

# Bootstrapping the Semantics of Tools: Affordance Analysis of Real World Objects on a Per-part Basis

Markus Schoeler and Florentin Wörgötter

**Abstract**—This study shows how understanding of object functionality arises by analyzing objects at the level of their parts where we focus here on primary tools. First, we create a set of primary tool functionalities, which we speculate is related to the possible functions of the human hand. The function of a tool is found by comparing it to this set. For this, the unknown tool is segmented, using a data-driven method, into its parts and evaluated using the geometrical part constellations against the training set. We demonstrate that various tools and even uncommon tool-versions can be recognized. The system “understands” that objects can be used as makeshift replacements. For example, a helmet or a hollow skull can be used to transport water. Our system supersedes state-of-the-art recognition algorithms in recognition and generalization performance. To support the conjecture of a possible cognitive hand-to-tool transfer we analyze, at the end of this study, primary tools by also incorporating tool-dynamics. We create an ontology of tool functions where we find only 32 of them. Being such a small set this would indeed allow bootstrapping tool-understanding by exploration-based learning of hand function and hand-to-tool transfer.

**Index Terms**—Function analysis, object recognition, tool bootstrapping, tool ontology.

## I. INTRODUCTION

THE complexity of shapes and arrangements of all objects, which we encounter every day, as well as just their sheer number, is humongous. Most of them, today, are human-made. But even during the advent of humankind, some million years ago, early hominids were faced with very many different natural objects in their environment. Different from all other animals they were able to handle this complexity and began “to make sense of them” arriving at an early semantics of objects and their potential use (e.g., as tools). Starting from this, supervision, teaching, and communication—hence cultural inheritance—allowed us to build our complex world. Still, it is puzzling how the process of understanding objects (tools) can be bootstrapped. The complexity of the world of natural objects seems just too high!

Manuscript received March 30, 2015; revised August 18, 2015; accepted September 26, 2015. Date of publication October 07, 2015; date of current version June 08, 2016. This work was supported by the European Communitys Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under Grant 270273, Xperience.

The authors are with the Third Physical Institute, University of Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany (e-mail: mschoeler@gwdg.de; worgott@gwdg.de).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TAMD.2015.2488284

Essentially we would like to speculate that our hand allows bootstrapping the understanding of basic tools. The association of possible hand-shapes plus the way we use these different shapes<sup>1</sup> leads to a rather small set of options, which can be ontologically ordered into a manageable system of tools. Hence, the claim is that understanding your hand allows transferring this understanding without too much effort to a set of primary tools. The set of arising options from using these tools is already rich enough to entail a wide variety of tool-induced changes at the target substrate(s). As a consequence we want to argue that the understanding of the thus-induced cause-effect relations may well have been a powerful drive towards cognitive complexity.

The current study addresses this issue in two interlinked parts: On the one hand, we provide a rigorous computer-vision algorithm that makes use of these speculations to perform probabilistic reasoning about the potential roles of objects. Hence, we show that our speculations might not be entirely unfounded. On the other hand, we extend our idea in a discussion on how this bootstrapping problem could have been solved and how an early ontology of tools and their uses could have been generated. This matter can be debated and is certainly opening the door to controversies.

It seems generally agreed that the fundamental properties of monkeys’ and especially hominids’ hands (e.g., opposing thumb, third metacarpal styloid process<sup>2</sup>) and the fact that the hands became free to be used as tools, as soon as we started to walk on two feet, amplified cognitive development [1]. Much less is known with respect to the possible mental transfer of hand-function to tool-function. More advanced primates show indeed a much wider variety of actions performed with their hands on objects [2]. It is, thus, indeed tempting to assume that at some point such a mental transfer might have taken place, for example realizing that using a stick instead of your finger makes a better borer, using a seashell instead of your cupped hand makes a better cup, etc. While it is generally agreed that already infants at the age of 1–2 years are able to understand which geometric properties of objects are important for certain tasks (like a rigid stick for pulling an objects closer) [3], as far as we see there are no studies in archeology, primatology, or child developmental psychology that specifically address the issue of a mental hand to tool transfer, such that this hypothesis has not been much considered.

The current study, though, adopts this hypothesis and we now introduce a computer vision based system that makes predic-

<sup>1</sup>For example, a fist can be used as a hammer, a single finger as a borer, a flat hand as a paddle.

<sup>2</sup>By this the hand can lock better to the wrist, allowing for larger pressure to be applied to wrist and hand while grasping.

Function	Handshape	Tool replacements
Contain		
Cut		
Hit		
Hook		
Poke		
Sieve		

Fig. 1. Primitive functions, their associated hand shapes and tools meant to substitute and improve efficiency.

tions about the possible functions of simple tools. This system uses “labeled data” for training. One central claim, related to our above speculations, is that there are only very few basic tool functions existing which are related to the shape and function of the human hand. Hence, understanding your hand-function provides you with all the labels for training the system “free of additional charge” (see Fig. 1). This argument is much strengthened in Section IX where we show how to structure such a system ontologically, thereby demonstrating its rather low complexity, which can well facilitate the here-suggested bootstrapping of tools by hands.

One problem is the fact that objects (also tools) consist of parts, where some contribute to the fundamental tool function (cup-container), while others might subserve a different function (cup-handle) or just be decorative. Also the number of parts might be different for the same tool-type (forks with different numbers of tips, cups with one or more handles), which do not fundamentally alter the functionality of the tool. Thus, we design a system that considers object-parts and part combinations to assign (tool-)functions. The advantage of this is that one does not have to train the system to wantonly different individual objects (an artist’s rendering of a cup with 8 handles is still a cup, see Fig. 1). As soon as the system has learned the fundamental requirements for “being a cup” (rather “being a small container”) these details do not matter anymore and categories are formed across objects with vastly different visual appearance.

