Real-Time Hierarchical Scene Segmentation and Classification

Andre Ückermann, Christof Elbrechter, Robert Haschke and Helge Ritter

Abstract—We present an extension to our previously reported real-time scene segmentation approach which generates a complete hierarchy of segmentation hypotheses. An object classifier traverses the hypotheses tree in a top-down manner, returning good object hypotheses and thus helping to select the correct level of abstraction for segmentation and avoiding over- and under-segmentation. Combining model-free, bottom-up segmentation results with trained, top-down classification results, our approach improves both classification and segmentation results. It allows for identification of object parts and complete objects (e.g., a mug composed from the handle and its inner and outer surfaces) in a uniform and scalable framework.

We discuss its advantages compared to existing approaches and present qualitative results. Finally, the approach is applied in an interactive robotics scenario to help the robot grasp objects in response to verbal commands.

I. INTRODUCTION

Real-time scene segmentation and object tracking are important tasks in real-world human-robot-interaction involving dynamically changing environments. Example tasks include online collision checking, ensuring that human collaborators are not harmed, grasping and manipulation of moving objects, or pointing gestures to reference specific objects in the scene.

Despite its importance, real-time capable approaches to generic and robust scene segmentation are scarce. Existing work can be classified by the degree of world knowledge presupposed. At one extreme, there are approaches that extensively employ previously acquired object models, to realize a simultaneous identification and localization of known objects. For matching, various features have been proposed, including 3D-augmented SIFT [1], viewpoint feature histogram [2], depth-encoded Hough voting [3], point pair features [4], iterative clustering-estimation [5], and deformable models trained on silhouettes [6]. These approaches robustly recognize partially occluded objects and correctly estimate their pose – if object models are available.

In order to also deal with previously unknown objects, at the other extreme model-free approaches were proposed that segment a scene based on generic rules, e.g., using smoothness of surface normals [7], color [8], or convexity [9]. Naturally, due to the missing world knowledge, these approaches cannot correctly resolve ambiguous situations and often tend to under- or over-segment, e.g., merging two objects that are close to each other or splitting highly textured or ragged objects.

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.

Fig. 1. Segmentation and classification result of a mug (red bounding box) and its sub-parts (yellow bounding boxes) obtained from smooth surface patches (middle). The resulting segmentation tree groups loosely coupled image regions first, proceeding to more strongly coupled regions.

Between these two extremes there are numerous approaches that employ varying degrees of world knowledge. Rusu et al. [10] used Euclidean clustering of table-top scenes to come up with coarse segments of spatially separated object clusters, which were further subdivided by fitting geometric primitives such as spheres, boxes, and cylinders. Residual points were modelled using meshes. Later, support for arbitrary rotational surfaces was added [11].

Further abstracting from specific shape models, Richtsfeld et al. [12] group initially extracted surface patches into object hypotheses using an SVM trained to predict the connectivity of pairs of surface patches modeled as NURBS. This approach is very generic and yields state-of-the-art segmentation results, however, it comes at the cost of a high computational effort (1-8s per frame), currently disqualifying it for real-time applications.

In previous work [7] we followed a similar approach and initially segmented the image into smoothly curved surface patches. However, instead of using learned relations to group patches into object hypotheses, we employed several heuristic rules (cutfree adjacency, coplanarity, and curvature similarity) to determine a weighted connectivity graph that is clustered using graph-cut with a fixed decomposition threshold. Both approaches sometimes suffer from wrong grouping decisions resulting in over- or under-segmentation, especially in highly cluttered scenes. In fact, a bottom-up approach cannot decide whether two geometrically alignable image regions indeed belong to the same object or not, as this requires world knowledge about typical object shapes.

