3D scene segmentation for autonomous robot grasping

Conference Paper *in* Proceedings of the ... IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems · October 2012

DOI: 10.1109/IROS.2012.6385692

Christof Elbrechter

Bielefeld University

Robert Haschke

Bielefeld University

3D Scene Segmentation for Autonomous Robot Grasping

Andre Ückermann, Christof Elbrechter, Robert Haschke and Helge Ritter

***Abstract*— We present an algorithm to segment an unstructured table top scene. Operating on the depth image of a Kinect camera, the algorithm robustly separates objects of previously unknown shape in cluttered scenes of stacked and partially occluded objects. The model-free algorithm finds smooth surface patches which are subsequently combined to form object hypotheses. We evaluate the algorithm regarding its robustness and real-time capabilities and discuss its advantages compared to existing approaches as well as its weak spots to be addressed in future work. We also report on an autonomous grasping experiment with the Shadow Robot Hand which employs the estimated shape and pose of segmented objects.**

I. INTRODUCTION

Autonomous grasping of objects from a pile of unknown objects is still a major challenge in robotics. Although grasp planning 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 employed 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 contrast to this work, biologically motivated approaches exist, which can successfully grasp objects based on coarse shape and pose information without prior planning. While our previous work into this direction [6] is limited to simple 2D scenes, in the present 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 only employs generic smoothness constraints. The obtained object information is too coarse for an application in 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 detection of "surface normal edges" and a partitioning of these regions into object hypotheses. The algorithm robustly separates objects in cluttered scenes as shown in Fig. 1. The main advantage in contrast to other methods is the capability to separate unknown, stacked, nearby, and partially occluded objects in a model-free manner, without prior knowledge about these objects. Naturally, this approach is limited 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 and fed as input to model-based, adaptive methods.

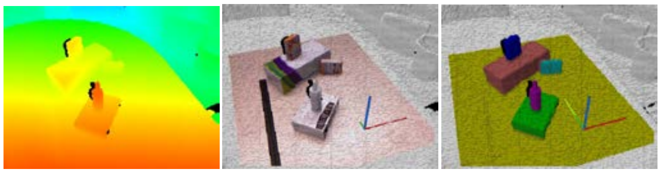

Fig. 1. Raw depth image (left), point cloud cropped to reachable task space (middle) and resulting object segmentation (right).

Many existing work aims for a simultaneous recognition of objects and their pose. To this end, image features are matched to a database of known objects. Various feature extraction methods were proposed, including 3D-augmented SIFT features [1], [7] and point-cloud features like viewpoint feature histogram [3], depth-encoded hough voting [4], point pair features [14], or iterative clustering-estimation [15]. Integrating RGB-stereo, time-of-flight, and thermal cameras, also shiny and translucent objects can be segmented [17], which is a challenging task otherwise. Although these approaches robustly recognize *partially occluded* objects, they are restricted to a predefined set of models.

Other approaches, directly operating on point clouds, better generalize to unknown objects. In [5], [16] table-top scenarios are segmented based on an initial clustering into horizontal supporting surfaces. Point clusters, supported by these planes, i.e. lying above the plane and within the 2D bounding box after projection, are considered as objects. Subsequently [5], [16] determine hybrid models from these clusters by fitting primitive shape models (cuboids, cylinders, etc.) and surface meshes (modeling residual points) into these points. It's also possible to estimate rotational surfaces [16]. While these models can derive very detailed geometrical models of point clusters, they fail to separate stacked objects. A similar method, tuned towards real-time performance, achieving 30Hz on 160×120 images, skips the modeling step completely [8]. Model-free methods, finding smoothly connected areas in point clouds, are presented in [9], [10]. Their region growing approach closely resembles the first segmentation step of our method, but cannot correctly separate individual objects. The present paper fills this gap, proposing a simple, model-free heuristics to combine found surface patches into object hypotheses. Additionally, we present a surface-segmentation method tuned towards real-time performance.

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|celbrech|rhaschke|helge}@techfak.uni-bielefeld.de

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], [9], [10]. 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 cross product of tangential vectors. For example, [8] proposes a method employing integral images. In this paper we compare these approaches regarding accuracy and efficiency.

