Real-Time 3D Segmentation of Cluttered Scenes for Robot Grasping

Andre Ückermann, Robert Haschke and Helge Ritter

***Abstract*— We present a real-time algorithm that segments unstructured and highly cluttered scenes. The algorithm robustly separates objects of unknown shape in congested scenes of stacked and partially occluded objects. The model-free approach finds smooth surface patches, using a depth image from a Kinect camera, which are subsequently combined to form highly probable object hypotheses. The real-time capabilities and the quality of the algorithm are evaluated on a benchmark database. Advantages compared to existing approaches as well as weaknesses are discussed. We also report on an autonomous grasping experiment with the Shadow Robot Hand which employs the estimated shape and pose of objects given by our algorithm in a task in which it cleans a table.**

I. INTRODUCTION

Autonomous grasping of objects from a pile of unknown objects is still a major challenge in robotics. Although robust grasp planning and execution approaches are available, they typically require precise shape and pose information to select a grasp in an offline optimization process [1], [2]. In order to acquire the necessary shape and pose information, traditional approaches typically employ a-priori knowledge about object models [1], [3], [4], [5], which is used for object recognition and the subsequent planning process. However, this approach restricts grasping to objects whose models, i.e. geometric shape and/or visual appearance, are known in advance. To relax this constraint, Bohg et al. assume a reflection symmetry of objects and appropriately augment an object's frontal view for grasp planning [6].

In contrast to these traditional, planning-based methods, biologically motivated approaches exist, which can successfully grasp objects based on coarse shape and pose information without prior planning. While our previous work in this direction [7] was limited to simple 2D scenes, in this paper we propose a 3D scene segmentation approach which separates objects and provides *coarse* shape and pose information suitable for grasping. The algorithm is model free and employs only generic smoothness constraints. The obtained object information is too coarse to be used for classical grasp planners, but the compliant grasping scheme tolerates those inaccuracies.

Our algorithm combines two segmentation methods, both operating on depth images only: the identification of object surfaces and edges based on the detection of "surface normal edges" and a composition of these sufaces into sensible object hypotheses. The algorithm robustly separates objects in cluttered scenes as shown in Fig. 1. The approach can operate in real-time to facilitate interactive usage in human-robot-cooperation tasks. The main advantage, in contrast to existing methods, is the capability to separate unknown, stacked, nearby, and partially occluded objects in a model-free manner, without prior knowledge of these objects. Naturally, this approach has its limitations compared to model-based approaches, especially if very complex object shapes are to be considered. However, it provides an initial object hypothesis in arbitrary situations, which can be refined by active exploration [8] and even fed as input to model-based adaptive methods.

Fig. 1. Raw depth image (left), color image (middle) and resulting 3D object segmentation (right).

Many existing segmentation algorithms aim for a simultaneous recognition of objects and their pose. To this end, found image features are matched to a database of known objects. Various feature extraction methods have been proposed, including 3D-augmented SIFT features [1], [8] and features directly obtained from range images such as a viewpoint feature histogram [3], depth-encoded hough voting [4], point pair features [15], and iterative clustering-estimation [16]. In [18] RGB-stereo, time-of-flight, and thermal cameras are fused to segment kitchen objects in a table-top setup. The addition of other sensor modalities, such as temperature, improves the segmentation accuracy and enables the method to segment shiny and translucent object. These approaches robustly recognize *partially occluded* objects and correctly estimate their pose from stored 3D models, but are always restricted to the known set of objects.

Other approaches, directly operating on point clouds, better generalize to unknown objects. In [5], [9], [17] table-top scenarios are segmented based on an initial clustering into horizontal support planes. Point clusters supported by these planes, i.e. lying above the plane and within its 2D bounding box after projection, are considered as objects. Subsequently in [5] hybrid object models comprising primitive shape models (planes, cylinders, spheres, cones fitted into the data points) and surface meshes (modelling residual points) are determined. The algorithm in [9] is tuned towards real-time performance, achieving frame rates of 30Hz on images sized 160×120. [17] adds support for arbitrary rotational surfaces.

