# Convexity based object partitioning for robot applications

Simon Christoph Stein, Florentin Wörgötter, Markus Schoeler, Jeremie Papon and Tomas Kulvicius

*Abstract*— **The idea that connected convex surfaces, separated by concave boundaries, play an important role for the perception of objects and their decomposition into parts has been discussed for a long time. Based on this idea, we present a new bottom-up approach for the segmentation of 3D point clouds into object parts. The algorithm approximates a scene using an adjacency-graph of spatially connected surface patches. Edges in the graph are then classified as either convex or concave using a novel, strictly local criterion. Region growing is employed to identify locally convex connected subgraphs, which represent the object parts. We show quantitatively that our algorithm, although conceptually easy to graph and fast to compute, produces results that are comparable to far more complex state-of-the-art methods which use classification, learning and model fitting. This suggests that convexity/concavity is a powerful feature for object partitioning using 3D data. Furthermore we demonstrate that for many objects a natural decomposition into "handle and body" emerges when employing our method. We exploit this property in a robotic application enabling a robot to automatically grasp objects by their handles.**

## I. Introduction

Robots must be able to interact with and manipulate objects. However, what is an *object*? As early as 1000 AD the first notions arose that shape/object perception relies on convexity and concavity information. In the first known book on visual science, written by the Arab scholar Alhazen (Ibn al-Haytham), 965 - ca. 1040 AD [1] he stated that *connected convex surfaces* lead to the perception of a solid object ("if the body has a convex surface that bulges towards the eye [...] then if sight perceives the convexity of the surface it will perceive the body's solidity"; [2], p. 169). A substantial body of psychophysical and theoretical literature exists that has tried to substantiate this claim for human perception, but almost exclusively dealing with 2D shapes [3]–[8]. In addition a few early studies in computer vision have used this concept to distinguish objects from each other also in 3D [9]–[11]. These older studies, however, suffered from a lack of good 3D-data, which only now has become readily available through the use of RGB-D sensors (like the "Kinect"). Thus, only recently the aspect of shape perception relying on convexity and concavity has become used in technical systems [12]–[14], with variable success. Why is object segmentation so difficult? One reason for this is that most objects are composed of parts, which can have their own functional semantics (very often: body and handle). Thus, data driven, bottom up *whole-object* segmentation is an ill-posed problem if not considering parts (and their combinations) early on.

*The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

Georg-August-Universität Göttingen, Bernstein Center for Computational Neuroscience, Friedrich-Hund Platz 1, DE-37077 Göttingen, Germany tomas@physik3.gwdg.de

In this study we address this problem by the use of a well-designed concave-convex criterion on point cloud data and show that this is exceedingly powerful for the data-driven finding of *parts of objects*. The main novelty of our approach lies in the definition of this strictly local 3D-partitioning criterion and its combination with a region-growing algorithm working on surface patches. This largely mirrors human perception and thereby creates object parts from the point cloud data in a natural, human-like way. Specifically, from such a partitioning we can demonstrate that the notion of "handle versus body" genuinely emerges for many objects, which is very useful for robotic grasping. This paper is organized as follow: First, in Section II we present our segmentation algorithm, for which technical details are given in the Appendix. In Section III we evaluate our method and benchmark it against other approaches. After that, we show one demonstrative example of a robotic application: Grasping objects by their handles. Finally in Section IV we discuss the results and compare them to the state of the art. The method source code is freely distributed as part of the Point Cloud Library (PCL)[1].

## II. Methods

### A. Method overview and basic definitions

The basic assumption of our segmentation is that object parts are usually separated from each other by concave boundaries. Early on we note that single-part objects are just a special case of this. Based on this hypothesis, the goal of the algorithm is to segment scenes by merging convex areas enclosed by concave boundaries. Convex is defined here in the usual way (Fig. 1 C, left): Two touching surfaces of an object form a convex configuration if a straight line, which connects one point on one surface with another point on the other surface, cuts through (the solid part of) the object. Accordingly, a concave configuration is given when the connecting line travels through "free space" (Fig. 1 C, right). However, there exist configurations where the observed surface is locally discontinuous and the classification into convex and concave does not make sense (Fig. 1 D ). Hence, just applying the concave-convex criterion as such can lead to wrong decisions and thus a wrong segmentation. A main contribution of this study is to address this problem using some simple geometric criteria.