In the next chapter we introduce our segmentation algorithm in detail. Afterwards, we evaluate the robustness and quality of the algorithms and present a grasping experiment using this algorithm in chapter III. At the end, we give a short conclusion and mention possible future work.

II. 3D SCENE SEGMENTATION METHOD

Before introducing the details of the processing 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 those low-level segments into high-level object segments. In contrast to the commonly employed segmentation method provided by the Point Cloud Library [11], 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 (cf. 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 future we will 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 at object edges. Consequently, the core of our algorithm is the determination of those "surface normal edges", which is the basis for a region growing method to segment the image in a first step into connected surface patches and separating edges.

The second part of the algorithm subsequently combines found surface patches into object segments. Similar to [16], we extract support planes and further separate object blobs using Euclidean segmentation. In contrast to these approaches, we then successively split these point sets (usually comprising several objects) along previously found surfaces in order to separate individual items. This novel, model-free heuristics enables to separate objects which are close or touching each other.

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. To reduce the computational effort, we restrict all calculations to the reachable task space, which is the space reachable either by the human user or the robot in a cooperative table-top scenario. To this end, we transform the depth image into a 3D point cloud. Figure 1 shows a raw depth image and the resulting point cloud. Cropped image regions outside the reachable space are colored grey.

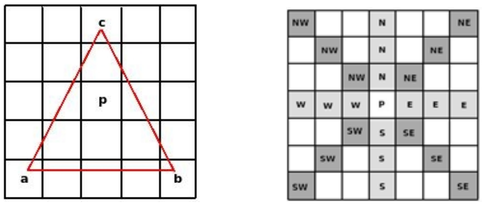

Fig. 2. Left: Three adjacent points used for normal calculation. Right: Neighboring pixels considered for detection of normal edges. Values from three pixels are averaged for each of eight directions.

a) Determination of Surface Normals: As the basis for computing "surface normal edges", we first determine surface normals for every image point. In order to reduce noise and to smooth surfaces, a 3×3 median filter is applied to the image. Although later processing steps of the algorithm robustly reduce noise as well, it is computationally more effective to apply the smoothing filter at this initial stage.

As the computation of normal edges is very robust w.r.t. noisy normal vectors, we can simply calculate surface normals from the plane spanned by three points as known by common 3D computer graphics. We also evaluated more accurate, but slower, calculation methods based on PCA, which did not provide better final segmentation results.

The three points are chosen in a distance of $r = 2$ in the vicinity of the central, considered 2D image point $\vec{p}$ as shown in Fig. 2. The determination of surface normals is directly done 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 results in more distinct edges. Using the cross product:

$$\vec{n} = (\vec{b} - \vec{a}) \times (\vec{c} - \vec{a}) \tag{1}$$

and subsequent normalization we easily obtain normals $\hat{n}$.

b) Detection of Surface Normal Edges: A major contribution of the present paper is the detection of surface normal edges, which is based on the computation of the angle between surface normals of adjacent image points, which can be efficiently done employing the scalar product:

$$cos(\angle(\hat{n}_1, \hat{n}_2)) = \hat{n}_1 \cdot \hat{n}_2 \,. \tag{2}$$

To obtain broad, uninterrupted edges suitable for the subsequently applied 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. To reduce the influence of the noisy normal calculation (1), along each direction we average the three scalar products obtained between the central and three adjacent pixels (cf. Fig. 2). For example, for the north-bound direction, we use:

$$cos(\theta_N) = \tfrac{1}{3} \textstyle\sum_{i=1}^{3} \hat{n}_{x,y} \cdot \hat{n}_{x,y+i} \tag{3}$$

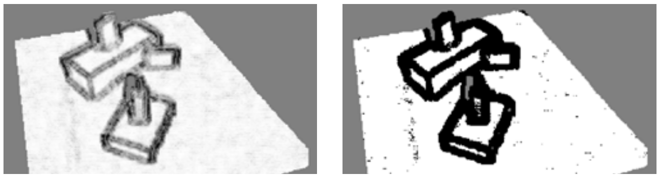

Fig. 3. Result of the surface normal detection applied to the depth image of Fig. 1 (left: angular values, right: binarized version). White surface patches are properly enclosed by black object edges.