We now, first, describe the algorithmic aspects and, second, in Section IX extend the speculative part of this paper by introducing our ontology of tools bootstrapped by hand shape and function.

## II. OVERVIEW

The core idea of this work is the assumption that functional objects should be described by their parts and part relations. This holds for tools, which are mostly considered in this study, but also for most, if not all, other objects. For example an object

for cutting may consist of a blade and a handle, an object for hitting has a handle and a head most of the time. But objects for the same function can have different number of parts. Moreover the way parts are attached to each other plays an important role for an object’s functionality. The objects in the lower part of Fig. 7 consist of the same parts, but can be used differently or not at all because of their part-to-part relations. This leads to huge intraclass variance in case of functional categories, which, as we show in this paper, can be faced by analyzing objects on the part level. Thus, our algorithm uses features of the different parts, but also their geometrical configurations relative to each other (“relative pose”) to define an object.

Fig. 2 shows all algorithmic components. It looks complex, but there are only 3 blocks existing: (A) preprocessing (yellow), (B) object Signature extraction (red), and (C) object similarity calculation (green). In the middle, we show the object signatures being the output of block B and the input to block C. Small section numbers refer to the sections where the different algorithmic components are described in this article.

Preprocessing (yellow) is the step where objects are being segmented into their parts employing a dedicated algorithm, *constrained planar cuts*, (CPC) [4] which uses the transition between convex and concave 3-D-image structures as indicator for a potential part-cutting plane.

Object signature extraction (red) contains three components: left, extraction of the individuals signatures of all parts; right, extraction of the pose relations between parts; middle, generation of a graph that contains at its nodes the part-signatures whereas the edges represent the pose-relations between the two parts at the connected nodes (see example graphs in the blue box below). We use the scores of a support vector machine to create second order part signatures. Pose signatures fundamentally consist of how parts are aligned (Alignment) and how they are attached (Attachment) to each other.

Object graphs of two or more objects—here object X, solid arrows and Y, dashed arrows—can now be compared (Object Similarity Calculation, green). This requires an association algorithm [see Fig. 2(b) Top], because it is *a priori* unknown which node in graph X corresponds to which in graph Y. All these checked (red arrows), we can finally calculate the actual object similarity (dark blue box).

## III. RELATED WORK

In this section, we discuss related work found in the literature and focus on the relevant biological as well as computer vision aspects.

### A. Biological Background

The idea of using subentities and their relations to describe objects is not new. Most prominent has been the suggestion to subdivide objects using so-called “Geons” [5]. Biederman [6] put it this way: In analogue to same letters forming different words, relations among the same set of parts (geons) can form different objects. Geons, however, are more related to abstract geometrical entities than to functional object parts and the here obtained parts are clearly “more than geons.” In a different study [4] we had shown that the here used method (CPC algorithm, see Section IV-A) is able to obtain parts, which correspond to meaningful entities. Those have their own function, which can be

functionally named. Hence, we usually have a semantic (nameable) cognitive concept for these parts, which does not hold for geons (they can be named but these names carry no functional meaning).

The importance of parts for the visual system is additionally supported by the experiments of Biederman [7] which show that objects can be identified as long a single parts of an object can be recognized. As drivers for the part segmentation in a human brain Richards and Hoffman [8], [9] indicated concavities (cusps) between parts, which is the mechanism that we also use in the CPC algorithm to obtain the parts for this study.

In addition to parts as such, it also seems that the geometrical relation between them (their pose relation) does indeed carry very fundamental information for us, because part-pose is represented in a certain brain area, the lateral occipital (LO) area [7]. This coding is independent of the *actual parts* or their local features, which is supported by the results of Behrmann *et al.* [10] noting that a male patient who suffered a left occipital-temporal lesion could distinguish parts, but no part-to-part pose relations.

## B. Computer Vision

A wide variety of (computer vision) approaches exist which try to classify and categorize objects, however, here we discuss only those approaches that use secondary constraints (e.g., context, part-relations, semantics, etc.), which are related to the current study. We do not discuss less related approaches that train classifiers “just on objects as such” or only with minimal constraints. The latter are nowadays many times successful in the context of “deep learning” [11], but quite unrelated to this approach as they do not generalize well to completely new (e.g., artistic) versions of tools. Several types of models exist here, that try to address the “object problem” from a higher-level perspective, which shall be discussed next.

*Scene segmentation* and *object partitioning* approaches can be categorized into three groups: First, purely bottom-up approaches which often use hierarchies to create a rank order that builds bottom-up from small local superpixels to higher level semantic regions [12], [13]. Second, purely top-down approaches employing multiscale sliding window detectors [14], consequently progressing to finer grained segmentations using the concept of object parts [15]. Third, a combination of bottom-up hierarchy building and top-down object and part-detectors [16]–[18]. While pure and partial top-down approaches generally yield good results they need trained classifiers, thus can only be applied to low intraclass variance categories or known objects. Both is not the case in the here presented work. Clustered viewpoint feature histograms [19] also split objects into parts by clustering smooth surface, consequently parts are patches rather than complete functional entities. In contrast to this, in our former works [20] *locally convex connected patches* (LCCP) as well as [4] *constrained planar cuts* CPC() we presented model- and learning-free bottom-up 3-D point cloud segmentation algorithms, which showed state-of-the-art results in several benchmarks. Both allow to split objects into nameable parts by analyzing concavities in the object (resembling the findings from [8]). While *LCCP* is better suited for bottom-up object segmentation, *CPC* has proven itself very powerful in segmenting parts of objects.