In this paper, we extend our previous approach to create a full hierarchy of grouping hypotheses – ranging from spatially neighbored point cloud blobs (at the roots of the determined forest) to object parts and individual object surfaces (at the leaves of particular trees). Explicitly representing the connectivity structure of the scene, we can postpone the final decision for a grouping hypothesis and exploit the
segmentation hierarchy by a higher-level decision process (involving world knowledge) to decide for the proper level of granularity and to find the optimal grouping hypothesis in a task-specific manner. To this end, we propose a simple nearest-neighbor classifier that traverses the hierarchy in a top-down fashion and stops if an object is recognized with high confidence.

As a “side effect”, the approach provides a spatial scene analysis, correctly grouping spatially connected objects, and it allows for an identification of objects parts, e.g. handles and knobs (see Fig. 1) that typically stick out from the base object shape and that were often separated by previous segmentation approaches due to missing world knowledge.

There has been some previous work done on hierarchical segmentation. In [13] contours detected on color images were used to build a hierarchy of homogeneously textured regions with increasing levels of detail down the hierarchy. The approach is applicable to any contour image, i.e. also to the depth contours estimated by our pre-segmentation algorithm. However, as only local homogeneity criteria were employed, the algorithm cannot recombine object parts that are spatially separated due to occlusion.

Model fitting approaches aiming to separate point cloud blobs into geometric primitives can also be considered to be hierarchical [14] – but they are naturally limited to the known set of geometric primitives. In contrast, our approach detects components in a first step and subsequently combines them to a hierarchy of object hypotheses. A similar approach is followed by Stein et al. [9], who employed a convexity criterion for recombination that roughly resembles our criterion for cut-free surfaces. However, by further adding heuristic rules to recombine parts separated by occlusions, we are able to avoid over-segmentation.

This paper is organized as follows: In the next section we shortly summarize common concepts of state-of-the-art segmentation approaches [7], [12], [9] and particularly outline the highlights of our previous method. Subsequently, in section III we introduce our extension to yield a hierarchical segmentation tree that is applicable to any underlying pre-segmentation approach. In sec. IV we describe the object classifier and its hierarchical application to the segmentation tree, before evaluating the qualitative results and describing our robotics demonstrator in sec. V. Finally, we discuss possible future work.

II. Real-Time Model-Free Segmentation

The state-of-the-art segmentation methods [7], [12], [9] follow a two-step algorithm: in the first phase, the scene is over-segmented according to some homogeneity criterion, finding regions of smoothly varying surface normals [7] or planar surface patches [12], [9]. In a subsequent step these low-level image segments are grouped to form object hypotheses according to some high-level grouping rules, using heuristic rules [7] or convexity of adjacent super-voxels [9], or using a SVM to predict connectivity of pairs of regions [12]. In all cases, the connectivity structure of image patches can be encoded within a graph with edge weights indicating the strength of connectivity between regions.

In our previous work, we used graph-cut [15] with a fixed cost threshold to determine clusters of strongly coupled nodes within this graph, which represents the final grouping hypothesis. In this work we avoid the restriction to a single grouping result, but aim for a grouping tree that encompasses all potential grouping hypotheses, leaving the decision for a particular solution to a higher-level process.

A. Pre-Segmentation into Surface Patches

As outlined above, the objective of the first processing phase is to segment the depth image into uniform regions – in our case into spatially separated regions of smoothly varying surface normals. To this end, we use the following processing pipeline (as detailed in [7]):

1) three-stage smoothing (median, temporal, Gaussian)
2) calculation of surface normals (from cross product)
3) detection of object edges from angle between normals ($\theta_1 \cdot \theta_2 < \Theta$) and Euclidean distance of adjacent points
4) connected component analysis to assign unique IDs to all surface regions
5) assignment of edge points to closest surface region

Fig. 2 shows the intermediate object edge image (step 3) and the resulting pre-segmentation. Notice, that edge pixels belonging to very bold edges or that are too distant from neighboring surfaces are not yet associated to any surface region. Those pixels (shown in grey) typically correspond to separate objects or parts and will form separate segments that will be linked to nearby surfaces within the subsequent processing step that creates the connectivity graph.

