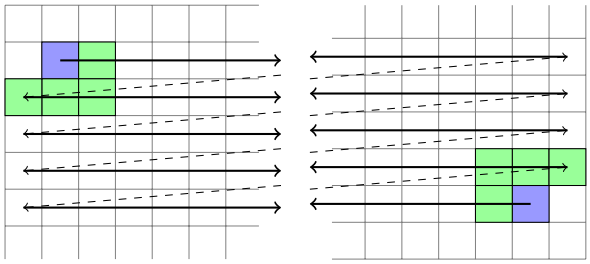


Fig. 8: The bottom up and top down processing steps following the same pattern: The center point (blue) is traversing the image pixelwise in the directions top-down (left) or bottom-up (right). When one of the center points neighboring pixels (green) is a suitable candidate for the center points plane hypothesis, it will get added to this plane.

approximated plane. For this the distance

$$d = \frac{|ax+by+cz-1|}{\sqrt{a^2+b^2+c^2}} < d_{th} \tag{18}$$

has to be below a threshold (e.g. 1 cm).

2) Segmentation: These patches can easily be grouped by any clustering algorithm that supports 4 or 8 connectivity. Neighboring planes or patches can be combined by meeting the criteria of pointing roughly in the same direction e.g. +-15 °.

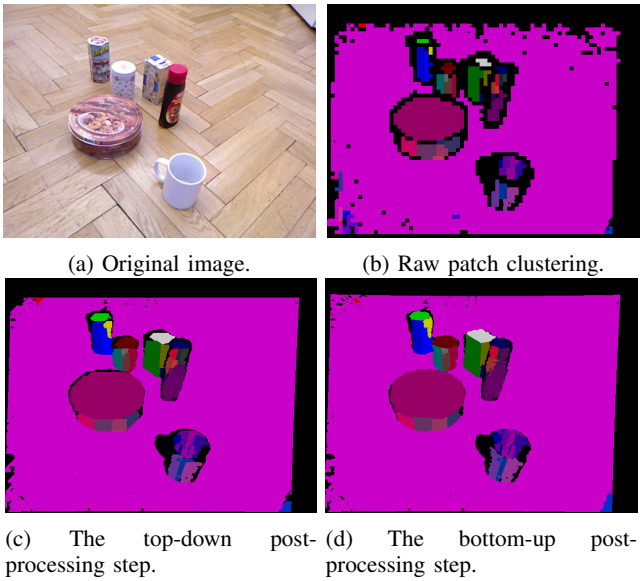
In this implementation it was sufficient to run one pass with the following strategy (see Figure 7):

1) If the current patch (blue) is not already assigned to a plane, create a new plane with this patch as first member.
2) If the neighboring (green) patch to the right has the same normal direction as the plane of the current patch, add the patch (green) to this plane. If the patch to the right is already assigned to a plane, and both plane normals are similar, merge the planes.
3) Merge the patch to the bottom with the current plane if the normal direction is similar.

3) Post-processing: The segmentation of the bigger patches are by far not satisfying because they leave a lot of pixel unassigned. In the last step the filter is running from top left to bottom right and vice versa (see Figure 8) to assign pixel to the most fitting plane. To assign a pixel to a plane it must meet one of the following criteria, otherwise it stays unassigned or assigned to its current plane.

- The considered point is unassigned and fits inside the neighboring plane.
- If the point is already assigned to a plane, which size is a lot smaller (e.g. factor of 10) than the new plane, the point simply has to be close enough ($d < d_{th}$ to get reassigned.
- If the point is already assigned to a plane, which is of similar size ($|P_{new}|f > |P_{current}| > |P_{new}|\frac{1}{f}$) the point has to be closer to the new plane, than to the old plane ($d_{P_{new}} < d_{P_{current}}$).

Note that small planes can't take away points from bigger planes but bigger planes sure can do this to smaller ones.



(a) Original image. (b) Raw patch clustering.

(c) The top-down post-processing step. (d) The bottom-up post-processing step.

Fig. 9: Synopsis of the segmentation process.

This is a strategy to eliminate smaller planes, that might be created in the first step due to oversegmentation. One might replace this strategy by a more sophisticated one. An other parameter that could additionally be taken into account is the normal vector of each point, which should show into the same direction as the plane it is added to.

VI. Results of Segmentation

The simple plane segmentation algorithm provides useable results for indoor scenarios as seen in Figure 9. It is notable that the depthmap quality degrades in the image corners. As a result the algorithm wrongly creates another plane in this region (bottom right corner). Besides this, the algorithm shows the desired behavior. The cylindrical regions around the cans and boxes are approximated by smaller planes, while smaller planar surface patches of boxes get detected as such. In terms of frame rate our algorithm is competitive as it runs at 22 Hz while processing a 640×480 pixel depth map. The algorithms described by Holz [5] (7 Hz) and Wang

[8] (25 Hz) additionally implement some kind of obstacle detection but do not utilize pre-filtering. Apart from this, the conditions are reasonably similar. On the hardware side all results where achieved on an Intel Core i7 with around 2 GHz while utilizing only one CPU core and the GPU for pre filtering.

VII. Conclusion

We introduced a new plane segmentation approach for 2.5 D data. It shows competitive results for both, quality and speed. Our algorithm relies on a filtering step that improves the quality of the input data. Hence, we conducted an analysis of three filters to find a fitting candidate.

We selected the Sigma Adaptive Bilateral Filter wich balances speed and quality. Our GPU implementation of the filter algorithm runs within 25 ms on an AMD Radeon HD 6750M. The mesh denoising algorithm [4], together with our modifications showed promising results. To utilize this algorithm in real-time, GPUs with higher performance could be a possible solution. Apart from this, both filters could be improved by adding a noise model that handles the increased noise levels at higher distances.

The proposed segmentation algorithm shows competitive results that were achieved with a hierarchical strategy. Splitting up the segmentation into a coarse pre-segmentation and a fine grained post-processing step holds the run-time competitive. Future work could extend this algorithm with a sensor model that leads to additional rules such as e.g. depth dependent thresholds.

References

[1] K. Bredies, K. Kunisch, and T. Pock, "Total generalized variation," *SIAM J. Img. Sci.*, vol. 3, no. 3, pp. 492–526, Sep. 2010.

[2] A. Deutschmann, *Kantenselektiver Filter für Punktwolken*. TU Wien, 2013.

[3] 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, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[4] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," in *ACM SIGGRAPH 2003 Papers*, ser. SIGGRAPH '03. New York, NY, USA: ACM, 2003, pp. 950–953.

[5] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, *Real-Time Plane Segmentation Using RGB-D Cameras*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 306–317.

[6] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259 – 268, 1992.

[7] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Jan 1998, pp. 839–846.

[8] Z. Wang, H. Liu, Y. Qian, and T. Xu, *Real-Time Plane Segmentation and Obstacle Detection of 3D Point Clouds for Indoor Scenes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 22–31.

[9] D. Yiruo, W. Wenjia, and K. Yukihiro, "Complex ground plane detection based on v-disparity map in off-road environment," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, June 2013, pp. 1137–1142.

[10] L. Zhang, D. Chen, and W. Liu, "Fast plane segmentation with line primitives for rgb-d sensor," *International Journal of Advanced Robotic Systems*, vol. 13, no. 6, p. 8, 2016.

[11] J. Zhao, J. Katupitiya, and J. Ward, "Global correlation based ground plane estimation using v-disparity image," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 529–534.