Therefore we use *CPC* for generating the object partitions (see Fig. 3 for some examples).

*Contextual reasoning models* have shown an increase in performance for object recognition in scenes [21]. Some authors extended contextual models to activity recognition [22], [23] by analyzing objects and pose of objects in 2-D images. For example a bottle and a human head in a certain relation show a drinking activity. In analogy to this we consider parts as the atoms in our recognition framework and the full object as the scene. Farhadi and Sadeghi [24] showed that combining two categories to context specific categories (phrasals) improves detection rates on images. For instance a detector trained on the interaction “person riding horse” has better performance than two detectors trained on “persons” and “horses” separately. While this helps to limit the intraclass variance, it exponentiates the number of potential classes. We on the other hand aim at creating general super-categories, spanning many traditional classes. We deal with the high intraclass variance by describing objects by their parts, part-to-part relations and part graphs. The way we combine these parts (atoms) allows for different functionalities (scenes). Consequently, we also employ context by not classifying each part separately, but giving the whole object a function score and projecting it back to retrieve the part labels. For example the head of an unknown hammer may look very similar to the handle of a known saw. A decision in the context of the other parts, however, allows for correctly labeling the part.

*Mixture Models and Topic Models*, with Latent-Dirichlet-Allocation [25] and probabilistic Latent Semantic Analysis [26] being among the most popular, have proven to be successful in 2-D object recognition. They typically model a category as a mixture of subcategories. This helps to cope with intraclass variations by partitioning the data into smaller clusters with lower variability. However, the subcategories are not located in the image, nor necessarily represent semantic entities. Our approach on the contrary names and locates the parts. Similar to the way topics reduce variability within a category, our part-decomposition reduces the variability of an object allowing for comparison and generalization between vastly different objects.

Other *part-based recognition systems* focus on nameable parts, too. Good performance has been achieved in face detection [15]), human-pose estimation [27] or car classification by partitioning into front, middle and rear parts [28]. Related to our approach is the work of Tenorth *et al.* [29]. They approximate *containing objects* by primitive geometrical parts like spheres, planes and cylinders and fit CAD models which requires similar objects in the training set. Still, all approaches focus on specific domains where a preselected pool of parts is available. Our approach is more generic in the sense that we do not require an object to have a set number of parts or being made of simple geometric primitives. Others introduced pictorial frameworks [30], [31], which split objects into part templates together with geometric constrains on part-to-part relations (using for example spring models). This has successfully been applied to human pose estimation, where parts are connected at fixed locations. As objects do not follow this constrain this method is not applicable to our problem. Shapira *et al.* [32] proposed a method for contextual part analogies. It bases on a part-to-part distance measure propagated through the part connectivity