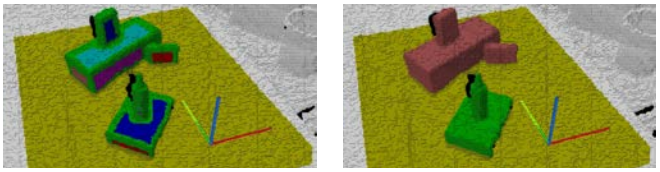

Fig. 4. Results of first segmentation into surface patches, object edges, and background pixels (left), and second segmentation into object blobs based on Euclidean distance (right).

The minimal value of these averaged scalar products is finally used as the outcome of the edge filter, which corresponds to the strongest angular deviation of normal vectors:

$$\min\{cos(\theta_N), cos(\theta_{NE}), \ldots, cos(\theta_W), cos(\theta_{NW})\} \quad (4)$$

Note, that we do not need to calculate actual angular values employing the costly arc cosine function, but directly can use the results of the scalar product. While large values, close to one, correspond to flat surfaces, smaller values indicate increasingly sharp object edges. Finally, binarizing the obtained edge image employing a threshold value $\theta_{max} = 0.85$ $(31, 8^\circ)$, we can easily separate edges from smooth faces. The threshold value is chosen to balance between noise and recognition of smooth *curved* surfaces.

Figure 3 illustrates the results 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 detected due to noise. However those regions are small and disjointed and thus can be easily filtered out in subsequent processing steps. Small or narrow objects are often represented by edges only.

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 surface patch as shown in Fig. 4.

The novel surface patch segmentation based on normal edges already provides a detailed segmentation of the scene into surface patches, which will be employed for the subsequent object segmentation step, introduced in the following.

B. High-Level Object Segmentation

In the second processing block, we ultimately aim for a segmentation on an object level, which means that the previously found surface patches need to be combined to form *object regions*. To this end, we start with the identification of supporting surfaces in order to separate the background from the foreground.

d) Identification of Support Planes: The background of table-top and other indoor scenes usually comprises large planes, e.g. table surfaces, walls, or the floor. As these background planes disturb the subsequent clustering of object regions, we first try to find them and exclude all their associated points from further consideration.

All face segments comprising more than $N_{bg} = 2500$ points are processed. A plane is fitted to each of these segments using RANSAC [12]. Because points are only sampled from previously found surface patches, i.e. probably a plane, only a very few RANSAC passes are required to find a good plane model.

Subsequently, all not yet processed points (that is all points in the first iteration) are associated to this plane segment if their distance to the plane is smaller than $\delta = 12mm$. If a point was part of another segment, it will be removed from this segment and reassigned to the plane segment and thus separated surface patches are combined. Notice, that RANSAC itself employs half the threshold to yield better fitting results. If the plane normal is parallel to the z-axis of the world coordinate frame (up to an angular threshold of $\alpha_z = 10^\circ$), this plane segment is considered as a horizontal supporting plane, which can be used to place objects. In all cases, the plane segment is marked as not graspable and all points belonging to this segment are marked as processed and are no longer considered for further processing. This method to identify background planes is iterated until no large face segments are left.

After this processing step, some face or edge segments may contain only a very few or no points at all. This can be due to initial noise (cf. Fig. 3) or due to the reassignment to background planes. Hence, segments comprising fewer than $N_{seg} = 5$ points, are removed and their remaining points are marked as processed.

e) Blob Segmentation: The next important step is the generation of potential object candidates standing out from the scene background. Having removed all background pixels in the previous processing step, nearby objects form smaller isolated point clouds in the scene. These different point blobs can be easily separated and identified by an additional region growing algorithm operating on the 2D depth image. However, the coherence measure used to decide whether a point should be included within a segment, is now the Euclidean distance of neighboring pixels. This distance is easily computed from the 3D position associated with each pixel in the depth image. Adjacent points with a distance smaller than $\Delta_{blob} = 10mm$ are grouped within a common object blob. This segmentation method correctly separates objects which are sufficiently separated in 3D space or separated by background pixels in image space as can be seen in Fig. 4. Blobs containing fewer than $N_{blob} = 50$ pixels are considered to be not graspable and are therefore removed. If no support plane could be found, e.g. due to a very cluttered table, one single "object" blob remains. This distance-based segmentation is common to most other state-of-the-art approaches [5], [8], [16], but cannot separate nearby objects.

