2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Realtime 3D Segmentation for Human-Robot Interaction

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 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. Coplanarity and curvature matching is used to recombine surfaces separated by occlusion.**

The real-time capabilities are proven and the quality of the algorithm is evaluated on a benchmark database. Advantages compared to existing approaches as well as weaknesses are discussed.

I. INTRODUCTION

Realtime segmentation is one of the most important tasks in visual perception for real-world robot interaction. For example, robust shape and pose estimation of individual objects is a major prerequisite for automomous grasping from a pile of unknown objects [1], [2].

Many of the current state-of-the-art segmentation algorithms can be classified into two categories: (1) simultaneous recognition and segmentation of learned objects and (2) support plane extraction and model-fitting.

For simultaneous recognition and segmentation, found image features are matched to a database of known objects. Various feature extraction methods have been proposed, including 3D-augmented SIFT features [3], [4] and features directly obtained from range images such as a viewpoint feature histogram [5], depth-encoded hough voting [6], point pair features [7], and iterative clustering-estimation [8]. 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 previously learned objects.

In [9], [10], [11], [12] 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. While [9] focuses on cylindric and box-like objects for grasp planning, in [10] subsequently 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 [11] is tuned towards real-time performance, achieving frame rates of 30Hz on images sized 160×120 for plane segmentation. [12] adds support for arbitrary rotational surfaces. This class of algorithms is good for spatially separated objects but becomes slow if object blobs must be decomposed by costly model-fitting operations.

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

Other approaches using multi sensor fusion [13], active exploration [4], [14], depth augmented color segmentation [15] or relation learning algorithms [16], [17]. In [13] 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. [4], [14] improve the segmentation by active exploration. While [14] uses an active camera system to explore the scene, [4] manipulates the objects in the scene. A spatio-temporal depth-supported color segmentation of video streams is presented in [15]. The algorithm segments object surfaces und tracks object movement in realtime, but does not combine found surfaces to an object hypothesis. Instead of learning individual object models, [16] employ learned relations of planes to recognize objects like boxes or stairs. Again, training is necessary and problems with cluttered scenes with occlusion can emerge. The combined approach in [17], [18] first executes a presegmentation and surface detection using NURBS and subsequently groups the found patches using a SVM, trained with complex object relations. This algorithm yields very good segmentation results, however, at the cost of a high computational effort (1-8s per frame), which (currently) disqualifies for real-time application. The fixation-based segmentation approach presented in [19], attempts to determine the overall object boundary combining depth and color cues. However, while the idea of extracting object edges arised simultaneously to our work [20], this approach cannot deal with complex scenes of stacked objects.

In the present work we extend our model-free and real-time capable segmentation approach presented in [20], [21] to a general probabilistic framework, which considers multi-modal cues in a uniform manner. The algorithm combines two segmentation methods: the identification of smooth object surfaces and the composition of these surfaces into sensi-

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

ble object hypotheses. While the idea of the pre-segmentation introduced in [20] remains the same, the algorithm was refined in [21] by adding smoothing and filtering resulting in thinner edges and by parallelizing the algorithm. In this work, we replace region growing by connected component analysis and implement motion sensitive temporal smoothing to avoid the motion blur effect of our previous work. While the high-level segmentation of our first work extracted support planes and decomposed the remaining blobs using binary space partitioning, the second contribution introduced the idea of composing cutfree neighboring surfaces. In contrast to our last work which used a greedy composition considering adjacency only, in the current work we are employing graph-cut on a probabilistically weighted similarity graph considering adjacency, curvature and coplanarity of found surface patches to enable the method to handle occluded and open curved objects. Additionally, the algorithms are further optimized for real-time challenges.

The main advantage of our method, in contrast to existing ones, is the capability to separate unknown, stacked, nearby, and occluded objects in a model-free manner as shown in Fig. 1. Naturally, this approach has its limitations compared to model-based approaches, especially if very complex object shapes are to be considered. However, it provides a meaningful initial object hypothesis in arbitrary situations, which can be refined by active exploration [4] or fed as input to model-based adaptive methods. The probabilistic nature of our method allows to focus these methods to selectively disambiguate uncertain object hypotheses.

The algorithm operates in real-time facilitating interactive usage in human-robot-cooperation tasks.