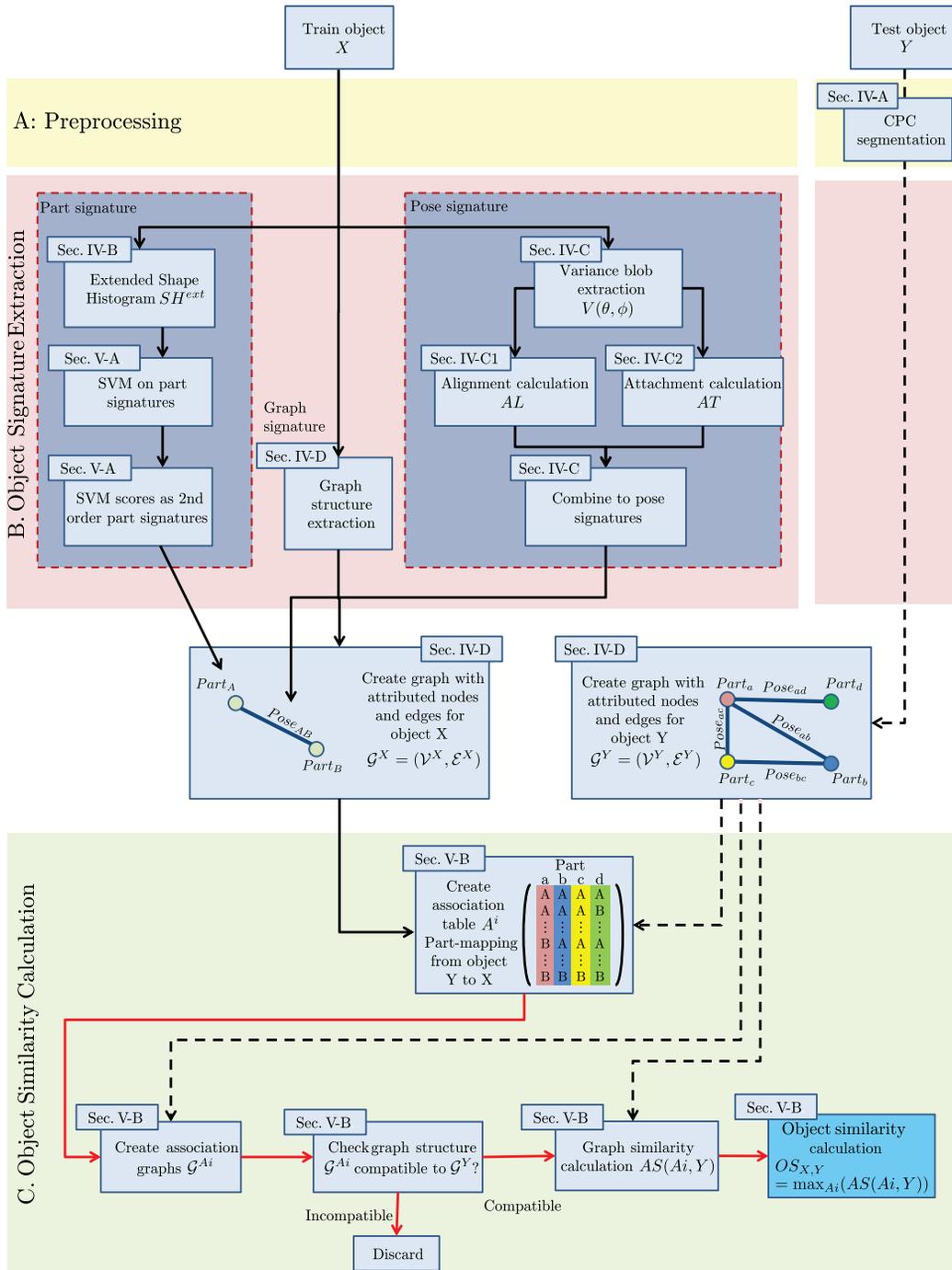


Fig. 2. Algorithm overview of the function analyzer: (A) preprocessing; (B) object signature extraction; (C) object similarity calculation.



Fig. 3. CPC algorithm applied to three different functional objects. Left: Femur model from shapes.aimatshape.net; Middle: Hatchet model from <http://tf3dm.com>; Right: WhiteCup scan from the KIT ObjectModels Web Database.

graph. It relies on characteristic appearance of parts, because they do not store the information how parts are attached to their neighbors (point of attachment and relative pose). We capture this information in our pose-signature and show that it is important to recognize function of tools (see Table. I).

In *Attribute based approaches* [33] classes are expressed as a mixture of human-specified high-level descriptions, allowing the classifier to be applied to new classes with known attributes. Our approach, too, can deal with novel object classes like a saw when other objects for cutting are known. In contrast to [33], we do not need to provide attributes, but summarize all objects allowing for certain functions into super-categories. Combined

TABLE I

CLASSIFICATION ACCURACY IN % FOR THE BENCHMARKS M1 AND M2 USING DIFFERENT PIPELINES. *BASE* ARE THE BASELINE CLASSIFIERS TRAINED AND TESTED ON FULL OBJECTS. *PART* ARE CLASSIFIERS TRAINED ON PARTS. *PART+POSE* SYSTEMS ADDITIONALLY MAKE USE OF THE GRAPH STRUCTURE AS WELL AS THE POSE SIGNATURES. THE BEST RESULTS (MARKED BOLT) IN BOTH BENCHMARKS ARE ACHIEVED BY THE  $SH_{4,4}^{Ext}$  CLASSIFIER USING *PART* & *POSE* SIGNATURES

	Pipeline	M1	M2
<i>Base</i>	SHOT-A	77.71	63.19
	SHOT-C	65.35	59.38
	ESF	75.97	60.42
<i>Part-only</i>	SHOT-A	85.83	81.18
	SHOT-C	67.50	72.01
	ESF	70.90	64.44
	$SH_{20,20}$	77.48	66.39
	$SH_{20,20}^{Ext}$	79.65	70.62
	$SH_{4,4}$	77.22	73.19
	$SH_{4,4}^{Ext}$	77.92	73.89
<i>Part &amp; Pose</i>	SHOT-A	89.58	87.08
	SHOT-C	78.06	83.26
	ESF	79.72	76.46
	$SH_{20,20}$	85.69	78.47
	$SH_{20,20}^{Ext}$	86.32	82.99
	$SH_{4,4}$	87.08	88.54
	$SH_{4,4}^{Ext}$	<b>90.42</b>	<b>88.68</b>

with the fact that we can assign multiple functions to objects, we also pave the way to makeshift tool replacements. This is an interesting concept as it allows robots for instance to bootstrap alternative solutions to problems.

#### IV. OBJECT DESCRIPTION AT THE PART LEVEL

All input data are full 3-D point clouds either sampled from publicly available mesh models/scans or using the procedural shape generator from the Point Cloud Library<sup>3</sup>.

##### A. Part Segmentation using the CPC Algorithm

To segment 3-D point clouds we use our *Constrained Planar Cuts* algorithm proposed in [4]. It is based on the idea that convex surfaces, separated by concave boundaries, play a crucial role for the perception of objects and their decomposition into parts. The algorithm starts by approximating a scene with surface patches using our supervoxel algorithm [34]. Next *CPC* analyses the connection between adjacent surface patches and classifies them as either concave or convex. The algorithm uses concave connections to propose planar cuts through as many concavities as possible, while minimizing the number of convex connections cut. Details of this procedure can be found in [4]. Some examples for the object to part segmentation are shown in Fig. 3.

##### B. Part Signatures

The goal of this section is to introduce methods to arrive at a characteristic description—called part signature—of the individual object parts. For this, visual features of the parts need to be extracted.