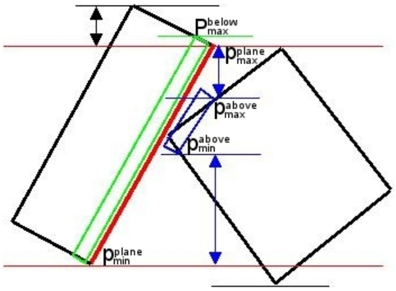

Fig. 5. Assigning separating plane using bounding boxes.

f) Binary Space Partitioning: In order to further separate stacked objects and objects standing in front of each other, we partition the detected object blobs employing the face segments computed in the first part of the algorithm. The central idea of this heuristic approach is that visible surface patches are part of an object's outer hull, such that points belonging to this object should either lie on the one or the other side of the associated plane. If we conversely find enough points on both sides of the plane, we assume two (or more) separated objects and split the blob into two blobs for further processing. Consider for example a bottle placed on top of a box as shown in Fig. 4. Three faces of the box are visible and the top face will be used to split the blob, because the associated plane separates both point sets.

As the plane fitting using RANSAC needs a sufficient number of face points, we first consider the number of edge pixels P_e and the number of face pixels P_f. If the ratio P_f/P_e is smaller than $k_{fe} = 0.1$ or P_f is smaller than $N_f = 30$, the blob is considered as a single object with no further subdivision taking place.

Otherwise, the largest face segment is selected and a plane is fitted to the segment's points. Subsequently *all* points of the blob are sorted into three categories based on their (signed) distance d to the plane: points on the plane ($[-\delta, \delta]$), points behind ($(-\infty, -\delta)$), and points in front ((δ, ∞)). If more than $N_{split} = 20$ points are on both sides of the plane, the blob is split into two parts, one comprising the points in front and the other comprising the points behind the plane. The threshold N_{split} is non-zero to tolerate a few outliers not perfectly fitted by the plane.

TABLE I

AV. AND MAX. FRAME-TO-FRAME ANGULAR VARIATION OF NORMALS

Method	Runtime	average	maximum
cross 3×3	30ms/<1ms	8.16°	43.60°
cross 5×5	30ms/<1ms	4.80°	24.39°
PCA 3×3	1674ms	7.15°	32.00°
PCA 5×5	2560ms	3.83°	14.35°

In order to decide to which of both sub blobs the separating plane can be attributed, we consider the bounding boxes of the plane and adjacent blob points. To this end, we again sort *all* points of the *original* blob into three groups (on/behind/infront plane), but this time the distance of points to be considered is limited to 2δ. This ensures that only a small slice of points close to the separating plane is considered for determination of bounding boxes. The bounding boxes (aligned to the world coordinate frame) of all three point sets are determined by two opposite points $\vec{p}^s_{min}$ and $\vec{p}^s_{max}$, where the superscript s=on/behind/infront indicates the point set. Comparing the bounding boxes of the plane to those of the "behind" respectively the "infront" point set, we can judge plane assignment. To this end we consider the distances:

$$E^s = \|\vec{p}^s_{min} - \vec{p}^{\text{plane}}_{min}\|_1 + \|\vec{p}^s_{max} - \vec{p}^{\text{plane}}_{max}\|_1 \tag{5}$$

where $\| \cdot \|_1$ denotes the L_1 norm. More similar bounding boxes will have a smaller distance value. Hence, if E^{behind} is smaller than E^{infront}, all plane points are assigned to the sub blob behind the plane and vice versa. Fig. 5 illustrates this assignment step in a 2D example. The need of the range reduction for the determination of the minima and maxima is obvious by the distances of the whole objects to the plane in contrast to the distances of the reduced spaces to the plane.

The newly created blob not containing the plane is added to the set of to-be-processed blobs, while the subdivision of the other blob comprising the plane is continued by looking for the second largest surface segment. This process is iterated for each blob, until no separating face segment can be found anymore. Then the process is repeated for all remaining blobs.