This work was supported by the German Collaborative Research Center "CRC 673: Alignment in Communication" and the Center of Excellence Cognitive Interaction Technology (CITEC), both granted by the DFG. The authors are with the Neuroinformatics Group at Bielefeld University, Germany. {aueckerm|rhaschke|helge}@techfak.uni-bielefeld.de

However, due to the projection approach to obtain object clusters, all three methods cannot handle stacked objects. Methods finding smoothly connected areas in point clouds are presented in [10], [11]. Their region growing approach closely resembles the first segmentation step of our method, but applies more costly operations. Using parallelized operations, our method is much faster with comparable results.

Point normal estimation is a basic component of many segmentation approaches as well as in our method. Most methods use a form of least squares, RANSAC, or PCA to fit a plane into a set of neighboring points [5], [10], [11]. As most methods focus on arbitrary 3D point clouds, emphasis is on efficient selection of those points. Exploiting the intrinsic grid structure of range images, rough normal estimations can be computed much faster from the crossproduct of tangential vectors. For example, [9] proposes a method employing integral images to yield real-time performance. In this work, we follow a similar approach to compute normal vectors.

The remaining paper is organized as follows: The next section introduces the segmentation algorithm in detail. In sec. III we evaluate the robustness and quality of the obtained segmentation results and present a grasping approach drawing on these results. Finally, we give a short conclusion and mention possible future work.

II. 3D SCENE SEGMENTATION METHOD

Before introducing the details of the process flow, we outline the overall structure of the algorithm. It can be split into two main parts: the determination of surface patches and object edges, and a subsequent combination of these low-level segments into high-level object segments. In contrast to the commonly employed segmentation method provided by the Point Cloud Library [12], which aims to fit specific object models, the proposed approach is model-free and can successfully handle unknown, stacked, and nearby objects.

The raw depth images, obtained from the Kinect camera, provide low-noise depth information (see Fig. 1). Hence, we decided to solely focus on depth images, ignoring color, and thus diminishing the impeding influence of strong textures giving rise to oversegmentation. In the future we plan to integrate color information to disambiguate difficult scenes. Looking more closely at the depth image, we can identify two situations, exposing object edges: (i) discontinuous jumps of depth values, and (ii) sudden changes of the surface normal direction, e.g. when an object is lying on the table. Consequently, at the core of our algorithm is the determination of those "surface normal edges", which are used as the basis to segment the image – in a first step – into connected surface patches and separating edges using a region growing method.

The second part of the algorithm subsequently combines found surface patches into meaningful object segments. To do this, we analyze the graph structure determined by the found surface nodes and their adjacency relation. Using a greedy strategy, we determine a highly probable object segmentation hypothesis. This novel approach allows us to segment highly cluttered scenes with unknown objects without employing any model knowledge.

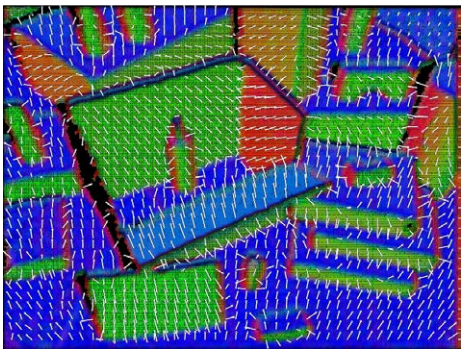


Fig. 2. Point cloud with normal directions (XYZ mapped to RGB).

A. Partition into Surfaces and Edges

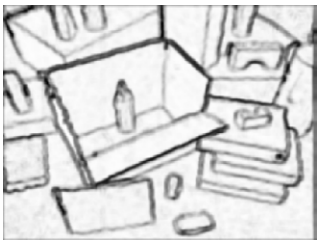
The objective of the first processing step is to segment the depth image into regions of (smoothly curved) surfaces, continuously enclosed by sharp object edges. Additionally, we transform the raw depth image into a 3D point cloud, which is represented w.r.t. a robot-defined coordinate frame.

a) Determination of Surface Normals: As a basis for computing "surface normal edges", we first determine surface normals for every image point. To this end, we simply determine the surface normals from the plane spanned by three points in the 3×3 neighbourhood of the considered central image point using the classical cross product. In our previous work, we evaluated more accurate methods based on principal component analysis [22]. However, they did not provide better segmentation results and were much slower.