For signature generation, in this work we investigate *signature of histograms of orientations* (SHOT) features [35], *ensemble of shape functions* (ESF) features [36] as well as an ex-

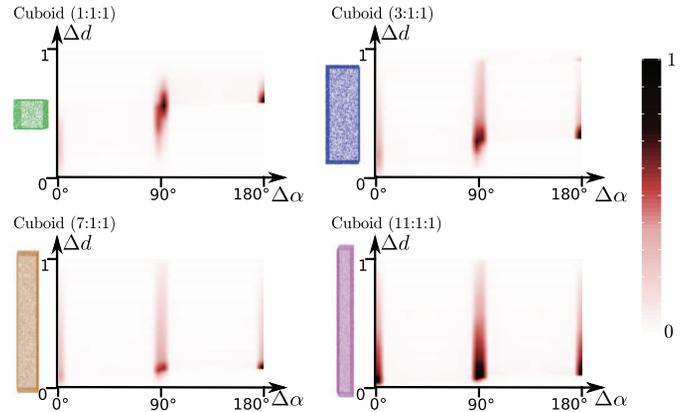


Fig. 4. Aspect Ratio dependency of the basic shape histogram  $SH$  for a cuboid with  $N_d = N_\alpha = 80$ .

tension of the two dimensional shape histogram proposed by Mustafa *et al.* [37]. Both SHOT and ESF features are geometry based and have shown state-of-the-art performance in object classification tasks [38]. All features use 3-D information as their input.

SHOT features are extracted at local feature points and oriented to the so-called Local-Reference-Frame. They divide the local neighborhood of a feature point using a spherical grid into 32 cells. Within each cell SHOT accumulates the neighboring points according to their normal directions into 11 bins. This results in a  $11 \cdot 32 = 352$  dimensional histogram. For generating global SHOT signatures we followed the procedure proposed in [39], which calculates one SHOT descriptor for the object's centroid. We call them SHOT-C in this paper. Additionally, we create global SHOT descriptors by calculating a descriptor for each point and averaging all. We denote them as SHOT-A. As neighbors for the feature and Local-Reference-Frame calculation we consider all other points in the cloud.

ESF is a global point cloud descriptor which encodes the geometric information in a point cloud using 10 concatenated 64-bin histograms, including distance between two random points, area of triangle formed by three random points, and angle formed by three random points from the point cloud. This results in a  $10 \cdot 64 = 640$  dimensional histogram.

Additionally, we extend the two dimensional shape-histograms proposed by Mustafa *et al.* [37]. They are generated by iterating through all possible point pairs within a part. For each pair we calculate the distance between points  $d$  and angle difference between point normals  $\alpha$  in degrees. The variable  $d$  is normalized in such a way that 1 corresponds to the biggest distance determined. Both measures are then discretized by binning and added to a  $N_\alpha \cdot N_d$  dimensional histogram with  $N_d$  and  $N_\alpha$  describing the number of distance and angle bins, respectively. We call these histograms *basic shape histograms*  $SH$ . Example basic shape histograms showing the aspect ratio dependency for a cube are shown in Fig. 4.

We extend the histogram by additionally checking if the pair is convex- or concave-connectable. Points are convex-connectable if there could exist a closed surface containing both points which is purely convex (see also Fig. 5). If there is no such surface, we call the pair concave-connectable. We can check this for two points  $i$  and  $j$  with normals  $\vec{n}_i$  and  $\vec{n}_j$  and

<sup>3</sup>[https://github.com/mschoeler/pc/archiveshape\\_generator.zip](https://github.com/mschoeler/pc/archiveshape_generator.zip)

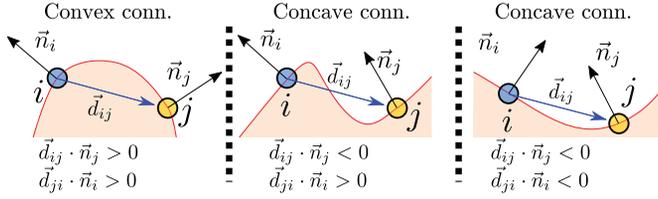


Fig. 5. Showing an example of convex versus concave connectable point pairs introduced in (1). The red lines denote a potential surface connecting both points.  $\vec{d}_{ji} = -\vec{d}_{ij}$ .

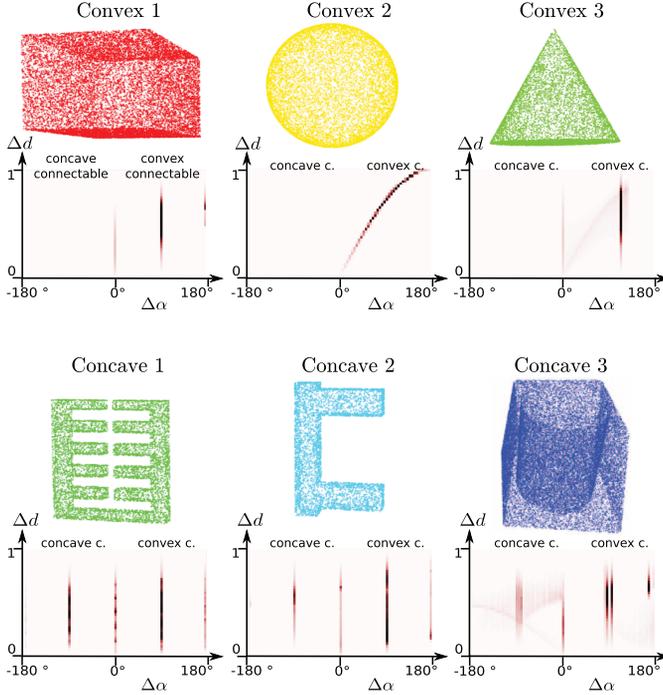


Fig. 6. Here we show three convex (top) and three concave shapes (bottom) together with their extended shape histogram  $SH^{Ext}$ . Each pair of points in the convex shapes are convex connectable (positive angles). This is why we do not have entries for concave connectable pairs (negative angles) in contrast to the concave shapes. See Eq. (1).