Finally, we obtain a scene segmentation, which not only distinguishes spatially separated objects (based on their Euclidean distance), but also nearby objects based on their planar convexity as shown in Fig. 1. Note, that all used parameters are tuned for a sensor distance of up to 3 meters, but can be easily adapted for other distances.

III. EVALUATION

In this section we report different evaluation results. Firstly, we compared various methods to estimate surface normals and to compute normal edges, both w.r.t. robustness and runtime performance. To this end, we evaluated the stability of results when observing a static scene, i.e. input images only influenced by sensor noise. Secondly, we describe the impact of all parameters onto the segmentation result and state their optimal value range. Thirdly, we evaluated the overall robustness of the segmentation results. Further we discuss weak spots of the method and report on a grasping experiment.

A. Normal Estimation and Edge Determination

An important part of our algorithm is the robust estimation of surface normals. We compared two algorithmic approaches, the proposed cross-product calculation and a PCA method, each using points from a 3×3 or 5×5 neighborhood. All methods calculate the normals directly based on the median-filtered depth image, which turned out to yield sharper normal edges in other experiments, we do not report on in detail.

Due to sensor noise, the input images also vary when observing a static scene. We evaluate, how these fluctuations (of size ± 1 in the filtered depth image at a range $[0..2047]$)

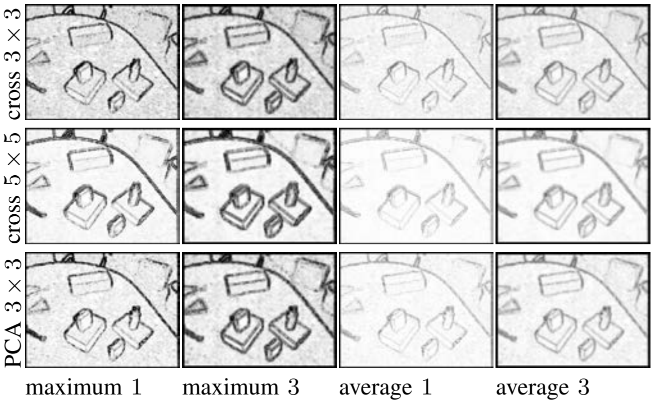

Fig. 6. Normal edges obtained for different methods and parameters.

are amplified by the algorithms to judge their robustness. To this end, we observed a 10×10 patch from the planar table surface for 50 frames and determined the angular deviation of surface normals from one frame to the next for all 100 points. Table I summarizes the results for both methods and parameter sets, showing the average and maximal deviation calculated over all points and frames, as well as the runtime. The PCA methods provide only little better robustness, while being much slower. Accordingly, we have chosen the cross-product calculation with a neighborhood radius of $r = 2$. Using a GPU-based parallel implementation, we could reduce the computation time even further, from 30 ms to less than 1 ms.

A similar evaluation w.r.t. robustness was done for different methods and parameters to compute surface normal edges. The previously proposed method employing the maximal angular change of normals along eight directions (eq. 4) is compared here to an averaging method yielding smoother edges. Again, we compare different neighborhood ranges, considering a single or three adjacent pixels per direction (cf. Fig. 2). The results are presented qualitatively in Fig. 6. Obviously, the averaging methods generate smoother edge images showing less noise. However, the maximum methods exhibit sharper edges and thus were finally selected for the algorithm. Variants using three adjacent pixels generate broader edges as required for the region growing method. Regarding noise, the cross-product-based calculation of normals with a radius of 2 achieve best results (middle row), even better than the much slower PCA method.

We also evaluated these methods using less directions for edge determination, e.g. only the north and the east direction. However, this resulted in disconnected edges, whose holes need to be closed by morphological operations, which do not better perform.

B. Parameter Selection

The meaning and significance of most parameters of our algorithm is obvious, and reasonable values can be found easily. The most important paramters are explained in the following and suitable value ranges are provided. All values were selected by manual inspection regarding the desired effect and the final segmentation result. The chosen parameters are optimized for a distance of up to 3 meters from the sensor, which is sufficient for table-top scenarios. The paramters can be easy adjusted for other ranges.