Note, that the determination of surface normals is directly performed on the raw depth image, instead of the 3D point cloud. That is, the 2D image coordinates are augmented by the depth value to yield valid three-dimensional vectors. This procedure yields much more distinct changes of the normal direction at the boundary of objects, because the smoothing effect due to 3D projection is avoided.

In order to reduce sensor noise and to obtain smooth and stable surface normal estimations, we apply a three-stage smoothing procedure. First a 3×3 median filter is applied to the raw depth image. Secondly, we apply a smoothing over time filter, averaging the median-filtered depth values of all individual image pixels within the last n frames. Finally, after calculating the normals, they are smoothed again, applying a convolution using a 5×5 Gaussian kernel. Fig. 2 shows the resulting surface normals for the image in Fig. 1, mapping xyz normal directions to the RGB color space.

This filter chain immensely reduces sensor noise, but also evokes motion blur due to the second filter stage. For low dynamic scenes we have found that a value of $n = 6$ is a good choice to balance motion blur and the smoothing effect. Another issue, the 2nd filter brought to our attention, was jumping depth values at object edges. Here the depth value fluctuates between the foreground (on the object) and the background, because of sensor noise evaluated in Kinect's



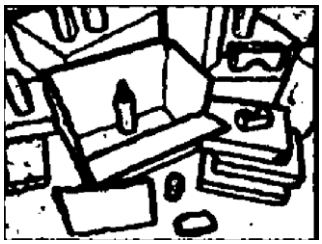
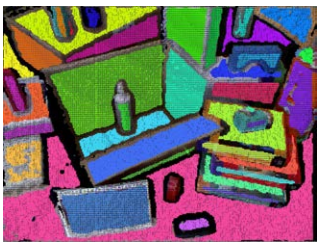


Fig. 3. Result of the edge filter applied to the image of surface normals of Fig. 2 (left: filter result, right: binarized version). White surface patches are properly enclosed by black, uninterrupted object edges. Restriction to point cloud data avoids oversegmentation due to changes in texture.



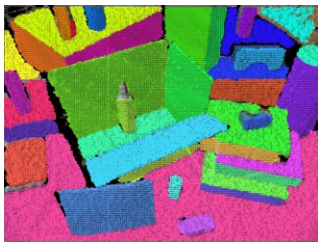


Fig. 4. Results of the first segmentation into surface patches and edges (left), and after assignment of edge points to closest surfaces (right).

internal processing pipeline. Instead of averaging between these extremal values, which would yield a poor mean estimation, we choose the minimal or maximal depth value (depending on which is closer to the mean) of all observed measurements within the given time window, if these extreme values diverge too much.

b) Detection of Surface Normal Edges: One major contribution of our paper is the fast detection of surface normal edges, which is based on the computation of the scalar product of adjacent surface normals n_1, n_2. To obtain clear, uninterrupted edges, suitable for subsequent application of a region growing algorithm, we look for edges in all eight directions defined by the neighboring pixels of a point, i.e. north (N), east (E), south (S), west (W), as well as NE, SE, SW, NW. The final result of the edge filter is obtained from averaging the results of all eight scalar products. While large values, close to one, correspond to flat surfaces, smaller values indicate increasingly sharp object edges.

Finally, binarizing the obtained edge image by employing a threshold value $\theta_{max} = 0.85\ (31, 8^\circ)$, we can easily separate edges from smoothly curved surfaces. Fig. 3 illustrates the result of this processing step: Object edges are clearly visible as bold lines, while smooth and large surfaces form homogeneous white regions. A considerable number of false edges are still detected due to noise. However those regions are small and disjointed and thus can be easily filtered out in subsequent processing steps. Note, that small or narrow objects are often represented by edges only, while smooth surfaces are separated by a relatively thin edge.

c) Segmentation into Surface Patches: Finally, we apply a simple region growing algorithm to the binarized edge image in order to associate each surface point with a unique patch ID as shown in Fig. 4.

The fast surface patch segmentation based on normal edges already provides a detailed segmentation of the scene into surface patches, and is then employed in the subsequent object segmentation step, introduced next.