displacement vector  $\vec{d}_{ij} = -\vec{d}_{ji}$ , pointing from  $i$  to  $j$ , using the following equation:

$$\text{Connectable}(i, j) := \begin{cases} \text{Convex} & \vec{d}_{ij} \cdot \vec{n}_j > 0 \text{ and} \\ & \vec{d}_{ji} \cdot \vec{n}_i > 0 \\ \text{Concave} & \text{otherwise.} \end{cases} \quad (1)$$

This extension is similar to the inside/outside classification used in ESF-features. The extended shape histogram accordingly results in a  $2 \cdot N_\alpha \cdot N_d$  dimensional histogram. Instead of showing 2 histograms, we use negative difference-values to denote the concave connectable pairs. Naturally, point pairs on convex objects are always convex connectable. This is why only objects with concavities have pairs with negative angles (see Fig. 6). The usage of extended shape histograms is denoted by the superscript  $SH^{Ext}$ .

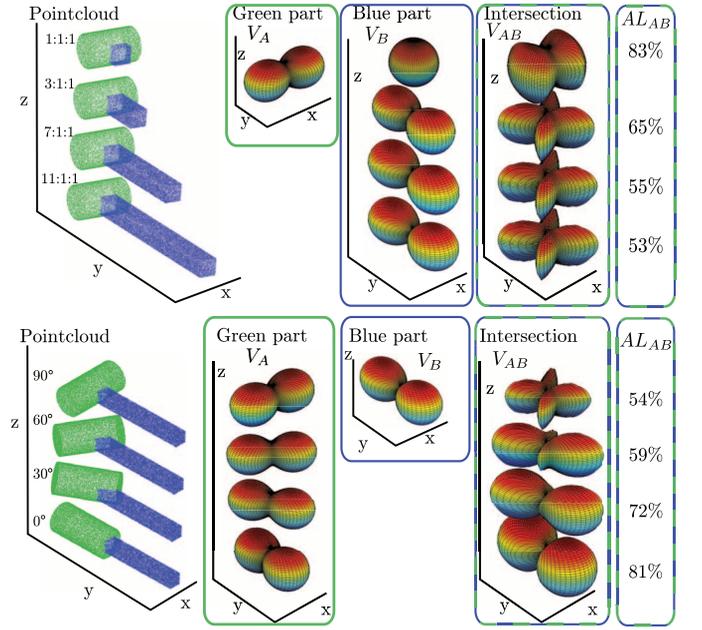


Fig. 7. Visualization of the influence of changing part shape (top) and relative orientation (bottom) on the variance blobs, intersection blobs, and the alignment number.  $V_A, V_B$ : Variance blobs for the green parts (A) and the blue parts (B). The bins of both variance blobs sum up to 1, i.e.,  $BS(V_A) = BS(V_B) = 1$  (see Eq. (3)).  $V_{AB}$ : Intersection of  $V_A$  and  $V_B$  using Eq. (4).  $AL_{AB}$ : Calculated Alignment number by summing over all bins of  $V_{AB}$ , Eq. (5), i.e.,  $AL_{AB} = BS(V_{AB}) \leq 1$ . Please note that  $AL_{AB}$  only reaches 100% if both parts are in-line (like the  $0^\circ$  object at the bottom) and have the exact same aspect ratio.

### C. Pose Signature

The second required descriptor is addressing the question how parts are attached to each other, for which we calculate the so-called pose signature consisting of an alignment and an attachment component. Hence, we define

- the way parts are rotated in respect to one another: the alignment  $AL$ ;
- the locations at which parts are attached to one another: the attachment  $AT$ .

Whenever we write “pose signature” it means both properties.

1) *Alignment  $AL$* : We define the alignment number between two parts A and B,  $AL_{AB}$ , as a scalar in the range of 0 to 1. It should change as soon as a part is rotated in respect to the other one. Still, to preserve overall rotational invariance,  $AL$  is not allowed to change, if identical transformations are applied to both parts. The more the parts are in-line, the bigger  $AL$  should be. One obvious solution to this would be to calculate each part’s axis of elongation (for example by applying principal component analysis (PCA) on the part’s point cloud and using the first principal component axis). The angle between these axes could then be used to calculate  $AL$ . This, however, only works for parts which are elongated enough, which many times is not the case. An example is shown in Fig. 7 top: The more the blue part is scaled down, the harder it is to define a proper axis of elongation. This would lead to a discontinuous jump in the alignment number close to the 1:1:1 case.

In the following, we show how to circumvent this problem and how we define  $AL$ : First, we represent each part with a so-called *variance blob*  $V$ . The *variance blob* represents the