B. Connectivity-Graph Creation

Based on the pre-segmentation, the connectivity graph is determined using three geometrically motivated matching criteria: (i) cut-free adjacency, (ii) co-planarity and (iii) similar curvature of surface patches. Each criterion contributes a subset of edges within the graph. In the following we consider each criterion in more detail.

1) Cut-free adjacency: In a first sweep we consider all pairs of surface patches that are adjacent in 3D Euclidean space. If one surface cuts the other, such that a considerable amount of points are lying on both sides of the former surface, the two surfaces presumably do not belong to a
common object. Otherwise an edge connecting both surfaces will be added. For illustration consider Fig. 3: Here, all adjacent surfaces are pairwise cut-free – except the pairs 1-5 and 1-6, because faces 5 and 6 split their supporting surface 1. Therefore, the resulting connectivity graph already correctly resembles the object structure of this scene. Please note, that this criterion closely resembles the convexity criterion proposed in [9]. In order to deal with noise and curved surfaces, we employ a RANSAC-based plane approximation.

2) Co-Planarity: In order to recombine object parts which were separated by occlusion (e.g. surfaces 2,8 and 3,5,1 in Fig. 4), we proposed to match the shapes of non-adjacent surface patches. To this end, we distinguish between planer and curved surfaces. Two planar surfaces are considered to be co-planar if they span a common plane, i.e. if their normals are similar and if random points chosen from both surfaces are aligned with the common plane. Additionally to this geometrical test, we need to check whether both surfaces are indeed separated by occlusion (and not accidentally aligned as surfaces 7 and 4 in Fig. 4). If any pixel along random lines connecting both surfaces (cf. Fig. 4) is “behind” the virtual common plane (i.e. having larger depth value), this pixel is considered as background and the surfaces are not considered for recombination (cf. red lines in Fig. 4). Otherwise new edges are added to the connectivity graph.

3) Curvature Matching: In order to handle curved surfaces in a similar fashion, we compare their curvatures based on their normalized curvature histograms representing the distributions of surface normals. The $x$-$y$-histogram of $11 \times 11$ bins measures the relative frequency of given $(x, y)$ components in the surface normals of each surface patch. As can be seen from Fig. 5 these histograms are compact fingerprints of the shape and orientation of the corresponding surface patches: Cylinders generate a line in the histogram, indicating a smooth transition of surface normals along a single arc. The line orientation resembles the arc orientation. On the other hand, a spherical object generates a full circle distribution indicating a smooth transition of normals along two dimensions. If two histograms have large enough overlap according to the similarity index $S(A, B) = \sum_{ij} \min(a_{ij}, b_{ij})$, their corresponding surfaces are considered for recombination. As before, we also check whether the two patches are aligned and separated due to occlusion. For further details we refer to [7].

C. Remaining Edge Points

As pointed out in section II-A, larger edge-point blobs were not yet assigned to a neighbored surface patch during the pre-segmentation phase. These blobs frequently correspond to separate and small objects (or parts, e.g. handles or knobs) and thus are handled as separate image regions within this work. In order to separate small, adjacent objects that form a single edge blob (e.g. several apples in a row), we segment those edge-point blobs using a region growing algorithm working in image space and employing the 3D Euclidean distance as the main homogeneity criterion. Additionally we consider maxima in the depth image as indicators of “separation clefts”: Two close-by objects usually exhibit a depth image as shown in Fig. 6 that exhibits distinct grooves where the objects touch. Therefore, we perform a maximum search along all eight region growing directions (horizontal, vertical, and diagonal) to identify those grooves. If found, the region growing will stop despite a small Euclidean distance of points. All newly found image regions will be added as nodes to the connectivity graph and linked to adjacent surface nodes if their shortest Euclidean distance is smaller than a threshold of 8mm.

