SEGMENTING MULTIPLE RANGE IMAGES WITH PRIMITIVE SHAPES

Irene Reisner-Kollmann [*]

Vienna University of Technology

Stefan Maierhofer

VRVis Research Center

ABSTRACT

We introduce a novel method for automatically segmenting multiple registered range images by detecting and optimizing geometric primitives. The resulting shapes provide high level information about scanned objects and are a valuable input for surface reconstruction, hole filling, or shape analysis. We begin by generating a global graph of sample points covering all input frames. The graph structure allows to compute a globally consistent segmentation with a memory and time-efficient solution, even for large sets of input images. We iteratively detect shapes with a Ransac-approach, optimize the assignments of graph nodes to shapes, and optimize the shape parameters. Finally, pixel-accurate segmentations can be extracted for each source image individually. By using range images instead of unstructured point clouds as input, we can exploit additional information such as connectivity or varying precision of depth measurements.

Index Terms— segmentation, shape detection, surface fitting, range data

1. INTRODUCTION

Recent developments of RGB-D sensors provide a cost-efficient, easy to use, and fast way of digitizing interior scenes. While Microsoft Kinect was initially created for gaming, the generated depth and color maps can also be converted to 3D point clouds of real-world objects.

In this paper we present a novel method for segmenting the 3D point cloud into geometric primitives. The detected primitives provide a higher level of abstraction for effectively using, analyzing or editing 3D data. Especially man-made objects, such as CAD-models or interior scenes, contain large parts which can be approximated by basic shapes. Besides surface reconstruction, the generated segmentation might be used for dividing large scenes into smaller parts for simplifying further processing.

We use multiple registered depth images with known intrinsic and extrinsic camera parameters as input. While depth maps can be easily converted to 3D point clouds, they provide additional useful information. The organization in 2D grids implicitly provides connectivity information. The precision of depth measurements usually decreases with the distance to the camera. This knowledge allows to use different distance thresholds across a scene for handling accurately defined as well as noisy parts at the same time.

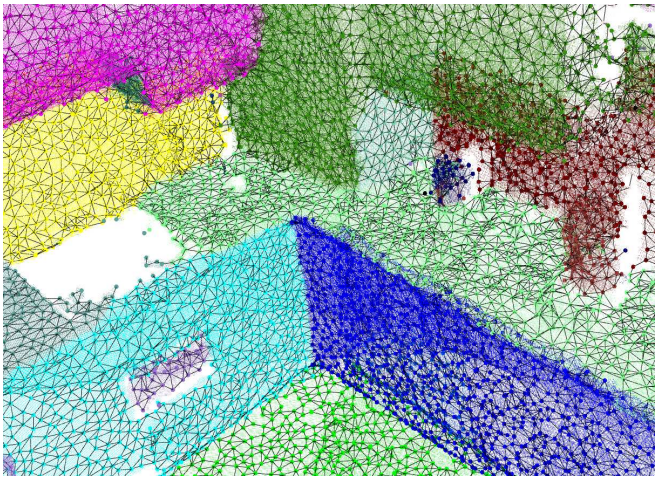


Fig. 1. Segmentation of an interior scene.

Providing a fast and memory-efficient algorithm is difficult for large scenes. We meet the challenge by dividing the problem into two parts. First, a globally consistent segmentation is computed for sample points covering the whole scene. Second, a pixel-accurate segmentation is computed individually for each frame. An example for our segmentation can be seen in Figure 1.

2. RELATED WORK

Segmentation is an important task in many areas of image processing and computer vision. Several approaches have been proposed for segmenting single range images with primitive shapes, e.g. with RANSAC [1], region growing [2], or a recover-and-select strategy [3]. In contrast to these methods we consider multiple registered range images.

Video segmentation exploits temporal coherence to achieve stable segmentations over time. Hierarchical approaches and dividing the video in small clips for processing handle the large amount of pixels [4]. These approaches could also be used for depth cameras, but we support unordered sequences of images and thus do not use any temporal information.

[*]Supported by Doctoral College on Computational Perception.

Multiple range scans of a moving object can be segmented based on its articulated motions [5]. Subsequent frames are registered by applying rigid transformations to differently moving model parts. Then a global optimization iteratively alternates between solving the transformation assignment with graph cuts and optimizing the transformation itself. Our approach is similar, but handles static scenes and points are assigned to shapes instead of transformations.