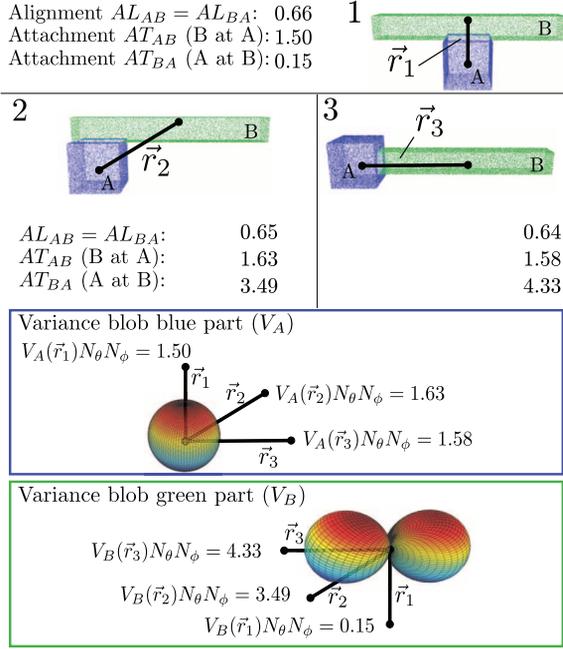


Fig. 8. Three objects consisting of identical parts. The blue part A (cube) is shifted from the long end of the green part B (elongated cuboid) to the middle. We did not draw the directions for the vectors  $\vec{r}$  since values on opposite sites of the variance blobs are equal (the variance calculation ignores orientation). The parts are not rotated in respect to each other, which leads to a stable alignment number. In contrast to this the attachment of the Part A (blue) in respect to Part B (green) (A at B) changes significantly. Since Part A (blue) is a cube, the B at A attachment is not influenced nearly as much.

variance of the point distribution of a part in all possible directions. To denote a direction  $\vec{r}$  in 3-D we use spherical coordinates, i.e.,  $\vec{r}(\theta, \phi)$ .  $\theta$  denotes the polar angle (measured from the pole), and  $\phi$  represents the azimuthal angle.  $\vec{r}$  is L2-normalized, such that  $\|\vec{r}\| = 1$ . The variance of the point cloud along a direction  $\vec{r}(\theta, \phi)$  is calculated using

$$V(\theta, \phi) = \frac{1}{N_p} \sum_{i=1}^{N_p} ((\vec{x}_i - \vec{\mu}) \cdot \vec{r}(\theta, \phi))^2. \quad (2)$$

Here  $N_p$  is the number of points in the part's point cloud,  $\vec{x}_i$  is the coordinate of the  $i$ th point and  $\vec{\mu}$  is the centroid of the part's point cloud.

If  $\vec{r}$  points in direction of the part's elongation, it results in a bigger variance as compared to a direction which is, for example, perpendicular to the elongation.

To make calculations feasible we only calculate the variance at discrete uniform steps using  $N_\theta$  and  $N_\phi$  bins. Therefore, this results in a two dimensional histogram with  $N_\theta \cdot N_\phi$  entries in total.

Visualizations of these *variance blobs*,  $V(\theta, \phi)$ , can be done using spherical plots as shown in Figs. 7 and 8. In the plots the variance is denoted by the radius of the surface from the origin. In case of constant variance this results in a constant radius in all directions, thus in a perfect sphere (e.g., the blue part in Fig. 8).

A *variance blob* represents the variance in the Euclidean space of the point cloud. This is why the aspect ratio / elongation of a part is directly visible in the elongation of its *variance*

*blob*. Exact shape detail (like round, cylindrical, etc.) is, in contrast, ignored.

We further want to ignore the total size of the parts, as this should not influence the alignment number between two parts. Therefore, we now normalize the variance blobs. Because we calculate the variance at uniform steps in spherical coordinates, the size of a bin (in Euclidean space of the point cloud) scales with the sine of the polar angle,  $\sin(\theta)$ . This effect is visible in smaller bin sizes close to the poles of the *variance blobs*.

Taking the changing bin size into account we can define the sum of all bins of a *variance blob*,  $BS(V)$ , as

$$BS(V) = \sum_{i=1}^{N_\theta \cdot N_\phi} \sin(\theta^i) V^i \quad (3)$$

with  $i$  iterating through all bins of the *variance blob* and  $V^i$  and  $\theta^i$  denoting the variance and polar angle of the  $i$ th bin.

Each part's variance blob  $V$  is normalized such that  $BS(V) = 1$ . To calculate  $AL_{AB}$  between part A and B, we use the histogram intersection similarity between  $V_A$  and  $V_B$ . This is done by calculating the intersection of each bin of *variance blob A* with each corresponding bin in *variance blob B* such that for the  $i$ th bin of the *intersection blob* we can write

$$V_{AB}^i = \min(V_A^i, V_B^i). \quad (4)$$

Finally, we calculate the alignment number  $AL_{AB}$  as the sum of the bins of the *intersection blob*, again taking the varying bin size into account, by using (3) and (4)

$$\begin{aligned} AL_{AB} &= BS(V_{AB}) = \sum_{i=1}^{N_\theta \cdot N_\phi} \sin(\theta^i) V_{AB}^i \\ &= \sum_{i=1}^{N_\theta \cdot N_\phi} \sin(\theta^i) \min(V_A^i, V_B^i). \end{aligned} \quad (5)$$

An example how the alignment  $AL$  changes when rotating one part in respect to another is depicted in Fig. 7 bottom. Since the min operation is commutative, the alignment number is commutative as well:  $AL_{AB} = AL_{BA}$ .

2) *Attachment AT*: The attachment number  $AT_{AB}$  reflects at which location of part A, part B is connected. Accordingly, if a part is attached at the tip of a long rod, in contrast to its side, the number should reflect that change. To determine  $AT_{AB}$ , we, first of all, calculate the vector connecting centroid A to centroid B,  $\vec{r}_{AB}$ . Please see Fig. 8 for more details. For  $AT_{AB}$  we retrieve the value on the variance blob of part A,  $V_A$ , in direction of  $\vec{r}_{AB}$ . Since  $V$  is normalized, the values of single bins scales reciprocal with the number of total bins. This is why we multiply by the total number of bins. Thus, the attachment number is defined as

$$AT_{AB} = V_A(r_\theta, r_\phi) N_\theta N_\phi. \quad (6)$$

To calculate  $AT_{BA}$  we use the value on B's variance blob instead. This is why  $AT_{AB} \neq AT_{BA}$ . The noncommutative property of the attachment is best explained looking at Fig. 8. Since the blue part is highly rotational symmetric it does not matter if

we attach another part to the top or any of the sides. On the contrary, the green part is anisotropic, hence it makes a big difference for the functionality/usability in which direction/locations we attach other parts.