B. High-Level Object Segmentation

In the second processing block, we ultimately aim for segmentation on an object level, which means that the previously found surface patches need to be combined to form proper *object regions*. In contrast to our previous work [22], where we employed a simple heuristic method to split point cloud clusters along cutting planes, here we propose a much more general approach which avoids many of the shortcomings of our previous method. In particular, we determine a directed, weighted graph, modeling the topological neighborhood structure and the probability of two adjacent patches belonging to the same object region. Subsequently this graph structure is analyzed to find the most probable segmentation into object regions in a greedy manner.

d) Adjacency Matrix and Assignment of Edge Points: The adjacency matrix representing the connectivity of surface patches is determined as follows: For every edge point p_r we consider all neighboring surface points p_i in a radius r (in image space), which have an Euclidean distance $\|p_r - p_i\|$ smaller than a given threshold $d_{\max}$. Each possible pair from this list of obtained surface IDs is marked as adjacent.

Additionally, the edge point p_r is associated to the surface patch having the smallest Euclidean distance to p_r. By this means, almost all edge points can be associated to their closest surface patch (cf. Fig. 4). An exception are edge points, which are part of very bold edges (such that no surface points within the radius r can be found) or edge points, whose closest matching surface point is more distant than $d_{\max}$. In both cases, these edge points probably belong to a separate, but small object, and they are processed later on. If the search radius r and the Euclidean distance $\|p_r - p_i\|$ were not restricted, such small objects would be absorbed by their supporting surface. Fig. 5 illustrates another reason for limiting the Euclidean distance to d_{max}: surfaces 6 and 10 are neighbors in image space, but obviously not in 3D space and therefore shouldn't be considered adjacent.

e) Cutfree Neighbors: To further improve the adjacency matrix, we apply a plausible heuristic check already proposed in our previous work [22]. Two neighboring surfaces presumably do *not* belong to a common object – and thus should be removed from the adjacency matrix – if one surface cuts the other, such that a considerable amount of points are lying on both sides of the former surface. For illustration, consider surfaces 1 and 10 in Fig. 5. While all points of surface 1 are on top of surface 10, the plane fitted into surface 1 cuts surface 10. Hence this surface combination is disregarded. On the other hand surfaces 1 and 2 are pairwise cut-free.

To speed up the cutting test, we approximate surfaces by a set of 20 planes fitted using RANSAC [13]. If most of these planes are cut-free with an adjacent surface (up to a small tolerance to account for outliers), the corresponding surface pair is kept for further consideration. The result of this processing step is a non-directed graph representing the

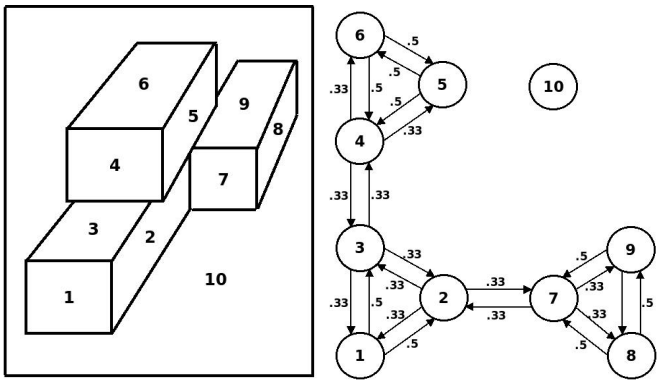


Fig. 5. Illustration of the probabilistic composition approach. A composition probability graph (right) is shown for an example scene (left).

topology of neighboring surfaces, modeled by a symmetric, boolean adjacency matrix.

f) Probabilistic Composition: In the next processing step, this graph is searched for strongly connected sub-graphs, which are considered as good candidates for object segments. To this end, we follow a greedy approach combining the most probable surface patches into an object segment. The method starts with all (ordered) pairs of surfaces (i, j) to which a uniform weight $w_{ij} = 1/n$ is assigned, where n denotes the number of adjacent nodes of node i. This results in a directed, weighted graph as illustrated in Fig. 5.