- $\theta_{max} = 0.85\ (31,8^\circ)$ is the binarization threshold for edge detection. Smaller values generate less noise, but edges may become holey. $\theta_{max} \in [0.75, 0.9]$. Caused by higher noise for higher distances, this parameter could be increased for higher distances.
- $\delta = 12mm$ is the distance threshold for RANSAC-based plane fitting. The more RANSAC passes, the smaller values are possible. Values in $[8, 25]$ are good for 7 passes. Small values can lead to unassigned points.
- $\Delta_{blob} = 10mm$ is the maximum distance for blob segmentation. Values in $[2, 50]$ are good, while smaller value cause over-segmentation. Under-segmentation can be fixed by the (more costly) space partitioning. For higher distances and thus higher point distances the paramter could be increased.
- $k_{fe} = 0.1$ is the minimal ratio between face and edge points. If the ratio gets small, most points are edge points and a decomposition using the faces is not applicable. $k_{fe} \in [0.05, 0.5]$.
- $N_{split} = 20$ is the minimal number of points on both sides of a plane to split a blob. A decomposition into too small parts is not desirable and could be caused by outliers of the plane fitting. $N_{split} \in [10, 30]$.

C. Runtime Performance and Overall Robustness

In this section we report on experiments to evaluate the overall robustness of segmentation results using the given parameter values. We consider 20 input images corresponding to a steady scene. However, sensor noise and non-deterministic elements of the algorithm may cause varying segmentation results over time. To evaluate robustness we again compare subsequent frames and calculate the percentage of image points associated to the same object region (segment overlap), as well as the average distance of object centroids. Furthermore we count the number of spurious segments, i.e. additional object regions.

In order to associate corresponding segments from frame to frame, we employ a tracking method from the Image Component Library[1], which employs the number of segment points, the centroid, and the bounding box to correctly identify segments. The determination of overlapping segment regions is performed for every segment. Values obtained for all segments and frames are averaged. Corresponding results are summarized in table II, where individual rows correspond to scenes of increasing complexity as shown in Fig. 7.

The first column (#seg) shows the number of expected object segments, including the table. The second and third columns report average and worst values for region overlap and centroid distance resp. Finally, the next-to-last column lists the absolute and relative number of spurious segments.

In most cases, the segmentation algorithm yields stable regions with only little reassociation of image points between

[1] http://www.iclcv.org/

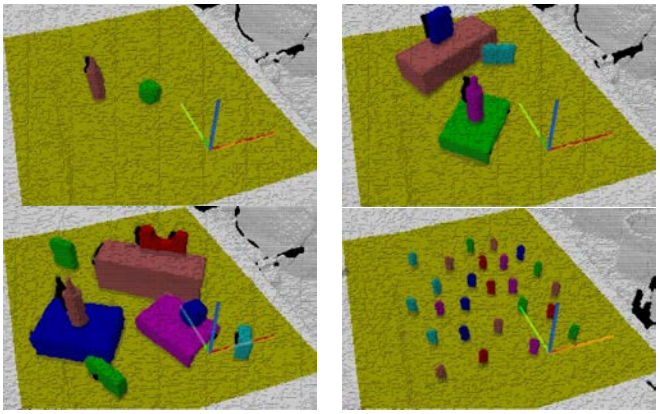

Fig. 7. Scenes of increasing complexity used for evaluation.

TABLE II

ROBUSTNESS OF SEGMENTATION RESULTS IN VARIOUS SCENES.

#seg	region overlap [%] avg / min	centroid distance [mm] avg / max	spurious segments abs / rel[%]	segmentation runtime [ms] faces + objects
3	99.9 / 96.0	1.2 / 3.9	0 / 0	5 + 41
6	99.5 / 92.9	2.3 / 8.7	0 / 0	5 + 832
10	98.6 / 82.7	2.2 / 21.0	5 / 2.5	5 + 3183
26	99.5 / 89.2	2.4 / 11.0	0 / 0	5 + 101

frames. However, the third row – corresponding to a complex scene with many stacked and nearby objects – reports five spurious object regions, evoking worse results in region overlap and centroid distance as well, usually caused by one additional false segment. Object centroids are stable enough for application in autonomous grasping scenarios.