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

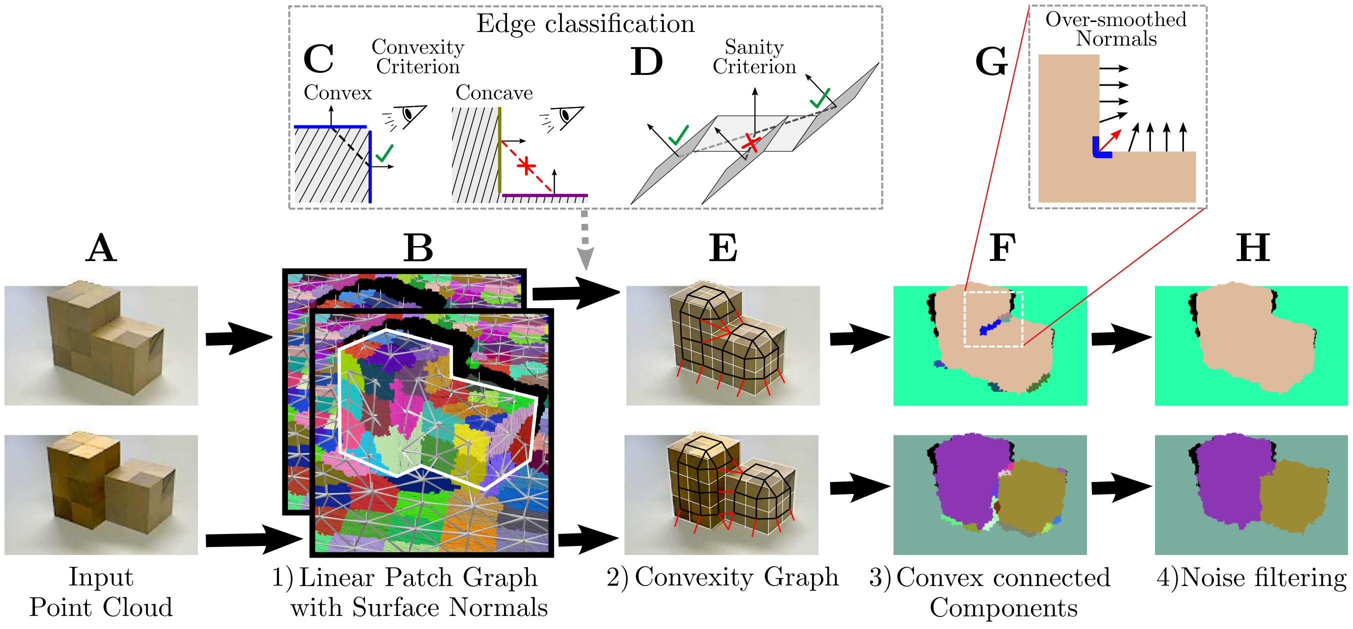

Fig. 1. Flow diagram of the segmentation algorithm. Two example cases are shown: single-object case (upper panels) and two-objects case (lower panels). Illustration of **A)** RGB images corresponding to the point clouds (not shown) of the scenes, which serve as an input to the segmentation algorithm. **B)** Graph of connected supervoxels (linear patches). For clarity, the displayed patches are bigger than the ones used for segmentation. **C)** Convexity criterion and **D)** sanity criterion for the classification of graph edges. **E)** Model depicting the classified graph. Black lines denote convex connections, red lines concave/invalid ones. **F)** Resulting Segmentation; object labels are shown by different colors. **G)** Magnification of noisy region in the segmented image which is due to over-smoothed normals. **H)** The final segmentation result after noise filtering.

## B. *Method flow-diagram*

The flow diagram of the *Locally Convex Connected Patches* (LCCP) segmentation algorithm is presented in Fig. 1. To explain our algorithm we have designed two simple objects using wooden cubes: an object that a human observer would consider as consisting of a single-part (upper row) and another one from two parts (bottom row). In the following we will give a general overview of the implementation of our algorithm. Details are given in the Appendix.

Our method is based on the segmentation of 3D point clouds recorded with a Kinect sensor, which serves as input to our algorithm (RGB images shown in Fig. 1 A). As we concentrate on geometric criteria, we omit the RGB data and use the depth data alone.

First, we build a *graph of connected linear patches* (Step 1, panel B) as an approximation of the observed surfaces in the point cloud. This is done using the Supervoxel algorithm of [15], which is an edge preserving oversegmentation, where each supervoxel in an adjacency graph is taken as a linear patch (e.g. a patch with zero curvature) and its normal vector is calculated. The main advantage of this step is that it reduces noise and thereby increases the stability of the convexity decision. Additionally a substantial data-reduction is achieved, making the algorithm faster.

Afterwards, we create a *convexity graph* (Step 2, panel E) by classifying edges of the linear patch graph. To decide whether a connection between patches is convex or concave we use two criteria, *convexity* and *sanity* (Fig. 1 C, D). Convexity is defined as described above, but connections between patches whose normals differ less than a small angle threshold $\beta_{Thresh}$ are always treated as convex. This threshold compensates for inaccuracies in the normal estimation and allows merging of small, spurious concavities. The sanity criterion is used to identify and invalidate connections where patches are only connected in a singular point making the convexity decision ambiguous.

Finally, in Step 3, we segment the obtained convexity graph in order to find all components connected by convex edges (Fig. 1 F). We achieve this by a region growing process. Starting from any seed-patch, region growing propagates the seed-label over those patches that have convex edges until a concavity is reached, for which the region cannot “grow around” and the process starts with a new seed (see Appendix for details). This can best be understood comparing panels F. In the upper panel only one object is labeled. The corresponding convexity graph (panel E) shows why this happens. Although a concave boundary exists for this object, region growing finds enough convex-connected patches such that the label can grow around the concave edge. A different arrangement is presented in the bottom panels. In this case the object splits into two separate parts (panel F). This is due to the fact that the front part of the structure is not a single plane anymore but shows a step-like structure. This discontinuity leads to an enclosing concave boundary which cuts the convexity graph (panel E) into two parts that cannot be bridged by region growing. As a result, the algorithm interprets this scene as two touching objects.

In the resulting segmented images (panels F) one can see small segments at the edge. This happens due to the gradual transition of the normals at the edge (Fig. 1 G), because

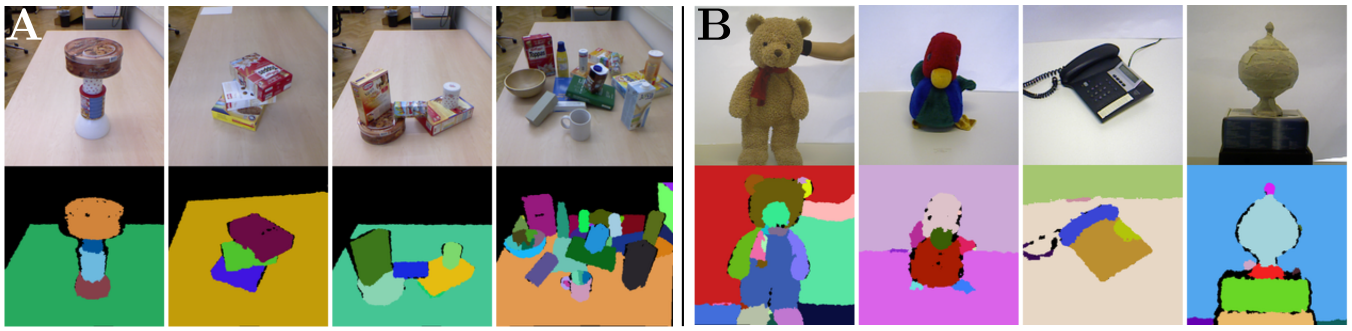

Fig. 2. Examples of segmentation: **A)** images from OSD dataset and **B)** images from our data set. Top and bottom panels show original and segmented images, respectively. Points beyond a distance of 1.3 m were cropped for visualization.

normals are estimated using a local neighborhood. Because of this, a group of normals may seem to have a concave connection to both surfaces leading to unwanted segments. As these patches are usually very small, they can be removed with filters in a post-processing step (Step 4, Fig. 1 H).