Subsequently, this list of pairwise node combinations is extended to triples, again assigning a weight according to the following heuristic rule: If (i, j) and (i, k) are edges in the graph and there also exists an edge (j, k), such that the sub-graph (i, j, k) is complete, we assign a weight $w_{ijk} = w_{ij} + w_{ik}$. Otherwise the combination i, j, k is not further considered. Finally, object regions are formed greedily, starting with surface groups of maximal weight. If multiple combinations have the same weight, combinations with more surfaces are preferred. This process is iterated until all surfaces are uniquely assigned to an object region.

Spherical, cylindrical and box-like objects are correctly handled by this approach, because they expose 1-3 faces. To cope with more complex objects, we are working on an even more flexible face clustering method.

g) Remaining Edge Points: In the final processing step, all remaining edge points (left after step II-B.0.d) have to be processed to obtain the final segmentation result shown in Fig. 1. Firstly, the remaining points are segmented using a region growing algorithm working in the image plane and using the Euclidean distance as the criterion of uniformity. These segments are then processed according to the following rules:

- If a segment has no neighboring faces (caused by missing depth information), it becomes a separate object.
- If a segment has one neighboring face and comprises very few points only, they are assigned to this neighbor.
- If a segment is completely enclosed by a single neighboring face, it becomes a new object. If it is not completely enclosed, all points are assigned to the neighboring region.
- If a segment has more than one neighbor and all neighbors are part of a common object, it will be assigned to this object.
- If a segment has more than one neighbor corresponding to different objects, all points are assigned to the best matching neighboring plane using RANSAC.

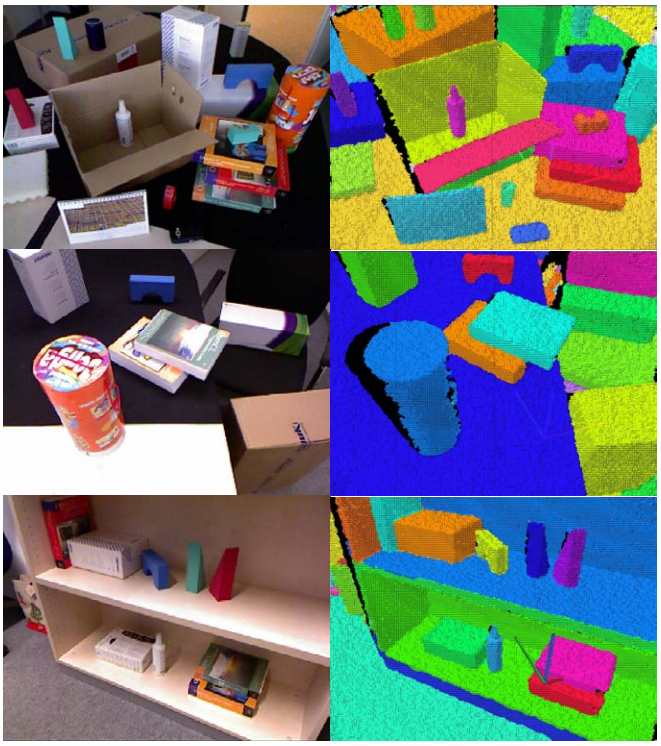


Fig. 6. Four complex scenes and their corresponding segmentation result.

This method yields meaningful segmentations of highly complex scenes with stacked and completely unknown objects without the use of prior knowledge, object models, or the need to extract support planes (which might not be possible in crowded scenes).

III. EVALUATION

A. Runtime Performance and Quality

In this section we evaluate the efficiency of the proposed approach, report on the runtimes of the algorithm and show some qualitative results. Fig. 6 shows three example scenes. The algorithm robustly segments table top scenes as well as other indoor environments, e.g. offices. It is applicable to scenes of different complexity with objects of various size and shape.

Table I shows the runtimes for these scenes. The more complex the scene, the faster the algorithm. This assertion seems to be counter-intuitive. However, most of the processing time is taken by the calculation of cut-free surface pairs. For a simple scene, e.g. a table with many spatially separated objects, many small faces emerge, which have to be checked against the large table surface. In more complex scenes of stacked objects, we obtain several small and mid-size faces. Hence, a possibly fewer number of pairwise tests has to be

TABLE I

RUNTIME (MS) OF ALGORITHMIC STAGES FOR SCENES SHOWN IN FIG. 6.

	scene 1	scene 2	scene 3
time smoothing	1	1	1
point cloud proj.	1	1	1
edge detection	1	1	1
region growing	7	7	7
adjacency matrix	10	8	9
cut-free pairs	55	53	80
remaining points	8	4	7
visualization	11	11	11
overall time (Hz)	94 (10.6)	86 (11.6)	117 (8.5)

accomplished, each with a much smaller number of points within each surface.