The last column reports the run times of the initial segmentation into surface patches (sec. II-A) and the subsequent determination of individual objects based on binary space partitioning (sec. II-B). As can be seen, the runtime of the initial part is fixed – except for measuring noise. However, the computational effort of the subsequent object segmentation exponentially grows with the complexity of the scene. Comparing the last two rows, it becomes apparent, that the complexity is determined by the number of possible split faces and not by the number of initial object blobs. For evaluation we used a single-threaded version of the algorithm running on a single core of a XEON 2.53 GHz processor. The runtime of the anterior face-segmentation is 1 ms for an OpenCL implementation on a GTX 560 graphics card including median-filtering, normal-determination, edge-detection and binarization and additional 4 ms for the recursive region-growing. The single-threaded version needs 230 ms for comparison.

D. Strong and Weak Points

The algorithm works very well with a huge number of scenes with stacked and nearby objects and of course with seperated objects, even in scenes with complex, non-convex objects (e.g. plants), apart from simple table-top scenes. Those types of scenes are shown in Fig. 8. The whole segmentation is model-free and without previous knowledge about the objects. But there are two weak points remaining, which are explained in the following.

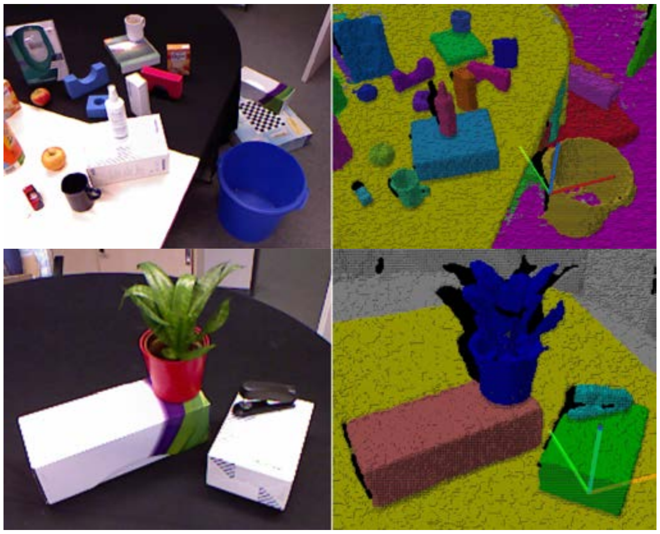

Fig. 8. Top: A complex scene with multiple support planes and non-convex objects. Bottom: A scene with a complex object (plant), segmented model-free and without previous knowledge about this object

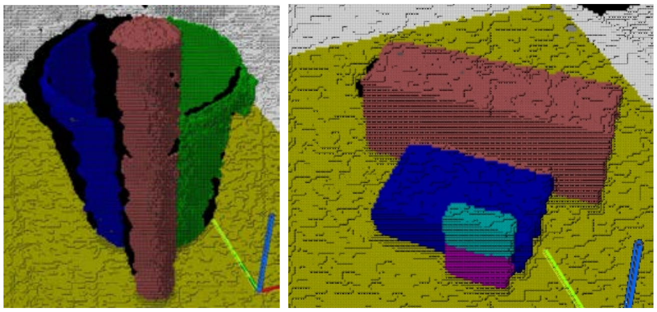

Fig. 9. Left: The bin is separated caused by planar model applied to curved surface. Right: Small box splitted due to wrong order of split faces.

Firstly, the binary space partitioning is currently restricted to planar surface models. Although the initial segmentation step can yield smooth curved surface patches, subsequent separation of object blobs fits a plane into these face segments. This works well, if the faces are planar like the surfaces of a box. Even if a curved object is in front of another the separation works well, but not if two curved object are directly side-by-side or if one object completely splits another object as shown in left part of Fig. 9. This issue can be solved by determination of curvature coefficients to describe curved surfaces. However, note that such situations are extremely rare in real-world scenarios.