### C. *Benchmarking and Measures*

This section provides a short overview of the benchmarks and measures that were used for quantitative evaluation. In addition to publicly available benchmarks we also use a set of self recorded scenes for examples and qualitative evaluation.

*1) Object Segmentation Database (OSD):* For quantitative analysis we used the *Object Segmentation Database* (OSD-v0.2) which was proposed by Richtsfeld *et al.* [12] in 2012. It consists of 111 scenes showing objects placed on a table. All scenes contain multiple objects, which have mostly box-like or cylindrical shape and are recorded in various positions. The data set includes scenes with partial and full occlusions and also cluttered scenes (in 2D as well as 3D). An important property of the data set is that most objects are not composed of parts. This makes the ground-truth data relatively non-ambiguous. Ground-truth images were created from the points in the labeled point clouds available on the OSD website[2]. Example scenes from the OSD dataset can be found in Fig. 2 A.

*2) Measures:* The first measure that we used for evaluation of our algorithm is *Weighted Overlap* ($WOv$) proposed by [16] and [17], which is a simple region based measure that is computed from the view of the ground-truth partition. The other measures we used are *false negative* ($fn$) and *false positive* ($fp$) scores from [13] and *over-* ($F_{os}$) and *under-segmentation* ($F_{us}$) scores from [12]. Definitions for these measures are given in the Appendix.

## III. Results

One strength of the LCCP algorithm is its robustness to parameter variations. For all segmentation images shown in this work, the parameters of the algorithm remained the same (parameter set P1, see Appendix Tab. II, with $\beta_{\text{Thresh}} = 10°$), except for one aspect of the robot application (Fig. 5 B), where object parts are intentionally merged.

[2]http://users.acin.tuwien.ac.at/arichtsfeld/?site=4

### A. *Segmentation Examples*

Some examples of results of our segmentation algorithm are presented in Fig. 2, where we show the segmentation of images from the OSD as well as our data set (panels A,B; resp.). In the OSD data set “single-part” objects dominate. Therefore, we selected images from our data set (with “multiple-parts”) in order to show that our algorithm is not only able to perform object segmentation but also object partitioning. One can see that in both cases our algorithm performs very well and is able to segment objects as well as objects’ parts. Hollow objects (bowls, etc.) will show multiple segments inside as surface normals on this concave surface change very strongly. This could be changed (to getting a single segment) by a different parameter set but this segment will always be different from the one that represents the outside of the object, as they are not connected in 3D space due to occlusion. Note that if point clouds from multiple viewpoints are used, bowls turn into single segments.

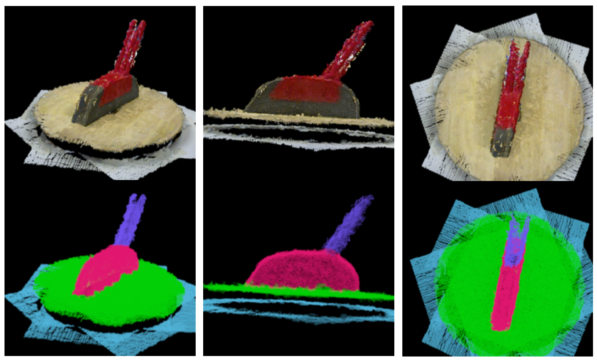

Fig. 3. Example segmentation of a point cloud combined from multiple views showing a foam hand broom. The input point cloud and segmentation result from different view points are shown in the top and bottom images respectively (also see supplementary video).

We would also like to stress that our method allows performing segmentation of point clouds taken from multiple views. An example is shown in Fig 3, where eight views of the same object were recorded using a turn table and their point clouds merged. To get a correct surface orientation, the normals are calculated for each cloud individually before they are combined. Normals of points inside a voxel are then averaged. Segmentation of clouds combining multiple views requires a method rigorously working in 3D space instead of the image plane, and is often not achievable in other approaches.

## *B. Method Evaluation, Statistics and Run-time*

We evaluated the performance of our algorithm on the OSD data set and compared it to two state-of-the-art methods. Note that benchmarking 3D segmentation is in itself non-trivial as the ground truth often contains inaccuracies (see Appendix for ”Evaluation Problems”).

Performance of our algorithm is quantified in Fig. 4 giving average scores on the OSD dataset for different $\beta_{Thresh}$ and two supervoxel sizes (P1=small and P2=large, see Appendix Table II). For small $\beta_{Thresh}$ (region *R1*), noise in the normal estimation influences the segmentation, resulting in high oversegmentation (high false negatives). When the merging angle is increased, oversegmentation is reduced and a stable plateau is visible (*R2*). For large merging thresholds (*R3*), undersegmentation occurs (high false positives). Differences between supervoxel sizes do not matter much for small $\beta_{Thresh}$ (regions *R1*, *R2*) but larger supervoxels improve results for large $\beta_{Thresh}$ (region *R3*).

A comparison to two other state-of-the-art methods using model fitting together with machine learning [12] or probabilistic reasoning [13] is presented in Table 1. It can be seen that, although simpler, our method can compete with the state-of-the-art methods. Oversegmentation ($fn, F_{os}$) is slightly higher with our method. This is due to two factors: we do not use model fitting (which helps against noise), and we sometimes detect parts (e.g. handles of bowls and cups) which is an oversegmentation according to the full-object ground truth labeling. We should note that the latter of which is not an error for our purposes. In terms of undersegmentation ($fp, F_{us}$), we perform better than [12] and slightly worse than [13].