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. 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 [22], 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 for now, and thus diminishing the impeding influence of strong textures giving rise to oversegmentation. 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.

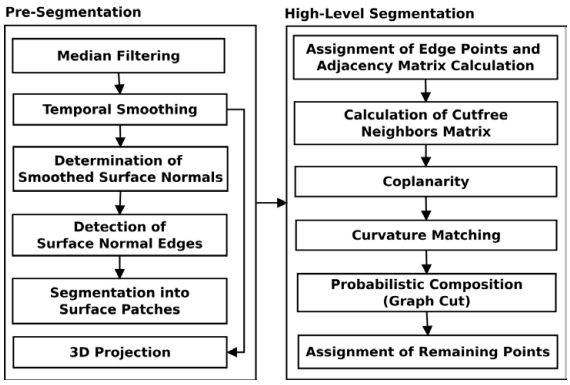

Fig. 2. Overview of the segmentation algorithm.

The second part of the algorithm subsequently combines found surface patches into meaningful object segments. To this end, we create a weighted graph describing the adjacency-, coplanarity- and curvature-relations between found surfaces. This graph is decomposed using a graph cut algorithm to achieve highly probable object hypotheses.

Figure 2 illustrates the structure of the segmentation algorithm.

A. Pre-Segmentation

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. In contrast to our previous publication [21], we changed the temporal smoothing to be motion sensitive and replaced the region growing algorithm by a faster connected component analysis algorithm.

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 [20]. 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 motion-sensitive temporal smoothing, averaging depth values of all

individual image pixels within the last $n = 6$ frames, if the difference of the depth values is smaller than $d = 10$. Finally, the calculated normals are smoothed applying a convolution using a 5×5 Gaussian kernel. Fig. 3 shows the resulting surface normals for the image in Fig. 1, mapping xyz normal directions to the RGB color space.

In contrast to our previous method, we now restrict temporal smoothing to points moderately varying over time, thus effectively avoiding motion blur, while tremendously reducing sensor noise if no motion is present.

b) Detection of Surface Normal Edges: The next step 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 fast connected component analysis algorithm, replacing the region growing from our previous work, 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*. We determine a weighted graph, modeling the probability of two surface patches belonging to the same object region. Subsequently this graph structure is analyzed to find the most probable segmentation into object regions using a graph cut algorithm. As an extension to our previous work [21], we also employ co-planarity and curvature cues to successfully combine objects patches which are separated due to occlusion.

d) Adjacency Matrix and Assignment of Edge Points: An initial adjacency matrix representing the basic connectivity of surface patches is determined as follows: For every edge point p_r all neighboring surface points p_i within a radius r in image space are considered, which have an Euclidean distance $\|p_r - p_i\|$ smaller than a threshold $d_{\max}$. All possible surface pairs obtained from this list are marked as adjacent. For example, consider Fig. 5: Here surfaces 2 and 10 are neighbors in image space, but not in 3D space and therefore aren't considered adjacent. Faces 7,9,11 fulfill the conditions and become connected in the graph.

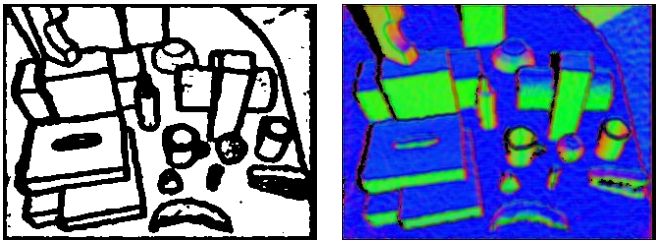

Fig. 3. Left: Result of the edge detection (binarized version). White surface patches are properly enclosed by black, uninterrupted object edges. Right: Point normals (XYZ direction mapped onto RGB colors).

Simultaneously, the edge point p_r is assigned to the nearest surface patch – measured in Euclidean space (cf. Fig. 4). Edge points, which are part of very bold edges or are too distant from neighboring surfaces, such that no adjacent surface points are found, are not yet assigned to a face. These points probably belong to a separate, but small object, and are processed in a later stage. If the search radius r and the distance threshold $d_{\max}$ were not restricted, such small objects would be absorbed by their supporting surface.

e) Cutfree Neighbors: To further improve the adjacency matrix, we apply a plausible heuristic check already proposed in our previous work [20]: 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 4 and 12 in Fig. 5. While all points of face 4 are on top of the supporting face 12, the plane fitted into surface 4 cuts surface 12. Hence this surface combination is disregarded. On the other hand surfaces 7,9,11 are all pairwise cut-free.