This new edge-point processing replaces a set of complex heuristic rules that we used in our previous work [7] and that were not able to separate small, close-by objects.
III. Hierarchical Segmentation

Taking the undirected and unweighted connectivity graph as its input, the hierarchical segmentation algorithm aims to find a hierarchical clustering of graph nodes into groups of increasing connectivity strength. To this end, we first assign connection weights to all graph edges indicating the connectivity strength or cutting costs. Subsequently we iteratively apply the minimum cut algorithm [15] to split nontrivial, connected subgraphs into pairs of subgraphs such that the sum of removed edge weights is minimal at each step. If the initial graph consists of several disconnected subgraphs (cf. Fig. 4), each of them will be treated independently and we will arrive at a forest of composition trees, each associated with grouping hypotheses for a sub-scene.

A. Initial edge weights

The naive approach, counting the number of removed edges (corresponding to edge weights equal one), would correctly separate loosely connected 3-cliques typically arising from boxes (e.g. 3-4-5 and 6-7-8 in Fig. 7). However, this approach will fail to prefer separation of subgraph 1-2 from 3-4-5, because there is only a single connecting edge in both cases, i.e. resulting in identical costs.

Our solution is to assign edge weights \( w_{ij} = 1/n \) to all edges \((i,j)\) originating from node \(i\), where \(n\) denotes the number of outgoing edges of node \(i\). While this results in a directed graph with asymmetric weights, and our aim is to have a symmetric cost distribution, weights of incoming and outgoing edges are averaged in order to ensure a symmetry

\[
W_{sym} = \frac{1}{2}(W + W^t)
\]

as illustrated in Fig. 7. This normalizes the costs of any graph cut to the range \([0,1]\) and causes the costs to separate a single node to be equal (or close to) one. In this context, please notice, that we consider only cuts into two subgraphs, that is in Fig. 7 only nodes 2, 4, 7, 8 could be isolated, always resulting in high costs close to one. On the other hand, smaller cliques gain stronger internal connectivity compared to their bridging links to adjacent cliques: Although individual edge weights within the cliques 3-4-5 and 6-7-8 are small, each cut dividing them would have costs close to one. In contrast, the bridges between cliques have even lower weights and thus become first candidates for cutting. Nodes 1 and 2 become more strongly connected, because there is only a single edge from node 2.

B. Fine-tuning of edge weights

Figure 8 illustrates another example comprising edges introduced due to different heuristic criteria: cut-free adjacency (black), co-planarity (red), curvature (blue), and remaining edge points (green). Analyzing the resulting weights, we observe two problems for a proper hierarchy deduction:

1. The subgraph 2-5-1-3-8 (after separation from 4-6-10) has two minimum cuts with identical costs: one splitting off node 1 and the other (preferred one) splitting off 2-8.
2. The subgraph 7-9-11 also offers two possible cuts: removing the edges originating from the curvature or the remaining points rules resp.

To resolve these ambiguities and to prefer correct groupings we propose to employ a-priori confidence weights for each edge type: Edges originating from the cut-free adjacency criterion have highest confidence and thus should be least preferred for cuts, when compared with other edges. Edges that were added to link regions classified as “remaining edge points” (Sec. II-C) should have highest cutting preference, since those only correspond to extensions of the main object body. Therefore we propose to assign weights \(\omega_a > \omega_p > \omega_c > \omega_r\) to the different criteria cut-free adjacency, co-planarity, curvature, and remaining points respectively.

By multiplying the basic edge weights with these feature-related weights, the relative influence of a single feature is increased or decreased and ambiguities are resolved.