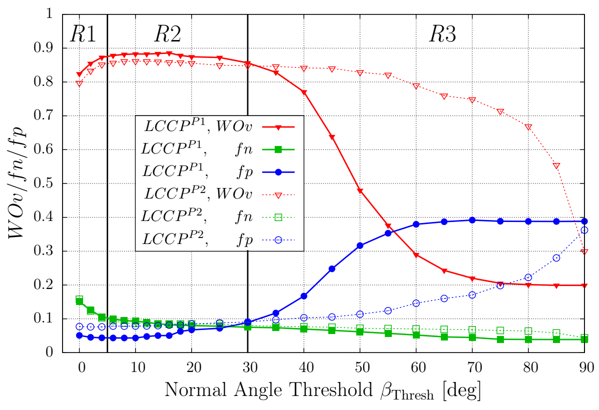

Fig. 4. Statistics obtained from segmentation of scenes from the OSD dataset using our method. Average results are shown. $LCCP^{P1}$ and $LCCP^{P2}$ stand for the different parameter sets with small (solid lines) and large (dashed lines) supervoxels respectively.

The average run-time on the OSD dataset using parameter set P1 (P2) was 549 ms (370 ms) with 518 ms (365 ms) spent computing the supervoxels and 31 ms (5 ms) for the segmentation using a Intel Core i7 3.2 GHz processor. Note, supervoxel calculation is currently not parallelized using GPUs, which should lead to a more than 10-fold speed-up.

## *C. Robotic application*

Finally, we applied our algorithm to a robot scenario where a KUKA LWR robot-arm [18] was used to grasp some objects. Several aspects, such as object recognition [19], robot grasping control [20], [21], and movement execution [22], rely on published work and will not be described here in detail.

The task for the robot was to identify eight different objects in a scene (Fig. 5 D) and grasp some of them by their handle to lift them above the table. In addition to segmentation, this task also requires object classification. General object classification, a difficult problem in its own right, is outside the scope of this paper. Here we have restricted the problem to only those eight classes and we could, thus, use an established classification algorithm [19] to recognize them.

The flow diagram of the implementation is presented in Fig. 5. As input to the robot-system we used the point clouds obtained from Kinect data (for segmentation) as well as a high resolution DSLR image (for object recognition). We enhanced our segmentation algorithm by an initial ground plane separation step (using PCL) and used the overall setup for two aspects required to solve this problem: 1) object-segmentation from the background (Fig. 5 B) and 2) partitioning the individual parts of a given object (Fig. 5 C). Parameters for (1) and (2) are necessarily different. Ground plane subtraction is necessary because the object-to-table and handle-to-object-body transitions are geometrically similar (90° edge) and the object-segmentation can thus not be solved by LCCP segmentation. As an additional benefit, it helps to segment very thin objects like the knife, which can hardly be (geometrically) distinguished from the supporting surface when laying on the side, because of the limited resolution and accuracy of the Kinect. For object and handle recognition we used the classification algorithm from [19], which is based on a combination of SIFT-features [23], CyColor-features and a radial orientation scheme [19]. We used a two layer architecture. The first layer consists of a classifier which is trained on all eight complete objects in the scene (see Fig. 5 A). This classifier finds the desired object(s) in the scene using the DSLR image on the eight possible object candidates, segmented by the object-segmentation step (Fig. 5 B). Here it detects the heat gun (Fig. 5 D). The second layer consists of several binary classifiers (one for each object), which classify a part, segmented by the part partitioning step (Fig. 5 C), as being “handle” or “body”. Thus, for handle classification we trained eight classifiers on handles vs. body of the respective object. Here it splits the heat gun into body and handle as required (Fig. 5 E) and the same happens for all objects in the scene.

For action execution we used the library of manipulation actions from [21], which is based on Semantic Event Chains (SECs) [20] and Modified Dynamic Movement Primitives

TABLE I

COMPARISON OF DIFFERENT SEGMENTATION METHODS.

| | $WOv$ | $tp$ | | $fp$ | | $fn$ | | $F_{os}$ | $F_{us}$ |
|---|---|---|---|---|---|---|---|---|---|
| | Mean | Mean | SD | Mean | SD | Mean | SD | Mean | Mean |
| $LCCP^{P1}$ ($\beta_{Thresh} = 10°$) | 87.0% | 90.7% | 8.7% | 4.3% | 2.5% | 9.3% | 8.7% | 8.4% | 3.9% |
| Richtsfeld et al. [12] | - | - | - | - | - | - | - | 4.5% | 7.9% |
| Ückermann et al. [13] | - | 92.2% | 7.3% | 1.9% | 3.3% | 7.8% | 7.3% | - | - |

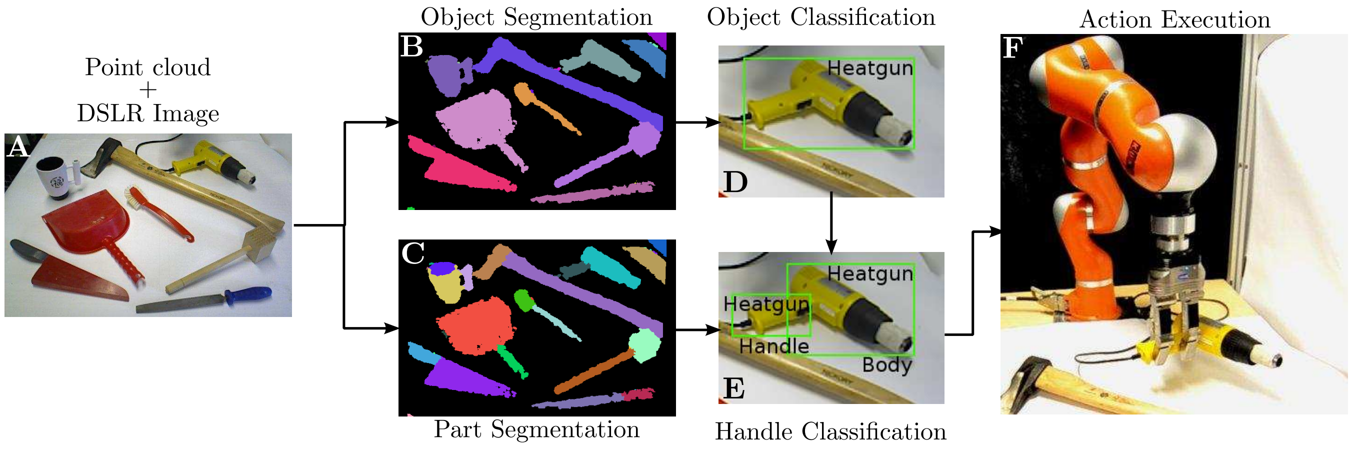

Fig. 5. Flow diagram of robotic application: **A)** Original image of the scene, **B, C)** object/part segmentation, **D, E)** object/part classification and **F)** action execution – robot grasping a heatgun by its handle. LCCP segmentation was applied after an initial ground plane subtraction. While part segmentation uses the usual parameters (P1, $\beta_{\text{Tresh}} = 10°$), they are necessarily different for object segmentation: ($v = 0.75\,\text{cm}$, $s = 2\,\text{cm}$, $\beta_{\text{Thresh}} = 180°$, $n_{\text{filter}} = 2$).