The runtime is composed of the time smoothing filter, the 3D point cloud projection, edge detection (including median filtering, normal calculation, smoothing, edge calculation and binarization), region growing, determination of the adjacency matrix (including assignment of edge points), the identification of cut-free surface pairs, the assignment of the remaining points, and the point cloud visualization.

The determination of the adjacency matrix, the assignment of remaining edge points, and the cut-free testing vary with the complexity of the scene, whereby the framerate is at least 8 fps. Most parts of the algorithm are parallelized using a nVidia GTX560 graphics card. The remaining parts are processed on a single core of a XEON 2.53 GHz processor. Using a faster graphics card and considering only subsets of points for plane fitting using RANSAC in the most costly cut-free calculations, the algorithm could be further tuned towards the maximum frame rate of the Kinect camera (30 Hz). The algorithm always operates on QVGA resolution (320×240).

B. Quantitative Results

In this section, we present quantitative segmentation results for data taken from the Object Segmentation Database [19], which provides a huge set of table-top scenes, including a color image, point cloud data, and a reference segmentation for each scene. To employ the database for our algorithm (which works on raw depth images), we back-projected the PCL point cloud to a depth image. Using the labeled segmentation results, we evaluate the segmentation quality of our algorithm. To this end, we consider the set R_i, comprising all points of the reference segmentation of object i and the set S_i, comprising all points of our segmentation result. $TP_i = R_i \cap S_i$ shall denote the overlap of both sets, i.e. the set of correctly segmented points (true positive). $FP_i = S_i \setminus TP_i$ and $FN_i = R_i \setminus TP_i$ shall denote the sets of false positive and false negative points, which are only assigned to one of the sets resp. The equations

$$tp = \frac{1}{n}\sum_{i=1}^{n}\frac{|TP_i|}{|R_i|},\ fp = \frac{1}{n}\sum_{i=1}^{n}\frac{|FP_i|}{|S_i|},\ fn = \frac{1}{n}\sum_{i=1}^{n}\frac{|FN_i|}{|R_i|}$$

calculate the average scores, where n is the number of object segments for an individual image of the database.

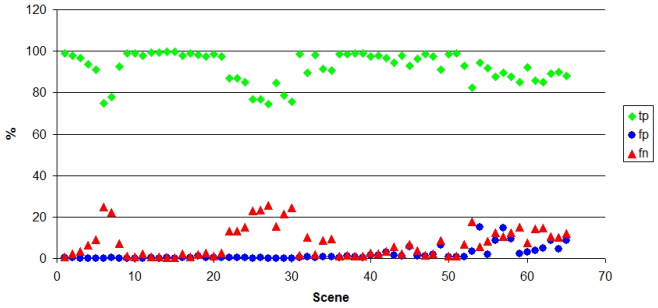


Fig. 7. Evaluation results of the Object Segmentation Database [19].

TABLE II

QUANTITATIVE SEGMENTATION RESULTS (IN %).

	true positive	false positive	false negative
Test51	98.8	0.9	1.1
Test63	89.2	8.6	10.7
Mean	92.2	1.9	7.8
Std. deviation	7.3	3.3	7.3

Table II shows these scores for selected scenes (shown in Fig. 8) as well as their mean and standard deviation obtained by averaging over all test scenes in the database. The results of all test scenes in the database are illustrated in Fig. 7. The true positive score is between 74.3% and 99.7%. More segmentation results are available at the project website [20]. The reasons for the low scores less than 90% are described in the next section. Unfortunately, we did not find other work employing this benchmark for comparison.