To speed up the cutting test, we approximate surfaces by a set of 20 planes fitted using parallelized RANSAC [23]. 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 topology of neighboring surfaces, modeled by a symmetric, boolean adjacency matrix.

f) Improving the Adjacency Matrix: In case of occlusion, a single face of an object is separated into two parts, which will not have a link in the adjacency matrix. Consequently, our previous approach was not able to properly recombine those parts into a single object hypothesis [21]. To overcome this limitation, we extend our approach and add further links to the matrix based on additional cues,

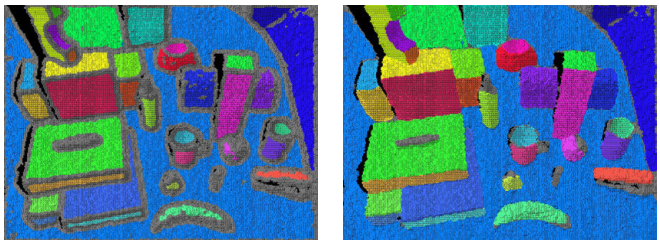

Fig. 4. Results of the first segmentation into surface patches and edges (left), and after assignment of edge points to closest surfaces (right).

namely co-planarity of flat faces and similar curvature of curved surfaces.

In both cases, we have to ensure, that surfaces became separated by occlusion. For illustration, consider Fig. 5. Here surfaces 3 and 8 as well as 4 and 7 are coplanar. However, only the former pair should be considered for combination, because they are separated by occlusion, while the other pair is separated by background. In order to check that two surface patches are separated by occlusion, but not background, we check whether image points in the area between the two surfaces are closer to the camera or not, i.e. have smaller depth value than expected or not. This condition is checked along selected lines connecting points from both surface patches. The predicted depth values along a line are calculated using linear interpolation.

Two flat surfaces are considered co-planar, if they span a common plane. To assess this condition, we first compute the mean normal and its variance for each surface. A face is considered flat resp. curved, if the variance – measured as the mean angular deviation of all surface normals from their mean – becomes small resp. large.

g) Co-Planarity: To check for co-planarity of two *flat* surfaces we proceed in two steps: If both surfaces have similar mean normals (up to a small noise margin), we check whether the faces are aligned, i.e. indeed span a *common plane*. In this case, any plane spanned by three points from both surfaces should have a similar normal as the two original mean normals. Because the normal of the spanned plane may crucially depend on the actual selection of points, this criterion is checked for a set of 50 randomly selected triples of points. If any of the calculated normals deviate too much, the two surfaces are not considered co-planar. Otherwise, the above described occlusion check is carried out, along several lines connecting two randomly selected points from both surfaces. If this check is passed as well, a corresponding link in the connectivity matrix is added for the given pair of surfaces. Figure 5 shows the resulting graphs before and after co-planarity extension. While the first graph results in four final objects, the second graph correctly results in three objects.

h) Curvature Matching: In order to handle curved surface patches in a similar fashion, we compare their curvatures. To this end, we compute a *curvature histogram* for every curved surface, representing the distribution of surface normals within the surface. The 2D histogram of 11×11 bins describes the relative frequency of observing surface

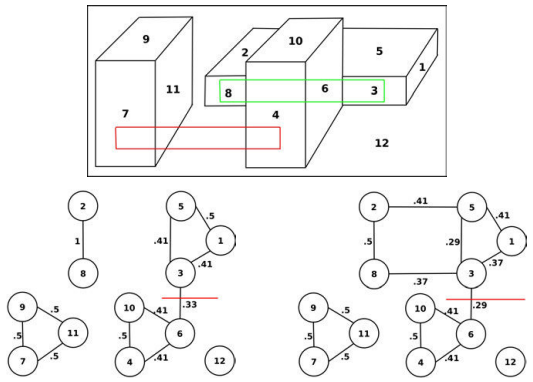

Fig. 5. Illustration of the coplanarity and the probabilistic composition. Coplanar surfaces separated by occlusion (green bounding box) shall be combined, coplanar surfaces separated by background (red bounding box) shall not be combined. Left graph without coplanarity (occluded box is separated into two parts), right graph with coplanarity (correct composition of the two box parts). Red line in the graph illustrates the cut in the subgraph.