(MDMPs) [22]. Here, specifically, we used a pick-and-place action, where the pose of the handle was calculated from its 3D shape obtained from the part partitioning algorithm. Note that in this case we have predefined grasps (grasp from top or grasp from side depending on the orientation of the handle) for specific objects. In Fig. 5 F we demonstrate a successful grasp on the handle for the heat gun (see supplementary video for grasping of other objects).

## IV. DISCUSSION

In this paper we presented a novel algorithm for the segmentation of 3D point clouds that can be used to partition objects into parts or to segment different objects from each other. The latter is a special case of the former. We demonstrated that our quite simple approach can compete with more complex state-of-the-art partitioning methods and that it performs equally well. We also presented an application of our algorithm in a robot scenario where the task of the robot was to grasp objects by their handle. In the following we will discuss our approach and relate it to existing methods.

### A. State of the Art

In general there exist several *bottom-up, data-driven* as well as *top-down, model-based* approaches. Several bottom-up approaches have recently been reported [24]–[26], which, however, do not reach the same level of performance compared to the method presented here. Most similar to our segmentation algorithm is the method from Moosmann *et al.* [11] sharing our thought to exploit convexities/concavities for segmentation using LIDAR data. The relatively noise-free LIDAR measurements allow direct use of 3D-points, different from RGB-D data. Our approach makes use of supervoxels [15] and a different set of criteria to gain robustness, which is not possible with the methods of Moosmann *et al.*. Recent top-down methods [12], [13], [16], [27] sometimes perform exceedingly well on similar benchmarks, but usually require quite a complex machinery to achieve the segmentation. It is, thus, remarkable that our very simple data-driven approach can compete with that of Richtsfeld *et al.* [12] as well as Ückermann *et al.* [13]. We take this as an indication of just how powerful the feature of local convexity is and suggest that it should also be considered as an important feature for future top-down approaches. In addition, our part partitioning bears a high similarity to the way a human would "describe" the parts of an object. We suggest that this might be a better starting point for "defining an object" (by composition from its parts) as compared to more arbitrary geometrical and surface model assumptions often found in the existing top-down approaches.

### B. A compositional view on affordances and objects

Why does our algorithm easily segment handles from the body of the object? The reason lies in the fact that almost all handles are designed so as to lead to a concave discontinuity relative to the body and this holds true also for many other handles, which we considered in our experiments (data not shown). Arguably the same is true for other manipulation-relevant parts like knobs, buttons, lids, etc., although the concavity might be hard to detect in RBG-D data using current cameras, which have quite a low resolution. There is no proof for this, but “looking around” seems to strongly support this speculation: most human-made manipulation-relevant parts seem to form a concave connection to the object body. Thus, the here presented algorithm will for all such cases produce a good guess for detecting the manipulation-relevant parts for a robot. The approach demonstrated in our robot experiments is, thus, a novel and efficient way to arrive at manipulation affordances for an artificial agent. In addition, parts as defined by our algorithm will many times form a convex figure and this figure will usually take a simple geometrical shape (cylinder, sphere, cube, torus, pyramid, etc.) which may be somewhat distorted and/or curved. Still, it should be possible to train classifiers for these object parts and thereby arrive at a compositional, generative approach for the (de-)construction and the understanding of complex object geometries. This is work in progress and we hope to be able to report on this in the near future.

## APPENDIX

### A. Formalism of the Segmentation Algorithm

Let us define local *convexity* and *concavity* for two neighboring surface patches as follows (see also Fig. 6 A). A **convex connection** of two linear surface patches is given when a straight line joining the patch centroids travels through regions that are **inside** the object, according to the direction of the patch normals. A **concave connection** is given when the line segment joining the patch centroids travels through free space, i.e., regions that are **outside**. In the following all steps of the algorithm are presented in detail.

*1) Building a Linear Patch Graph:* We build a linear patch graph using an approximation of the point cloud by finite linear patches with a neighborhood relation. An effective way to construct such an approximation is using a Supervoxel adjacency graph $G(V, E)$ [15], where each supervoxel $\vec{p}_i = (\vec{x}_i, \vec{n}_i, \dots), \vec{p}_i \in V$ is taken as a linear patch. In the following the centroid of patch $\vec{p}_i$ will be denoted as $\vec{x}_i$ and its normal vector as $\vec{n}_i$. Supervoxels allow feature specific weights to be set to respect boundaries in different features (e.g. color, normal direction). As we are interested only in geometric features, we set all weights to zero except the spatial weight $w_s = 1$ and the normal direction weight $w_n = 4$. Two parameter settings for the voxel size $v$ and supervoxel size $s$ were used (see Tab. II).

*2) Building a Convexity Graph:* Afterwards, we create a segmented graph model by classifying edges of the linear patch graph. To decide whether the connection $e = (\vec{p}_i, \vec{p}_j)$ between two patches is convex or concave/invalid we present one criterion for the basic convexity decision and an additional criterion increasing the robustness of the decision.

**Convexity Criterion (CC):** Consider two adjacent linear patches with centroids at the positions $\vec{x}_1, \vec{x}_2$ and normals $\vec{n}_1, \vec{n}_2$ as depicted in Fig. 6 A. Whether the connection between these patches is convex or concave can be inferred from the relation of the surface normals to the vector joining the two patch centroids.