C. Strengths and Weaknesses

The presented real-time segmentation algorithm works very well with a huge number of complex scenes. However, there are three remaining issues which are not yet handled by our algorithm. These problems are illustrated in Fig. 9. Firstly, the inner and outer part of open, curved containers, like cylinders, are decomposed into separate object regions, because their curved surfaces do not directly neighbor with a

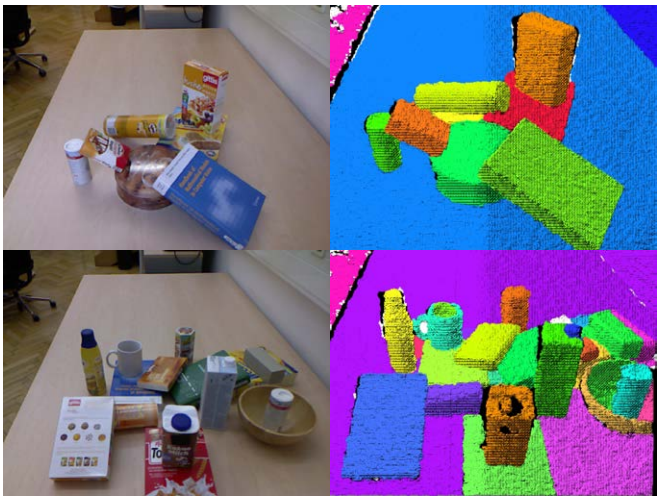


Fig. 8. Segmentation results for scenes 42, 51, and 63 of the Object Segmentation Database [19].

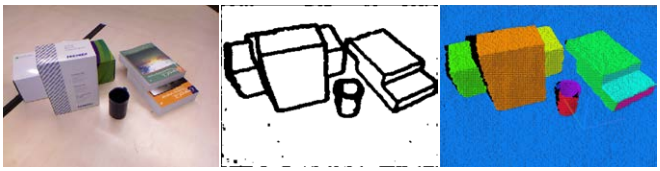


Fig. 9. Weak points of the algorithm: (1) open curved objects, (2) object split by complete occlusion, (3) failing edge detection for aligned object surfaces.

distance smaller than $d_{\max}$. However, for many applications it might be beneficial to identify openings in concave vessels, e.g. in order to be able to fill in something. If desired, these regions can easily be combined using model-rich approaches.

Secondly, surfaces which are disjointed due to occlusion by another object, are not recombined by our model-free approach. Using RANSAC-based model fitting and integrating texture information, both object parts can be easily combined if needed. For our grasping application, this is not necessary, because we can simply grasp one or the other object part. The very low scores smaller than 80% in Fig. 7 are all caused by an object separation through complete occlusion.

Finally, objects, whose surfaces are perfectly aligned (like steps of a staircase), are not correctly segmented. In this case, the first-stage segmentation combines the surface patches of both objects into a common region, because a visible edge doesn't emerge.

D. Grasping Experiment

We used the segmentation approach for autonomous grasping with the 24-DOF Shadow Robot Hand employing our biologically inspired grasping strategy [7]. To obtain a coarse shape model of the object – which is required to select a grasp prototype, i.e. power, precision, or pincer grasp based on object size, and to correctly align the hand to the object – we fit a superquadrics model [14] to the 3D points of a selected object blob given by our segmentation algorithm. This model determines the position and orientation as well as the coarse size and shape of the object. With this information, we can apply our grasping strategy. We also evaluated PCA to determine object pose and shape. However, while the PCA model is always shifted towards the camera, the superquadrics model can correctly account for the invisible backside of the object. In the cleanup experiment, shown in the accompanying video [21], the robot selects autonomously the object with the largest z-position, grasps it and puts it into a bowl. This procedure is iterated until all objects are removed. Too big and thus non-graspable objects are marked and not further considered.

IV. CONCLUSION AND FUTURE WORK

In this paper, we introduced a model-free segmentation algorithm for cluttered scenes which is not restricted by a given set of object models, world knowledge, or the ability to extract supporting planes, which represents the current state-of-the-art. A fast algorithm to determine object edges using edge detection on surface normals was combined with a novel graph-based method to combine surface patches to form highly probable object hypotheses. The algorithm can deal with stacked, nearby, and partially occluded objects, which is achieved by finding object edges in depth images and the novel idea to identify adjacent and cut-free surface patches, which can be combine to form object regions. The algorithm was evaluated w.r.t. real-time capabilities and segmentation quality.