normals with given x and y components. Note, that the associated frequency of z components is determined by the fact, that normals are normalized to magnitude one. As can be seen from Fig. 6 these histograms are compact fingerprints of the shape and orientation of the corresponding surface patches. While the first two, very similar histograms belong to the two separated pieces of the lying cylinder, the other two histograms belong to two different objects: the standing bottle and the sphere. To measure the distance of histograms we estimate the mutual overlap of their distributions:

$$\mathrm{D}(A,B) = \textstyle\sum_{ij} |a_{ij} - b_{ij}| \qquad (1)$$

Exploiting the normalization of histograms and the fact:

$$\min(a,b) = \tfrac{1}{2}(a+b) - \tfrac{1}{2}|a-b| \qquad (2)$$

we more efficiently compute the similarity index:

$$S(A,B) = 1 - \tfrac{1}{2}D(A,B) = \textstyle\sum_{ij} \min(a_{ij}, b_{ij}) \qquad (3)$$

which is a score between 0 and 1. Surface pairs with a score larger than $\Theta_h = 0.5$ are considered for recombination.

In the following, we differentiate between *open curved* objects and *occluded curved* objects. To recombine the inner and outer surfaces of an *open* object (like a cup or bowl), two conditions must be fulfilled: (1) both surfaces are neighboring in image space and (2) the surfaces are concave and convex respectively. The first condition considers the fact, that the calculation of the initial adjacency matrix is restricted to neighbored surfaces in Euclidean space. However, the inner and outer surface of a cup, are typically quite distant in Euclidean space and thus became separated in our previous work [21]. The second condition simply checks for presence of an *open* object.

To assess the convexity / concavity of a surface, we again consider the curvature histogram, namely the two extremal bins $h_{\min}$ and $h_{\max}$ along the major axis of the histogram blob. Back-projecting these bins into the image space, we yield point sets $P_{\min}$ and $P_{\max}$, whose normals are mapped onto the corresponding bins. These point sets typically are

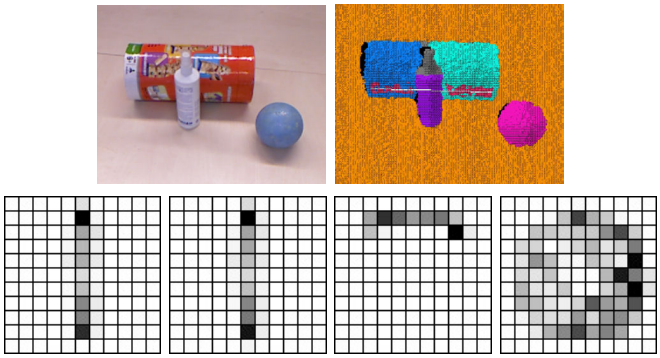

Fig. 6. Normal histograms of four surfaces: The first two histograms belong to the two parts of the lying cylinder, the other two to the standing bottle and the sphere. The segmentation image also emphasizes all image points whose normals are mapped to the peak bin in the histogram as well as the fitted line employed for the alignment and occlusion tests.

located at the boundary of the surface patch (assuming smoothly curved objects). In case of cylinders they form lines along its axis, in case of spherical objects they form smaller regions (cf. Fig. 7).

To proceed, we compute the mean image coordinates $\bar{p}_{\min}$ and $\bar{p}_{\max}$ of these point sets. The crucial point now is, that $\bar{p}_{\min}$ and $\bar{p}_{\max}$ are located on different object sides depending on the convexity of the object as illustrated in the rightmost figure of Fig. 7. Accordingly, we can assess convexity by considering the scalar product

$$(\bar{p}_{\max} - \bar{p}_{\min}) \cdot (h_{\max} - h_{\min}) \tag{4}$$

between the directional vectors formed by the extremal points in image vs. histogram coordinates. If this value is positive, i.e. both vectors pointing into a similar direction, the surface is convex, otherwise it is concave.