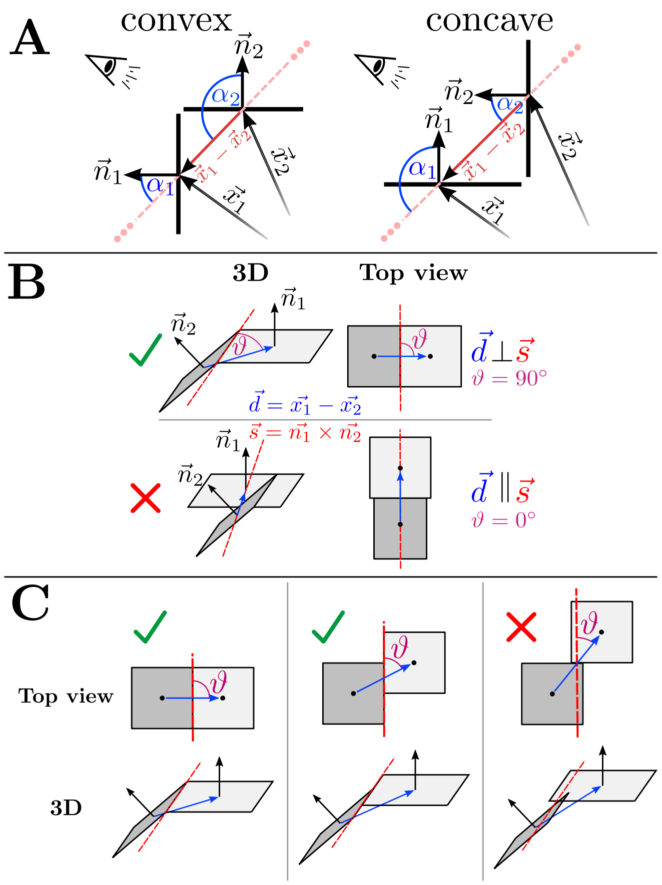

Fig. 6. **A)** Illustration of convex and concave connections between two linear patches. **B,C)** Illustration of the sanity criterion: **B)** A singular connection can be obtained by measuring the angle $\vartheta$ between the line of intersection $\vec{s}$ of the two planes represented by the patches and the vector $\vec{d}$, which connects the centroids of the patches. **C)** Change of the angle $\vartheta$ when the relative position of the patches is changed. The shared boundary is reduced by decreasing $\vartheta$, until a singular configuration is reached.

The angle of the patch normals to the vector $\vec{d} = \vec{x}_1 - \vec{x}_2$ joining the centroids can be calculated easily using the identity for the dot product $\vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}| \cdot \cos(\alpha)$ with $\alpha = \measuredangle(\vec{a}, \vec{b})$. One can see in Fig. 6 A, that $\alpha_1$ is smaller than $\alpha_2$ for *convex* connections. This can be expressed as:

$$\alpha_1 < \alpha_2 \Rightarrow \cos(\alpha_1) - \cos(\alpha_2) > 0 \Leftrightarrow \vec{n}_1 \cdot \hat{d} - \vec{n}_2 \cdot \hat{d} > 0,$$

where $\hat{d} = \frac{\vec{x}_1 - \vec{x}_2}{||\vec{x}_1 - \vec{x}_2||}$. Similarly, for a *concave* connection we get:

$$\alpha_1 > \alpha_2 \Leftrightarrow \vec{n}_1 \cdot \hat{d} - \vec{n}_2 \cdot \hat{d} < 0.$$

Note that the choice which patch is $\vec{x}_1$, i.e. in which direction the vector $\vec{d}$ points, is arbitrary and does not change the result. Also the criterion is still valid if the $\vec{x}_i$ are displaced, as long as they stay in the area of the patch.

To compensate for the noise in the RGB-D data, a bias is introduced to treat concave connections with very similar normals, that is

$$\beta = \measuredangle(\vec{n}_1, \vec{n}_2) = |\alpha_1 - \alpha_2| = \cos^{-1}(\vec{n}_1 \cdot \vec{n}_2) < \beta_{\text{Thresh}} \ ,$$

as convex, since those usually represent flat surfaces. Depending on the value of the threshold, concave surfaces with low curvature are seen as convex and thus merged in the segmentation. This behavior may be desired to ignore small concavities. This results in the definition of the convexity criterion $CC$:

$$CC(\vec{p}_i, \vec{p}_j) := \begin{cases} \text{true} & (\vec{n_1} - \vec{n_2}) \cdot \hat{d} > 0 \ \vee \ (\beta < \beta_{\text{Thresh}}) \\ \text{false} & \text{otherwise.} \end{cases} \tag{1}$$

We also experimented using *local convexity* as defined by Moosmann *et al.* [11] instead, but achieved lower performance, presumably because their criterion is susceptible to the noise present in the Kinect point clouds.

**Sanity criterion (SC):** In certain cases, the classification of the connection of two linear patches into convex or concave does not make sense. If the surface is discontinuous this is evidence for a geometric boundary. This means that the corresponding (originally potentially convex) connections should be identified and invalidated.

The vector $\vec{d}$ connecting the patch centroids and the line of intersection $\vec{s}$ of the planes represented by the linear patches can be calculated using $\vec{d}(\vec{x}_1, \vec{x}_2) = \vec{x}_1 - \vec{x}_2$ and $\vec{s}(\vec{n}_1, \vec{n}_2) = \vec{n}_1 \times \vec{n}_2$. As illustrated in Fig. 6 B, singular configurations can be identified by looking at the angle $\vartheta$ between $\vec{d}$ and $\vec{s}$. For two patches sharing one side of their boundary, the two directions are orthogonal. If the directions are parallel, the situation is clearly singular. Because the orientation of $\vec{s}$ is arbitrary, we define $\vartheta$ to be the minimum angle between the two directions, that is:

$$\begin{aligned} \vartheta(\vec{p}_1, \vec{p}_2) &= \min(\measuredangle(\vec{d}, \vec{s}), \measuredangle(\vec{d}, -\vec{s})) \\ &= \min(\measuredangle(\vec{d}, \vec{s}), 180^\circ - \measuredangle(\vec{d}, \vec{s})) \end{aligned} \tag{2}$$