Graph cut optimization has also been used to segment image sequences, oriented with structure-from-motion [6]. All images are individually segmented into small regions, which are then divided into foreground and background by a global graph cut approach. The segmentation can be refined along the boundary with a pixel-based graph cut optimization. We use a similar coarse-to-fine solution, but due to the dense mapping between frames we are able to start with a global graph instead of segmenting each frame individually.

Very good results for segmenting unordered point clouds into shape primitives have been achieved with a RANSAC-approach [7]. We use this algorithm for initializing the segmentation, but we globally optimize the results with the connectivity given by depth maps.

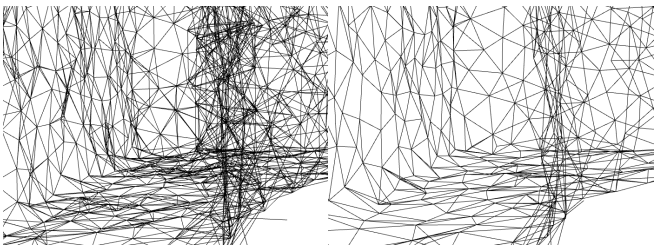
3. ALGORITHM DESCRIPTION

Multiple depth images are aligned to each other, e.g. with iterative closest points (ICP), and optionally improved by a denoising algorithm [8]. The oriented frames can be interpreted as a large graph of 3D points, in which neighboring pixels within one frame as well as corresponding points of different frames are connected by graph edges. As the whole data from multiple frames would be too large, we apply the segmentation first on a set of sample points and compute the final segmentation for each frame based on the global segmentation. This approach allows to globally segment multiple frames with efficient time and memory consumption.

3.1. Graph setup

The central data structure for our algorithm is a sample point graph which is formed from a subset of points sampled from all input frames. Whenever a new frame is added, the sample graph has to be updated to include newly captured areas. The point cloud is sampled such that the average distance between neighboring points is approximately τ. This parameter adjusts the number of node in the graph and provides a trade-off between accuracy and computational time.

We use the modified best-candidate sampling algorithm provided by [5] for selecting new samples. We first create a subset of existing graph points, which are projected onto the current frame or slightly outside of it. Random sample points from the current frame are generated iteratively and the best candidate (with the largest distance to existing sample points) is selected. The average distance of the best candidates is kept over the last 100 trials and adding points is stopped after it falls below the empirically determined value of 0.4τ.



(a) Initial sampling (b) With adaptation

Fig. 2. Adapting sample points to multiple frames highly reduces noise in the sample point graph.

Graph nodes are connected by an edge if their distance is below 2τ. Additionally we assign each pixel in the current frame to its nearest sample point in 3D space. Edges of two nodes are only accepted if at least two neighboring pixels are assigned to the according sample points. This approach prevents connecting close points of different surfaces, e.g. at sharp edges. The 3D points of all assigned pixels are also used for estimating the graph node's orientation with least squares.

In order to reduce noise in the sample point graph, we adapt positions and normals when an existing sample point is visible in a new frame. We compute the weighted average between an existing node and corresponding points in the new frame, though points can only move along their normal. The weights are inversely proportional to the camera depths, and the weights for new points are decreased with every considered frame. Thus, we are pushing sample points towards precisely measured points. Figure 2 shows how noise is removed from the graph and the number of nodes and edges is reduced.

3.2. Global optimization

The global optimization uses only the sample graph and is completely independent from the source images. The goal is to find primitive shapes (plane, cylinder, sphere, cone) in the graph positions and assign each node to the best shape. The optimization is divided into three parts, namely detecting new shapes, optimizing labels and optimizing shapes, which are repeated until convergence.

Detecting shapes We use a RANSAC approach [7] for detecting primitive shapes in all graph positions which are not yet assigned to a shape. In the beginning, this will include all graph positions. The parameter ϵ defines the maximum distance between shapes and points.

Labeling Assigning graph nodes to shapes can be formulated as a labeling problem, which we solve with graph

cuts [9]. The optimization objective is defined as $E_{data} + \lambda E_{smooth}$, where the first term optimizes the distances between graph samples and shapes and the second term strives for assigning neighboring graph nodes to the same shape.