Secondly, we consider the case of *non-neighboring, similarly curved* surfaces, which are separated by occlusion. As for co-planarity we have to check for similarity and alignment of both surface patches as well as for occlusion. Similarity is again assessed using the similarity index Eq. (3). To judge alignment, we consider the bins with maximal density in both histograms. Back-projecting into the image space, we yield a common point set P describing the major axis of the object as illustrated in Fig. 6 for the two parts of the cylinder. If we fit a line through these 3D points using linear regression, we can exploit the regression coefficient as a quality measure of alignment of both surfaces. Finally, the occlusion check as described above is executed along the fitted line.

i) Probabilistic Object Composition (Graph Cut): The result of the previous steps is an adjacency matrix representing a graph with edges for all possible surface combinations arising from cutfree neighborhood, co-planarity and curvature matching. This graph is turned into a weighted graph, such that edge weights represent the strength of connectivity between two connected nodes. Finally, the graph is partitioned into sub graphs of high internal connectivity using a graph cut algorithm. These sub graphs represent the final object hypotheses. In contrast to our previous work [21]

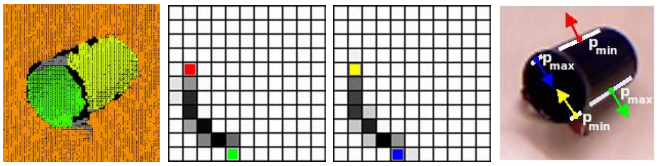

Fig. 7. Curvature histograms of the inner and outer surfaces of a cup. The rightmost subfigure visualizes the points contributing to the extremal bins in the histogram blobs, their means $\bar{p}_{\min}$, $\bar{p}_{\max}$ and the associated surface normals. As can be seen from this figure, the directional vector from $\bar{p}_{\max}$ to $\bar{p}_{\min}$ has opposite directions for convex vs. concave surfaces.

where we used a greedy strategy to compose complete sub graphs, the graph cut algorithm achieves better composition results, especially on partially occluded objects.

To determine the connectivity weights, we initially assign a common weight $w_{ij} = 1/n$ to all edges (i, j) originating from node i. Here, n denotes the number of nodes adjacent to node i. This results in a directed graph, where all outgoing edges of a node have the same weight and thus the same probability for composition with this node. To create an undirected graph, we average the weights of incoming and outgoing edges:

$$W_{sym} = \tfrac{1}{2}(W + W^t).$$

Figure 5 shows the resulting graph for an example scene. The higher the connectivity of two nodes, the higher their connecting weight. Graphs obtained without/with considering co-planarity are shown.

Exploiting the weighted graph, we apply a very simple graph cut algorithm. In contrast to the popular min-cut algorithm we do not need to find the minimal cut, but simply all cuts smaller than a given threshold $\Theta_c = 0.5$. This threshold balances between under- and over-segmentation. A very small value, close to zero, generates a single segment for every initially connected subgraph, while a very high value generates an individual segment for every surface node. The threshold of 0.5 achieves the optimal segmentation for most of the scenes, while an adjustment of the threshold allows to balance the granularity of the segmentation.

Starting with individual nodes, the algorithm calculates all connected sub graphs in ascending size and their corresponding cuts. A cut is the set of all edges outgoing from the sub graph and the associated costs is the sum of the corresponding edge weights. If the costs are smaller than Θ_c, a cut is found and the sub graph is extracted as a single object. If the subgraph exceeds $n/2$ in size, the algorithm aborts, because all potential cuts were considered.

In the example of Fig. 5 five segments on the left resp. four segments on the right are found. The partitioning cut is illustrated with a red line.

j) Remaining Edge Points: In the final processing step, all remaining edge points 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:

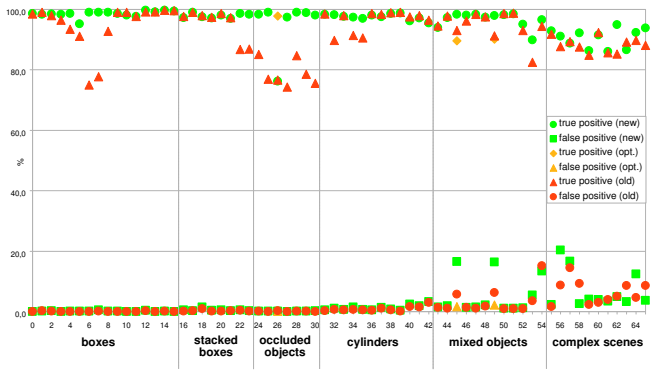

Fig. 8. Evaluation results of the Object Segmentation Database [24]. Comparison of our previous work [21] (red), the current work (green) and selected test-sets with optimized parameters (orange).