Secondly, the order of selecting split faces (currently descending by size) is of importance if no additional measures are taken. Choosing the wrong order, may cause an object to be decomposed into two segments as shown on the right of Fig. 9. Here, the upper surface of the mid-size box was chosen for splitting before its front surface, resulting in a significant amount of pixels on both sides of the plane. In the next version of the algorithm we solve this issue by prohibiting already found surface patches to be split.

E. Grasping Experiment

We used the segmentation approach for autonomous grasping with the 24-DOF Shadow Robot Hand employing our biologically inspired grasping strategy [6]. 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 [13] to the 3D points of the selected object blob. This model determines 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 a simple PCA fitting 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. The experiment is shown in the accompanying video[2] and in Fig. 10. For the shown experiment we used a simple pointing gesture detection which calculates a ray through the arm of the user towards the task space. Thus, a human user can interactively select objects in the scene which are then grasped by the robot hand.

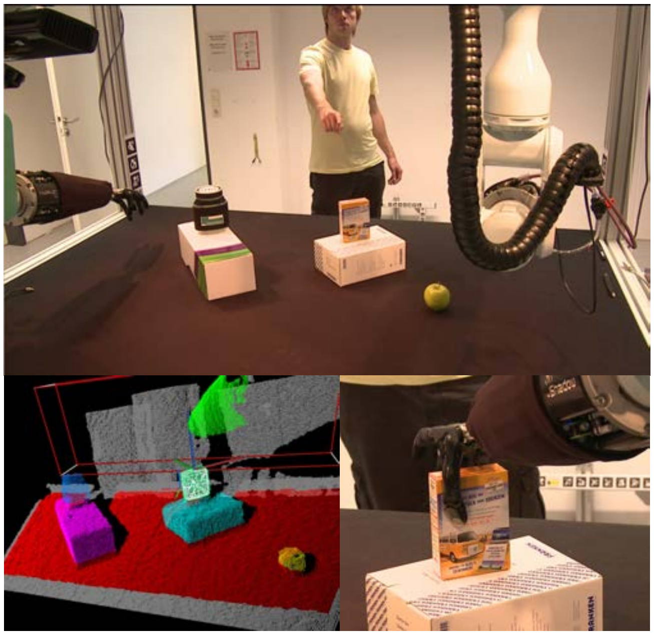

Fig. 10. Top: the setup of the grasping experiment and pointing in the scene. Bottom left: arm movement to the selected object in the point cloud. Bottom right: grasping the object.

IV. CONCLUSION AND FUTURE WORK

In this paper, we introduced a model-free segmentation algorithm for cluttered scenes with stacked and nearby objects which is not restricted by a given set of object models. Surface-normal-edge-detection for surface determination using edge detection on point normals was combined with RANSAC, Euclidean segmentation and binary space partitioning to decompose the scene into individual objects. The pre-segmentation into surface patches has the strong advantage, that randomly sampled points for RANSAC, already originate from a uniform surface, thus reducing computational effort. The algorithm can deal with stacked, nearby and partially occluded objects as well as with complex objects which could not be segmented by model-based approaches, e.g. the plant in Fig. 8. This is due to the novel binary space partitioning method to separate point cloud blobs of nearby objects. The algorithm was evaluated concerning robustness and quality.

To allow direct interaction with users, a segmentation of objects from the human hand is desirable. To this end, color histograms could be used additionally. This color information is also applicable to the blob decompositions as an additional factor in the binary space partitioning but involves the danger of over segmentation in the case of highly textured objects. The segmentation of the scene is an initial hypothesis of the scene structure. This can be refined by active exploration. At the moment, the second part of the algorithm is implemented straight forward without any optimization or parallelization. This will be done to improve runtime performance and thus to achieve real time capabilities.

[2] http://www.youtube.com/watch?v=xJjs-q1QCcQ

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] 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

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

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

[9] 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

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

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

[12] 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

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

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

[15] 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

[16] Z.C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, M. Beetz, General 3D Modelling of Novel Objects from a Single View, *Proc. IROS*, 2010

[17] Z.C. Marton, R.B. Rusu, D. Jain, U. Klang, M. Beetz, Probabilistic Categorization of Kitchen Objects in Table Settings with Composite Sensor, *Proc. IROS*, 2009