Graph nodes should be assigned to a shape, if the distance to the surface is small and if the point normal coincides with the surface normal, which leads to the following data term:

$$E_{data} = \sum_{i \in G} \frac{d(p_i, S_{l(i)})^2}{3\epsilon} \left(1 - (n_i \cdot n(S_{l(i)}, p_i))\right) \qquad (1)$$

G contains all graph nodes, p_i and n_i denote positions and normals, $l(i)$ is the index of the assigned shape S. The function d computes the distance between a point and its closest point on a shape surface, the function n retrieves the surface normal at the closest point on the shape.

In order to handle outliers and scene parts that cannot be modelled by primitive shapes, it is also possible to label points as unassigned. A constant cost value is applied to such unassigned nodes. A low value might lead to large areas of unassigned points, while a high value might pollute shapes with outliers. In our experiments we generally use a value of 0.8 as cost for unassigned nodes.

The smoothness term strives for assigning nearby points to the same shape. It is defined as

$$E_{smooth} = \sum_{(i,j) \in E} I(l(i) \neq l(j))(n_i \cdot n_j), \qquad (2)$$

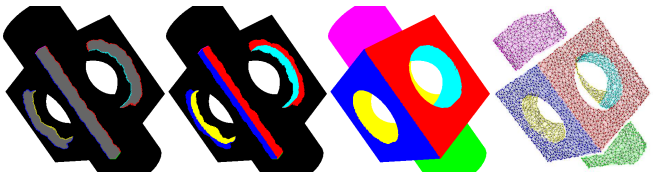
where E is the set of all edges in the sample graph and $I(\cdot)$ is the Potts model, which returns 1 if the argument is true and 0 otherwise. All edges are weighted based on their normal deviation. If two nearby points have very different orientations, it is more likely that they belong to different shapes.

Shape Optimization After the set of assigned points has changed, a refitting step is applied. The geometric error of the shapes is optimized with weighted least-squares. Shapes which have no or only very few points assigned are deleted.

Some scene parts might be slightly distorted in the point cloud, especially when uncalibrated cameras are used and the scene is captured only from few view points. This might lead to wrong initial shape types, e.g. a planar wall is represented by a small part of a cylinder with a large radius. We test all shapes if they can be replaced by a plane without increasing the average residual more than twice. Cones and spheres might also be replaced by a cylinder.

3.3. Frame optimization

In this step we generate a per-pixel segmentation for each frame from the global graph segmentation. Only pixels with a valid depth value are considered because 3D positions are needed. For each pixel, the graph positions within a distance of 3τ are investigated. If all have the same label and the pixel's 3D position and normal are consistent with the according shape, the pixel is labeled with this shape. Otherwise the pixel is marked as unsolved, and the best label has to be determined by optimization.



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

Fig. 3. Frame optimization (white pixels do not have depth values, black pixels are not included in optimization): (a) Dilated pixels with fixed labels, (b) optimized pixels, (c) all pixels (optimized and directly used from global graph), and (d) 3D view.

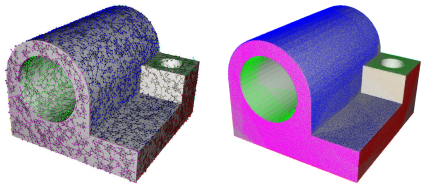


Fig. 4. Segmentation of virtually created depth maps of the joint model. The model is courtesy of AIM@SHAPE.

We apply a graph cut optimization for all unsolved image regions. The regions are slightly enlarged by dilation to ensure valid transitions to the rest of the image. A unique label has already been determined for these dilated pixels. Thus, we apply a cost value of zero to this label and a very high cost value for all other labels. For the unsolved pixels the same optimization objectives are used as for the global graph optimization (Eq. 1 and 2). The smoothness cost is applied to neighboring pixels if they are not divided by large depth discontinuities. Additionally, we want to ensure that the labeling of the global graph is exactly taken over to the individual frames. An additional penalty is added to the data costs for labels not present in the neighborhood of a pixel. Figure 3 shows several stages of optimizing a frame.

Each frame is handled individually and is completely independent from other frames. This approach is memory-efficient and multiple frames can be easily processed in parallel. Depending on the application, it might be a problem that small overlapping areas of different frames are inconsistently labeled when there are no sharp features.