- 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 single 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.

III. EVALUATION

A. Quantitative Results

In this section, we present quantitative segmentation results for data taken from the Object Segmentation Database [24], 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 positives). $FP_i = S_i \setminus TP_i$ 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

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

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

The results of all test scenes in the database are illustrated in Fig. 8. The graph shows the true positive and false positive scores obtained from our previous (red) and the presented (green) algorithm. The false negative score is not illustrated in the graph, because it is simply $1 - tp$. Table I shows the mean scores and standard deviations obtained by averaging over all test scenes in the database. Three test scenes were segmented with an optimized cut threshold Θ_c as discussed below. All other scenes were segmented with the standard value of $\Theta_c = \frac{1}{2}$.

TABLE I
QUANTITATIVE SEGMENTATION RESULTS (IN %).

	true positive	false positive	false negative
old algorithm			
Mean	92.3	1.9	7.7
Std. deviation	7.3	3.3	7.3
new algorithm			
Mean	96.3	2.5	3.7
Std. deviation	4.1	4.5	4.1
optimized parameters			
Mean	96.4	2.0	3.6
Std. deviation	3.5	3.8	3.5

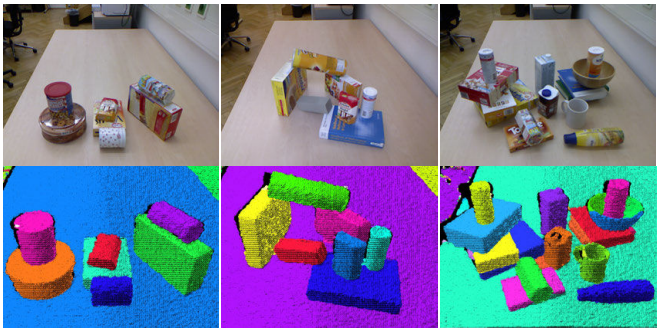

Fig. 9. Segmentation results for scenes 46, 50 and 65 of the OSD [24]

The true positive score of the new algorithm is noticeable higher than the score from the old algorithm, especially for highly occluded scenes in the database. The new algorithm has a slightly higher tendency to under-segmentation (false positive) in very complex scenes, what can be compensated by an adjustment of Θ_c. More segmentation results are available at the project website [25]. Figure 9 shows three example scenes from the Object Segmentation Database and the segmentation results of the introduced approach.

In [18] a correlation-learning based algorithm is evaluated against the OSD using one ore two SVMs. While with one SVM, the algorithm is slightly weaker than our algorithm in both, under- and over-segmentation, two SVMs let the algorithm slightly outperform our approach in the true positive score, but with the cost of much higher under-segmentation. In addition, our algorithm does not need prior training and is real-time capable. Table II shows the comparison of this algorithm with the presented approach.

B. Runtime Performance

In this section we evaluate the efficiency of the proposed approach, report on the runtimes of the algorithm and show some qualitative results. Fig. 10 shows three example scenes. The algorithm robustly segments table top scenes as well as

TABLE II

COMPARISON OF [18] AND OUR APPROACH (IN %).

	current work				[18]			
			opt.		one SVM		two SVMs	
	tp	fp	tp	fp	tp	fp	tp	fp
Boxes	98.6	0.1	98.6	0.1	96.5	0.4	97.8	2.3
St. boxes	97.9	0.6	97.9	0.6	97.7	4.6	99.3	4.7
Occl. obj.	95.3	0.2	98.4	0.2	81.4	1.5	94.1	1.0
Cylinders	97.5	1.3	97.5	1.3	97.8	0.8	98.6	12.7
Mixed	96.6	5.3	95.2	2.9	97.7	5.7	98.2	7.9
Complex	90.6	7.1	90.6	7.1	93.0	4.9	93.8	4.9
Total	96.3	2.5	96.4	2.0	94.9	2.8	97.2	5.8
Std. dev.	4.1	4.5	3.5	3.8	7.2	5.6	4.8	10.3