To allow direct interaction with users, a distinction of objects from the human hand would be desirable. To this end, color histograms could be used. Color and texture will also provide valuable information in the segmentation process to disambiguate certain situations and to recombine split object parts. However, the current approach has the benefit of autonomously detecting exposed object parts. The obtained segmentation result should be considered as an initial, high-quality hypothesis for the structure of a scene which can be further refined by active exploration [8], or model-rich approaches.

REFERENCES

[1] J. Kuehnle, A. Verl, Z. Xue, S. Ruehl, M. Zöllner, R. Dillmann, T. Grundmann, R. Eidenberger, R. Zöllner, 6D object localization and obstacle detection for collision-free manipulation, *Proc. ICAR*, 2009

[2] Z. Xue, A. Kasper, M. Zöllner, R. Dillmann, An automatic grasp planning system for service robots, *Proc. ICAR*, 2009

[3] R.B. Rusu, G. Bradski, R. Thibaux, J. Hsu, Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram, *Proc. IROS*, 2010

[4] Sun, Xu, Bradski, Savarese, Depth-Encoded Hough Voting for Joint Object Detection and Shape Recovery, *Proc. ECCV*, 2010

[5] R.B. Rusu, N. Blodow, Z.C. Marton, M. Beetz, Close-range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Domestic Environments, *Proc. IROS*, 2009

[6] J. Bohg, M. Johnson-Roberson, B. León, J. Felip, X. Gratal, N. Bergström, D. Kragic, A. Morales, Mind the Gap - Robotic Grasping under Incomplete Observation, *Proc. ICRA*, 2011

[7] F. Röthling, R. Haschke, J.J. Steil, H.J. Ritter, Platform Portable Anthropomorphic Grasping with the Bielefeld 20-DOF Shadow and 9-DOF TUM Hand, *Proc. IROS*, 2007

[8] E.S. Kuzmič, A. Ude, Object segmentation and learning through feature grouping and manipulation, *Proc. Humanoids*, 2010

[9] D. Holz, S. Holzer, R.B. Rusu, S. Behnke, Real-Time Plane Segmentation using RGB-D Cameras, *RoboCup Symposium*, 2011

[10] T. Rabbani, F.A. van den Heuvel, G. Vosselman, Segmentation of Point Clouds using Smoothness Constraint, *Int. Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, vol. 36(5), 2006

[11] E. Castillo, H. Zhao, Point Cloud Segmentation via Constrained Nonlinear Least Squares Surface Normal Estimates, *Recent UCLA Computational and Applied Mathematics Reports*, 2009

[12] R.B. Rusu, S. Cousins, 3D is here: Point Cloud Library (PCL), *IEEE International Conference on Robotics and Automation (ICRA)*, 2011

[13] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Communications of the ACM*, vol. 24, issue 6, 1981

[14] A.H. Barr, Superquadrics and Angle-Preserving Transformations, *IEEE Computer Graphics and Applications*, vol. 1, issue 1, 1981

[15] E. Kim, G. Medioni, 3D Object Recognition in Range Images Using Visibility Context, *Proc. IROS*, 2011

[16] A. Collet, M. Martinez, S.S. Srinivasa, The MOPED framework: Object Recognition and Pose Estimation for Manipulation, *International Journal of Robotics Research*, vol. 30, issue 10, 2011

[17] Marton, Pangercic, Blodow, Kleinehellefort, Beetz, General 3D Modelling of Novel Objects from a Single View, *Proc. IROS*, 2010

[18] Marton, Rusu, Jain, Klang, Beetz, Probabilistic Categorization of Kitchen Objects in Table Settings with Composite Sensor, *IROS*, 2009

[19] Object Segmentation Database: http://www.acin.tuwien.ac.at/?id=289

[20] Project web site: http://ni.www.techfak.uni-bielefeld.de/node/3249

[21] Project video: http://www.youtube.com/watch?v=Z2SwggQTBC8

[22] A. Ückermann, C. Elbrechter, R. Haschke, H. Ritter, 3D Scene Segmentation for Autonomous Robot Grasping, *Proc. IROS*, 2012