C. Hierarchical Segmentation Tree

The segmentation forest resulting from processing all separated sub-graphs is shown in Fig. 9. For example, the sub-graph 7-9-11 is first divided along the remaining-points edge, separating the mug handle from its body. The two sub-graphs of each cut are added as children of a new virtual parent node that replaces the original graph. The sub-graphs are in turn processed with minimum-cut, now separating the inner and outer surface of the mug that were linked due to the curvature matching criterion. This process is repeated until a single node (corresponding to a single surface patch) remains at the leaf of the tree. These leaf nodes contain the actual point cloud data, while all branching nodes correspond to potential grouping hypotheses. Due to the minimum cut strategy, branches at the root have weakest connectivity, while leaf nodes have strongest connectivity.
IV. HIERARCHICAL CLASSIFICATION

The obtained region hierarchy plays the central role to realize a very flexible interplay between bottom-up region segmentation and top-down application of world knowledge about the appearance of semantically relevant entities: we use the segmentation hierarchy to control the spatial attention of a region-based classifier that comprises all world knowledge about the appearance of semantically relevant scene entities. To this end, a fast NN-based classifier traverses all trees of the segmentation forest from top to bottom, computing at each node, that is at all segmentation scales, the class probability distribution for the image region corresponding to that node. Peaks of classification confidence will indicate salient entities in the scene. Typically, we expect these to correspond to objects. However, the hierarchical application of the classifier admits simultaneous confidence peaks at several levels that reflect the detection of salient object parts (at lower levels – e.g. a handle), as well as the detection of salient object groupings (at higher levels – e.g. a heap of apples). Thereby, the region hierarchy connects very efficiently two complementary representations of regularities in the world: (i) very generic, low-level regularities about homogeneity structures that govern segmentation, and (ii) specialized, high-level feature correlations that are indicative of objects, object parts or special object configurations. An illustrative example is depicted in Fig. 1, where the mug’s handle and body as well its inner and outer surface could be recognized individually.

Below, we present a concrete implementation for a first evaluation of the described, very general scheme. Major design criteria for this implementation were a simple, yet extensible feature representation along with a fast and robust trainability of the classifier, but the overall approach is open to many straightforward extensions and enhancements to address additional requirements beyond the scenarios we are considering.

A. Bag-of-features NN-classifier

For classification we employ a standard nearest-neighbor (NN) classifier that internally works on a bag-of-features model, namely using size, elongation, and color features as detailed in next section. For each feature type \( i = 1 \ldots n \) with an associated feature set \( F_i \subseteq \mathbb{R}^{d_i} \) we define a symmetric distance function

\[
d_i : F_i \times F_i \rightarrow \mathbb{R}^+
\]

that computes a normalized distance of a pair of features. If they exactly match, the distance equals zero, otherwise the distance should not exceed a value of one. This normalization ensures that distances from different feature types can be easily combined, weighted, and compared with each other. The overall feature vector \( \mathbf{f} = (f_1 \ldots f_n) \in F = F_1 \times \ldots \times F_n \) aggregates all individual features. To compute the overall distance of two feature vectors \( \mathbf{f}^a, \mathbf{f}^b \) we use the maximum of all individual feature distances:

\[
d(\mathbf{f}^a, \mathbf{f}^b) = \max_{i=1 \ldots n} d_i(f_i^a, f_i^b)
\]

Using the maximum ensures that a perfectly matching feature type \( i \) cannot balance out a not well matching feature type \( j \). Rather, it requires that all \( n \) feature types match to a common degree. As an example, consider the classification of bananas and lemons: both classes exactly match by color, but have distinct size and shape, and thus cannot be confused.

NN classifiers facilitate incremental online learning, as new training samples \( \mathbf{f} \) can simply be added, with each object class \( c \) represented by the set of so far seen training samples \( \{\mathbf{f}_c^i\} \) of this class. In order to classify an observed feature vector \( \mathbf{f}_c \), the closest training sample is searched and its associated class label is returned:

\[
c^* = \arg \min_c \min_j d(\mathbf{f}_c^j, \mathbf{f}_c).
\]

By using the minimum prototype distance per class, the training becomes very aggressive, yielding very fast learning capabilities – usually very few or just a single prototype suffices – at the cost of limited generalization behavior. However, in our scenario with a small set of well distinguished classes, we yield a very responsive classification performance where both training and classification can be performed in real-time. Better generalization could be achieved by using a \( k \)NN-classifier, which, however, needs an absolute minimum of \( k \) prototypes per class.