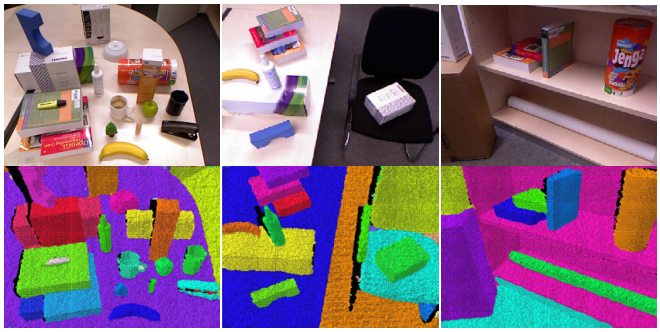

Fig. 10. Three complex scenes and their corresponding segmentation result.

other indoor environments, e.g. offices. It is applicable to scenes of different complexity with objects of various size and shape. Table III shows the runtimes for these scenes.

The runtime is composed of the motion sensitive temporal 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, coplanarity check, curvature matching, graph cut, the assignment of the remaining points, and the point cloud visualization.

The determination of the adjacency matrix, the assignment of remaining edge points, the coplanarity and curvature check, the graph cut and the cut-free testing vary with the complexity of the scene, whereby the framerate is at least

TABLE III

RUNTIME (MS) OF ALGORITHMIC STAGES FOR SCENES SHOWN IN FIG. 10.

	scene 1	scene 2	scene 3
temporal smoothing	0.5	0.5	0.5
edge detection	1.1	1.1	1.1
region growing	1.4	1.4	1.4
pointcloud	1.2	1.2	1.2
point assignment	7.5	6.7	10.5
cutfree pairs	3.2	2.2	3.8
coplanarity	3.2	2.4	2.1
curvature	8.2	2.8	3.9
graphcut	0.7	0.6	5.2
remaining points	4.2	2.5	2.4
visualization	2.5	2.5	2.5
overall time (Hz)	33.7 (29.7)	23.9 (41.8*)	34.6 (28.9)

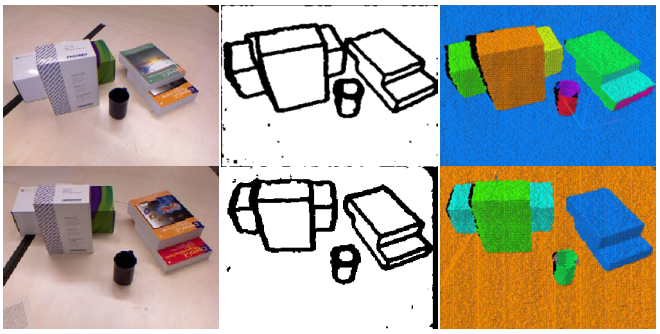

Fig. 11. Weak points of the previous algorithm: (1) open curved objects, (2) object split by complete occlusion, (3) staicase, and the correct segmentation of the current approach.

28 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. Faster hardware can ensure full 30 Hz, even on highly complex scenes. The algorithm always operates on QVGA resolution (320×240).

C. Strengths and Weaknesses

The presented real-time segmentation algorithm works very well with a huge number of complex scenes and solves some issues of our previous work [21]. These problems are illustrated in Fig. 11 in comparison with the segmentation results of the current approach. Firstly, the inner and outer part of open, curved containers, like cups, were decomposed into separate object regions. Using curvature matching these surfaces are now correctly recombined.

Secondly, surfaces which were separated due to occlusion by another object, were not recombined by the previous approach. In the current work, we add edges to the adjacency graph using co-planarity checks and thus are capable to recombine occluded object parts.

Finally, objects, whose surfaces are perfectly aligned (like steps of a staircase), were separated into multiple segments in the previous approach but in a strange manner. Even if the two books should be separated (what is hard without world knowledge) the segmentation as a single staircase-like object is to favor.

The two thresholds Θ_c for the composition using the graph cut algorithm and Θ_h for curvature matching yield correct results with their default value of 0.5 for most of the scenes. However, in some cases, an adjustment increases the segmentation quality. Figure 12 shows three OSD scenes, which were treated with specially tuned parameters. The resulting scores are visualized in Fig. 8 with orange markers.