The angle $\vartheta$ changes with the relative positions of the patches (see Fig. 6 C). If we start at a valid configuration where both patches have a common boundary edge ($\vartheta = 90^\circ$) and slide one patch along the boundary of the other, it can be seen that $\vartheta$ is decreased. Singular configurations occur for small values of $\vartheta$ and can thus be handled by introducing the threshold $\vartheta_{\text{Thresh}}$. For $\vartheta < \vartheta_{\text{Thresh}}$ the connection must be invalidated. Similar to the convexity criterion, this condition has to be relaxed for patches with very similar normals, to compensate for sensor noise. This is done by setting $\vartheta_{\text{Thresh}}(\measuredangle(\vec{n}_1, \vec{n}_2))$ to a sigmoid function of the angle between normals:

$$\vartheta_{\text{Thresh}}(\beta) = \vartheta_{\text{Thresh}}^{\max} \cdot \left(1 + \exp\left[-a \cdot (\beta - \beta_{\text{off}})\right]\right)^{-1}, \tag{3}$$

where $\beta = \measuredangle(\vec{n}_1, \vec{n}_2)$ is the angle between normals. We use the experimentally derived values $\vartheta_{\text{Thresh}}^{\max} = 60^\circ$, $\beta_{\text{off}} = 25^\circ$ and $a = 0.25$.

The sanity criterion $SC$ is then defined as

$$SC(\vec{p}_i, \vec{p}_j) := \begin{cases} \text{true} & \vartheta(\vec{p}_i, \vec{p}_j) > \vartheta_{\text{Thresh}}(\beta(\vec{n}_1, \vec{n}_2)) \\ \text{false} & \text{otherwise} \end{cases} \tag{4}$$

Note that the criterion is most effective if the aspect ratio of the considered patches does not deviate too much from one.

*3) Convex connected components:* The previously presented criteria are combined to the overall predicate $conv(\vec{p}_i, \vec{p}_j)$ defining local convexity:

$$conv(\vec{p}_i, \vec{p}_j) := \begin{cases} \text{true} & CC(\vec{p}_i, \vec{p}_j) \wedge SC(\vec{p}_i, \vec{p}_j) \\ \text{false} & \text{otherwise} \end{cases} \tag{5}$$

The last step of the segmentation is to find all components connected by convex edges (defined by the convexity predicate). This can be achieved by region growing. In the beginning, an arbitrary seed supervoxel is chosen as a start point. The segment label 1 is assigned to this supervoxel and this label is propagated over the graph with a depth search that is only allowed to grow over convex edges. Once no new supervoxel can be assigned to the segment, we increment the assigned label by 1 and choose a new seed supervoxel that has not been processed yet. We then propagate the new label in the same way as before and repeat the process until segment labels have been assigned to all supervoxels. Note that all our criteria are commutative, so the output of the region growing does not depend on the choice of the seeds.

*4) Noise filtering:* Concave boundaries are more reliably detected if the merging threshold $\beta_{\text{Thresh}}$ in the convexity criterion ($CC$) is low. For low thresholds, the segmentation will however suffer from small isolated patches that are created from noise present in the normal estimation. In these cases a post-processing step filtering the noise patches may improve quality. We implement a simple filter using the user selected filter size $n_{\text{filter}} \in \mathbb{N}_+$. For every segment $S_i$ of the segmentation, we check if it consists of at least $n_{\text{filter}}$ supervoxels. If a segment's size $|S_i|$ is smaller or equal to the filter size, we merge it with the largest neighboring segment. Filtering continues until no segments (that have neighbors) smaller than the filter size are present in the image.

TABLE II
PARAMETER SETS USED FOR PART SEGMENTATION.

| Parameter set | $v$ | $s$ | $n_{\text{filter}}$ |
|---|---|---|---|
| P1 | 0.5 cm | 2 cm | 3 |
| P2 | 0.75 cm | 6 cm | 1 |

## *B. Definition of Measures*

We define the ground-truth partition $G = \{G_1, G_2, \ldots, G_M\}$ as a set of human annotated regions $G_i$ and the segmentation $S = \{S_1, S_2, \ldots, S_N\}$ as a set of pixel regions $S_j$ of the same image. Furthermore $N_G := |G|$ is the number of ground-truth regions.

*1) Maximum Overlap:* For every object represented by a ground-truth region, the segment with the greatest overlap is taken as the best estimator. Thus, we define the maximum overlap for ground-truth region $G_i$ as $Ov_i = \max_{S_j}(|G_i \cap S_j|/|G_i \cup S_j|)$. The overall score, *Weighted Overlap* ($WOv$), is computed as a weighted average with respect to the size of the regions [16], [17]:

$$WOv = \frac{1}{\sum_i |G_i|} \sum_i |G_i| \cdot \text{Ov}_i. \tag{6}$$

Values range from 0 to 1, where 1 is considered the perfect segmentation with identical segmentation and ground-truth partition.

*2) True- and False-positive Scores:* Let us define true positive (correctly segmented) points $TP_i = G_i \cap S_i$ as overlap of both sets. Then we can define false positive points $FP_i = S_i \setminus TP_i$ and false negative points $FN_i = G_i \setminus TP_i$, which are exclusively assigned to one of the ground truth sets. Finally, average scores are defined as follows [13]:

$$tp = \frac{1}{N_G}\sum_i \frac{|TP_i|}{|G_i|},\ fp = \frac{1}{N_G}\sum_i \frac{|FP_i|}{|S_i|},$$
$$fn = \frac{1}{N_G}\sum_i \frac{|FN_i|}{|G_i|}. \tag{7}$$

*3) Over- and Under-segmentation:* Over-segmentation $F_{os}$ is the number of correctly assigned object pixels normalized by the number of all object pixels, whereas under-segmentation $F_{us}$ is the number of incorrectly assigned pixels normalized by the number of all object pixels [12]:

$$F_{os} = 1 - \frac{N_{true}}{N_{all}},\ \ F_{us} = \frac{N_{false}}{N_{all}}. \tag{8}$$

## *C. Evaluation problems*

It seems necessary to point out to the community that some problems arise when benchmarking 3D point cloud segmentation. In general, one wants to evaluate how good a segmentation algorithm performs in continuous 3D point cloud space. Since currently there are no evaluation methods for 3D data available, one has to fall back on conventional 2D methods. However, this leads to evaluation problems, which are mainly due to the way the ground-truth partition was created. 3D ground-truth data is usually simply constructed by transferring the labels from the RGB camera (2D image). An example of such a case is shown in Fig. 7 where a scene (panel A), its ground-truth (panel B) from the OSD data set and the segmentation result of our algorithm (panel C) are presented. Due to mismatches in the calibration between the depth and rgb sensor, the ground truth is inconsistent with the 3D geometry of the scene (this is the case for all scenes). Despite being virtually perfect from the view of the point cloud, the pictured segmentation achieves only a weighted overlap of $WOv = 91.3\%$. These problems should be kept in mind when interpreting the absolute values in Fig 4.