B. Shape and Color Features

The main performance bottleneck for \( (k) \)NN classifiers is usually the feature matching step, which is why a lot of research has been focused on its optimization [16], trying to find highly descriptive, but low-dimensional feature vectors that can be matched efficiently. The bag-of-features approach allows to easily combine several domain-specific features \( i \) that can be matched very efficiently using individual distance functions \( d_i \). Particularly we use two shape features (size and elongation) and a color feature as follows:

The size feature denotes the volume of the segment, which is easily estimated from the eigenvalues \( \lambda_i \) obtained from a PCA applied to the segment’s point cloud:

\[
v = \prod_i \sqrt{\lambda_i}.
\]

The corresponding distance function is defined as:

\[
d_{\text{volume}}(v_1, v_2) = |v_1 - v_2| / \max(v_1, v_2),
\]

where the division by the maximal volume ensures proper normalization.
The elongation feature denotes the ratio between length and thickness of an object, and is again computed from the PCA eigenvalues:

$$e = \min(\lambda_1/\lambda_2, \lambda_2/\lambda_1). \quad (6)$$

The associated distance function is identical to Eq. 5. We explicitly avoid using the third (smallest) eigenvalue, which often corresponds to the object’s visible extension along the camera’s view direction: the object’s point cloud is always heavily truncated along this direction, because a single camera can only provide a frontal view.

To compare the color distribution of two point clouds, we use a normalized and non-uniformly binned hue/saturation histogram: The lightness value $L$ from the HLS color space is ignored to achieve robustness against varying light conditions and shadows. To allow for a fine-grained distinction of color values, we use 128 bins for hue, but only 8 bins for saturation. Using powers of two allows to efficiently address histogram bins using the 7 resp. 3 most significant bits of hue resp. saturation values.

Histograms are blurred, applying an $11 \times 3$ Gaussian filter, and subsequently normalized ($\sum H_{h,s} = 1$) to achieve better generalization and to allow for comparison. In order to avoid a special treatment of the circular hue space, the Gaussian kernel mask is not cropped but applied in a cyclic fashion along the histogram’s hue axis. The distance function simply computes the overlap of two histograms:

$$d_{\text{color}}(H^1, H^2) := \frac{1}{2} \sum_{h,s} |H^1_{h,s} - H^2_{h,s}|. \quad (7)$$

V. Evaluation

TU Vienna openly provides an object segmentation database (OSD), which consists of a large set of labeled RGB-D images, to allow for the evaluation of 3D segmentation algorithms [17]. Table I provides results of recent work evaluated using this database and shows that our previous approach [7] performs best for real-time performance. We note that the OSD author’s own results are slightly better w.r.t. over-segmentation (false-positives). By design, the OSD training set does not contain all objects from the test set. Since this means that we cannot train our system’s classifier on all objects that occur in the scenes, a quantitative analysis of our approach was not possible using the OSD dataset. However, inspecting the accompanying video or the cluttered scene image in Fig. 11, we observe the same qualitative level of segmentation accuracy. Based on some generic examples, we will in the following illustrate the improvements of the new approach over our previous work.

<table>
<thead>
<tr>
<th>Method</th>
<th>True-pos.</th>
<th>False-pos.</th>
<th>False-neg.</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uckermann [7]</td>
<td>96.3±4.1</td>
<td>2.5±4.5</td>
<td>3.7±4.1</td>
<td>30-40ms</td>
</tr>
<tr>
<td>Richtsfeld [12]</td>
<td>97.2±4.8</td>
<td>5.8±10.3</td>
<td>2.8±4.8</td>
<td>1-8s</td>
</tr>
<tr>
<td>Stein [9]</td>
<td>90.7±8.7</td>
<td>4.3±2.5</td>
<td>9.3±8.7</td>
<td>550ms</td>
</tr>
</tbody>
</table>

As can be seen from Fig. 1 showing classification results for a mug, our approach can correctly identify all sub parts of an object as well as the object itself. The identification of handles or knobs and of concavities is particularly important for task-specific grasping and for pouring tasks resp. In previous work such object parts were either absorbed within the overall object point cloud or were segmented as separate regions without the link to the main object.