The bottle in the first image remains decomposed using $\Theta_h = 0.5$, because the two surfaces parts are considered as non-aligned. By decreasing the matching threshold Θ_h, the bottle is correctly recombined. The two other images show a cluttered scene, where a box is separated into multiple small surface patches through occlusion. The result is a highly connected graph with cut costs higher than $\Theta_c = 0.5$. By increasing the threshold, the under-segmented part turns into an over-segmented part, which is preferable in this case.

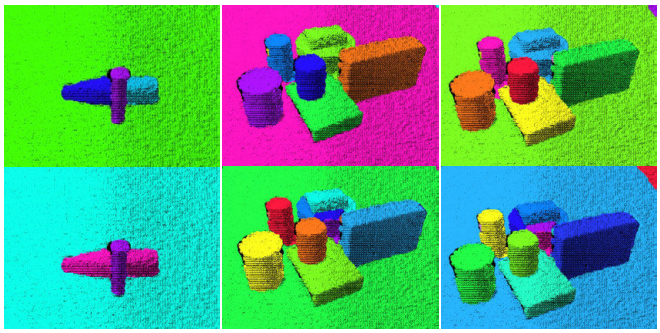

Fig. 12. Parameter optimization for better segmentation: (1) decrease Θ_h due to different surface orientations, (2)+(3) increase Θ_c due to under-segmentation. The examples are the test scenes 26, 45 and 49 of the OSD [24], marked as opt. in figure 8.

D. Human-Robot Interaction

We used the segmentation approach for autonomous grasping with the 24-DOF Shadow Robot Hand employing our biologically inspired grasping strategy [1]. 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 [26] to the 3D points of a selected object blob obtained 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. In the human-robot interaction experiment, shown in the accompanying video, the user selects an object with a pointing gesture. The robot grasps the object and hands it over to the human.

IV. CONCLUSION AND FUTURE WORK

In this paper, we extended our 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. Coplanarity checks and curvature matching were added to handle occluded and open curved objects. A graph cut algorithm for more flexible composition replaced the greedy strategy of our previous work and the runtimes were tuned towards the maximal framerate of the sensor employing optimized parallel algorithms. The algorithm can deal with stacked, nearby, and occluded objects, which is achieved by finding object edges in depth images and the novel idea to identify adjacent and cut-free surface patches, as well as coplanar surfaces, separated by occlusion, which can be combined 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. Employing color matching algorithms to enrich the adjacency matrix by similarly colored surface connections turned out to be counter-productive. However, color can still be a useful cue for the assignment of edge points and the separation of the human hand from objects. 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 [4], or model-rich approaches.

REFERENCES

[1] 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

[2] 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

[3] 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

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

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

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

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

[8] 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

[9] L.C. Goron, Z.C. Marton, G. Lazea, M. Beetz, Robustly Segmenting Cylindrical and Box-like Objects in Cluttered Scenes using Depth Cameras, *Proc. ROBOTIK*, 2012

[10] 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

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

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

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

[14] L. Nalpantidis, M. Björkman, D. Kragic, YES-YEt another object Segmentation: exploiting camera movement, *Proc. IROS*, 2012

[15] A. Albramov, K. Pauwels, J. Papon, F. Wörgötter, B. Dellen, Depth-supported real-time video segmentation with the Kinect, *Proc. WACV*, 2012

[16] R. Farid, C. Sammut, A Relational Approach to Plane-based Object Categorization, *RSS Workshop*, 2012

[17] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, M. Vincze, Segmentation of Unknown Objects in Indoor Environments, *Proc. IROS*, 2012

[18] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, M. Vincze, Learning of Perceptual Grouping for Object Segmentation on RGB-D Data, *to appear in Journal of Visual Communication and Image Representation (JVCI)*, 2013

[19] A.K. Mishra, A. Shrivastava, Y. Aloimonos, Segmenting Simple Objects Using RGB-D, *Proc. ICRA*, 2012

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

[21] A. Ückermann, R. Haschke, H. Ritter, Real-Time 3D Segmentation of Cluttered Scenes for Robot Grasping, *Proc. Humanoids*, 2012

[22] R.B. Rusu, S. Cousins, 3D is here: Point Cloud Library (PCL), *Proc. ICRA*, 2011

[23] 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

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

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

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