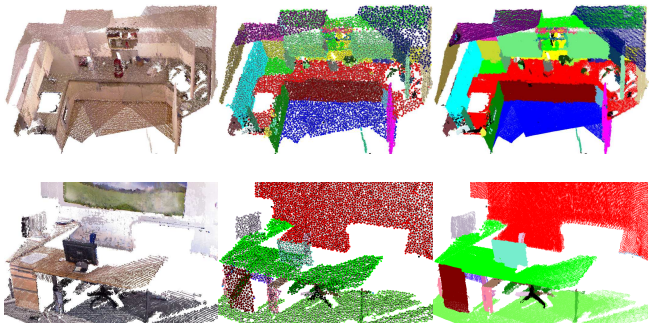


Fig. 5. Real-world scenes of a kitchen and an office. From left to right: colored input frames, segmentation of sample point graph, and all segmented frames.

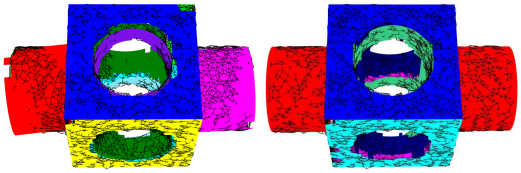


Fig. 6. Block model: initally detected shapes, and result of our optimization. The model is courtesy of AIM@SHAPE.

4. RESULTS

We have tested our algorithm on different input scenes. The results show that basic shapes are successfully detected and the depth maps are segmented along object boundaries. Figure 4 shows results on virtually created depth maps and real world examples can be seen in Figure 5. Figure 6 shows how our new optimization can improve the initially detected shapes. In this case, especially the shape parameters of the left horizontal cylinder have been improved and the left and right cylinder have been merged. As can be seen in Table 1, the frame optimization is the most time-consuming part of our algorithm, but for some applications the global optimization alone might be sufficient.

	Kitchen	Block	Joint
Frames	104	20	30
Points/Frame	253777	133387	113271
Nodes	22809	6323	14104
Shapes	31	11	11
Init/Frame	0.637	0.256	0.365
Optimization	275.2	3.270	7.708
Opt/Frame	8.101	2.925	1.948

Table 1. Runtime measurements on processed models, executed on an Intel Core i7, 2.67 GHz CPU. All timings are in units of seconds. The frames have a resolution of 640x480 pixels, but not all pixels have a valid depth value.

5. CONCLUSIONS

We have presented a novel method for segmenting multiple depth maps into basic shapes. We organize the data in a sample point graph, where we exploit information from depth images such as connectivity and knowledge about varying noise levels. Our algorithm is memory and time efficient by dividing it in a global optimization of the sample point graph and individual optimizations of depth images.

In the future, we would like to combine our approach with segmentation based on color information or object recognition. Our algorithm can be improved for ordered image sequences by including temporal information. Furthermore, we would like to investigate if the registration can be improved with the detected shapes.

6. REFERENCES

[1] P. F. Gotardo, O. R. Bellon, and L. Silva, "Range image segmentation by surface extraction using an improved robust estimator," in *CVPR*, 2003, vol. 2, pp. 33–38.

[2] M. Djebali, M. Melkemi, and N. Sapidis, "Range-image segmentation and model reconstruction based on a fit-and-merge strategy," in *7th ACM symposium on Solid modeling and applications (SMA)*, 2002, pp. 127–138.

[3] D. Marshall, G. Lukacs, and R. Martin, "Robust segmentation of primitives from range data in the presence of geometric degeneracy," *PAMI*, vol. 23, no. 3, pp. 304–314, 2001.

[4] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient Hierarchical Graph-Based Video Segmentation," in *CVPR*, 2010, pp. 2141 –2148.

[5] W. Chang and M. Zwicker, "Global Registration of Dynamic Range Scans for Articulated Model Reconstruction," *ACM Trans. Graph.*, vol. 30, pp. 26:1–26:15, 2011.

[6] M. Sormann, C. Zach, and K. Karner, "Graph Cut Based Multiple View Segmentation for 3D Reconstruction," in *3DPVT*, 2006, pp. 1085–1092.

[7] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for Point-Cloud Shape Detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, 2007.

[8] I. Reisner-Kollmann and S. Maierhofer, "Consolidation of Multiple Depth Maps," in *IEEE Workshop on Consumer Depth Cameras for Computer Vision*, 2011.

[9] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images," in *ICCV*, 2001, pp. 105–112.