Figure 10 shows situations that can only be segmented and classified correctly by combining bottom-up and top-down processing streams: In both images, a box is separated into two parts due to the occluding cylinder. Although having different color distributions, both parts in the left image belong to the same object (Kinect box). On the other hand, in the right image, both parts actually belong to different, but perfectly aligned boxes. A proper grouping decision can be achieved in this case only if learned appearance models of the objects are exploited. In the corresponding segmentation tree, whose leaf nodes are colored according to the found surface patches of the pre-segmentation phase, the grouping will either combine both branches or split them – depending on the confidence level of the classifier.

A. The robotics demonstrator

We applied our algorithm in an interactive robot scenario, where a human user can command several tasks, like “Put the apple into the left basket!” or “Give me this big banana!”.

In these tasks, object references usually need to be resolved to a single object to be grasped. However, due to ambiguities in speech and noisy scene understanding, references are not unique. The robot resolves these ambiguities by asking for individual object attributes provided by the classifier (color, size, elongation) or by external features (relative position, pointing gesture). In order to find a proper place position (e.g. “in”, “left of” a reference object), we determine the free space according to the requested spatial relation. For grasping, a superquadrics model is fitted to the segmented object point cloud allowing to determine a grasp prototype (from
shape/size parameters) and the correct hand alignment (from position and orientation). Potential grasp candidates are ranked with respect to collision and kinematic constraints. Finally the object is grasped in a compliant fashion [18]. Both, the interactive dialogue and the grasping phase rely on accurate segmentation hypotheses provided at real-time by the proposed algorithm, thus allowing to react to changes in the environment. A video illustrating the described human-robot-interaction is available at youtu.be/mkGp_V0oDvo.

VI. SUMMARY AND OUTLOOK

This paper proposed a generic method to yield a hierarchical segmentation tree by iteratively applying minimum graph-cut to a weighted connectivity graph defined on a meaningful over-segmentation of RGB-D images. As such, the approach can also be applied to other state-of-the-art pre-segmentation approaches to yield a whole set of segmentation hypotheses with an increasing degree of detail. Combining the bottom-up segmentation algorithm with a top-down classification we were able to improve our segmentation results in complex scenes by autonomously finding the correct grouping level. Furthermore, the approach allows for a task-dependent focus on the grouping hierarchy in order to identify task-relevant object parts, like handles or knobs for grasping and concavities for pouring.

Obviously, the choice of specific rules to create graph connections and their weightings is crucial for the resulting segmentation hierarchy. In this work we relied on heuristic rules already introduced in our previous work [7]. However, several alternatives are possible. The basic local connectivity could be established with the more general convexity criterion proposed in [9]. However, as this is only a local criterion considering adjacent image regions, it cannot recombine object parts separated due to occlusion. To this end, more complex shape and color matching heuristics need to be used. An alternative to our static rules could be to use learned rules involving shape and color features as was done in [12]. However, an important filter for re-combination should be the occlusion check proposed in this work to ensure that object parts are indeed separated by occlusion. Even if this check is passed, object parts might only accidentally match in color and shape. Therefore, their 3D Euclidean distance should be exploited to modulate the weighting of an introduced graph edge. The more two regions are separated, the less likely they belong together.

Our method can also be extended to handle both, known and unknown objects at the same time. To do this we can simply resort to our previously presented, fixed threshold-based grouping approach to generate grouping hypotheses for individual trees that cannot correctly be classified.

REFERENCES