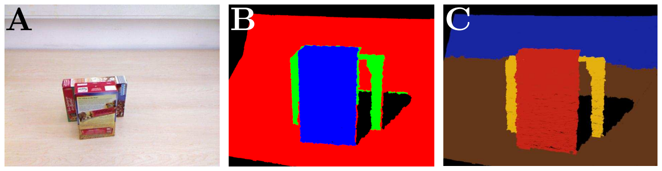

Fig. 7. **A)** Original image, **B)** ground-truth from point cloud perspective from the OSD dataset and **C)** segmentation result of our method.

## References

[1] I. P. Howard, “Alhazens neglected discoveries of visual phenomena.” *Perception*, vol. 25, pp. 1203–1217, 1996.

[2] A. I. Sabra, *The optics of Ibn al-Haytham: Books I-III: On direct vision. (English translation and commentary by Sabra, A.I.)*. London: Warburg Institute, 1989.

[3] J. J. Koenderink, “What does the occluding contour tell us about solid shape?” *Perception*, vol. 13, pp. 321–330, 1984.

[4] L. M. Vaina and S. D. Zlateva, “The largest convex patches: A boundary-based method for obtaining object parts,” *Biological Cybernetics*, vol. 62, no. 3, pp. 225–236, 1990.

[5] P. L. Rosin, “Shape partitioning by convexity,” *IEEE Trans. Systems, Man, and Cybernetics, part A*, vol. 30, pp. 202–210, 2000.

[6] T. Matsuno and M. Tomonaga, “An advantage for concavities in shape perception by chimpanzees (pan troglodytes),” *Behavioural Processes*, vol. 75, no. 3, pp. 253–258, 2007.

[7] A. D. Cate and M. Behrmann, “Perceiving parts and shapes from concave surfaces,” *Attention, Perception, & Psychophysics*, vol. 72, no. 1, pp. 153–167, 2010.

[8] M. Bertamini and J. Wagemans, “Processing convexity and concavity along a 2-D contour: figureground, structural shape, and attention,” *Psychonomic Bulletin & Review*, vol. 20, no. 2, pp. 191–207, 2013.

[9] R. Hoffman and A. K. Jain, “Segmentation and classification of range images,” *IEEE Tran. Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 608–620, 1987.

[10] K. Wu and M. Levine, “3d part segmentation using simulated electrical charge distributions,” *IEEE Tran. Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, pp. 1223–1235, 1997.

[11] F. Moosmann, O. Pink, and C. Stiller, “Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion,” in *2009 IEEE Intelligent Vehicles Symposium*, 2009, pp. 215–220.

[12] A. Richtsfeld, T. Morwald, J. Prankl, M. Zillich, and M. Vincze, “Segmentation of unknown objects in indoor environments.” in *IROS*. IEEE, 2012, pp. 4791–4796.

[13] A. Ückermann, R. Haschke, and H. Ritter, “Real-time 3D segmentation of cluttered scenes for robot grasping,” in *IEEE-RAS International Conference on Humanoid Robots*, 2012.

[14] A. Karpathy, S. Miller, and L. Fei-Fei, “Object discovery in 3d scenes via shape analysis,” in *International Conference on Robotics and Automation (ICRA)*, 2013.

[15] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter, “Voxel cloud connectivity segmentation - supervoxels for point clouds,” in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, june 2013.

[16] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, “Indoor segmentation and support inference from rgbd images,” in *ECCV*, 2012.

[17] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert, “Recovering occlusion boundaries from a single image,” in *ICCV*, 2007, pp. 1–8.

[18] Kuka Robot Systems. [Online]. Available: http://www.kuka-robotics.com

[19] M. Schoeler, S. C. Stein, A. Abramov, J. Papon, and F. Wörgötter, “Fast self-supervised on-line training for object recognition specifically for robotic applications,” in *VISAPP 2014 - 9th International Conference on Computer Vision Theory and Applications*, 2014.

[20] E. E. Aksoy, A. Abramov, J. Dörr, N. Kejun, B. Dellen, and Wörgötter, “Learning the semantics of object-action relations by observation,” *The International Journal of Robotics Research*, vol. 30, no. 10, pp. 1229–1249, 2011.

[21] M. J. Aein, E. E. Aksoy, M. Tamosiunaite, J. Papon, A. Ude, and F. Wörgötter, “Toward a library of manipulation actions based on semantic object-action relations,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and System*, 2013, p. in press.

[22] T. Kulvicius, K. J. Ning, M. Tamosiunaite, and F. Wörgötter, “Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting,” *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 145–157, 2012.

[23] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[24] T. Rabbani, F. A. van den Heuvel, and G. Vosselmann, “Segmentation of point clouds using smoothness constraint,” in *IEVM06*, 2006.

[25] M. Johnson-Roberson, J. Bohg, M. Bjrkman, and D. Kragic, “Attention-based active 3d point cloud segmentation.” in *IROS*. IEEE, 2010, pp. 1165–1170.

[26] E. Castillo, J. Liang, and H. Zhao, *Point cloud segmentation via constrained nonlinear least squares surface normal estimates*. Springer, 2013, ch. 13.

[27] Z. Jia, A. Gallagher, A. Saxena, and T. Chen, “3d-based reasoning with blocks, support, and stability,” in *CVPR*, 2013.