#### D. Object Representation

To be able to combine these signatures and describe an object  $X$  as a whole, we use a graph representation  $\mathcal{G}^X = (\mathcal{V}^X, \mathcal{E}^X)$  with the  $i$ th part signature corresponding to  $i$ th attributed node  $\mathcal{V}_i^X$  and the pose signature between part  $i$  and  $j$  forming the attributed directed edges  $\mathcal{E}_{ij}^X$  between node  $i$  and  $j$ . We define edges in the graph only between parts which are touching. To determine part proximity we first estimate the point density within both part's point clouds. We do this by generating kd-trees for each. Next we select  $M$  random points for both parts and determine the distance to their nearest neighbors within their own part. From these distances we take the maximum  $d_{max}$  and set the threshold  $\tau = 2d_{max}$ .

Again using the kd-trees, we calculate the closest distance for  $K$  random points from part A to the points in part B and vice versa. If any of these distances is smaller than  $\tau$ , we consider the parts as touching. For the experiments we use  $M = 30$  and  $K = 1000$ . Often parts are separated by a large margin, which makes this part of the algorithm robust.

### V. FUNCTION ANALYZER

In the following sections we describe how the algorithm allows comparing objects and assigning functional meanings to them. Thus, we want the function analyzer to have the following properties:

- 1) object as well as parts should be recognized;
- 2) the analyzer should be able to generalize across objects with different number of parts, to recognize the function of a cup having 6 handles, as shown in Figs. 1 and 10, should not require cups with six handles in the training set;
- 3) multiple function assignment should be possible. One object may be used for different functions with parts used for different purposes.

#### A. Training

The function analyzer's training procedure is outlined in Fig. 9. As input it uses labeled and segmented synthetic data. We used synthetic data for the training in order to create minimalistic stereotypical examples of tools, without including unnecessary details found in real-world objects.

Full objects get a primary function label (contain, cut, poke, hit, ...), and each part gets its part-functionality attached (contain:container, cut:blade, hit:head, ...). We will, in the Discussion section, address the question how much complexity arises from such a labeling procedure.

In the next step we calculate part and pose signatures (see Section IV-B and IV-C) for the training set. Inspired by the idea of Torresani *et al.* [40], we use the output of a support-vector-machine (SVM) to convert the raw part signatures to second order signatures. Using a support-vector-machine at this level allows the function analyzer to better generalize across parts with the same function. At the same time it emphasizes properties of the raw signatures, which are important for the discrimination of different functional parts. We train

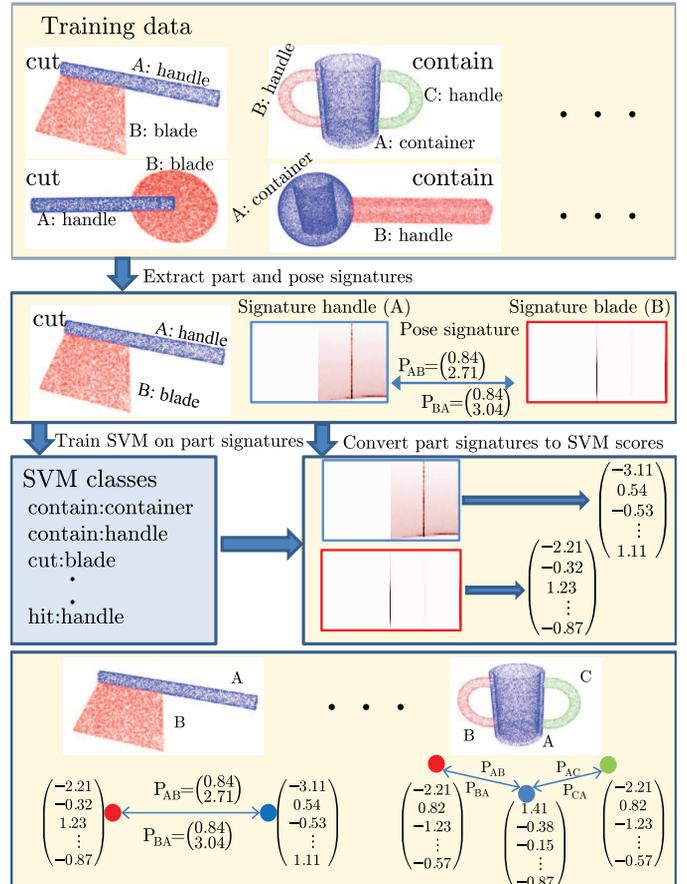


Fig. 9. Training procedure for the function analyzer.

a one-versus-rest SVM with a Chi-square kernel on the part signatures using each unique combination of function and part-name as a label (cut:blade, hit:head, contain:container, ...). For  $K$  functions with  $N$  parts each, this would result in  $K \cdot N$  classes, which accordingly results in a  $K \cdot N$  dimensional second order part signature. These signatures, the pose signatures, the graph structure, and the labels (for full objects and parts) are then stored for the later testing. Please note that we neither require all objects to have the same number of parts, nor that they should be composed of all possible parts. For example, a fillable object does not need a handle or can have multiple handles and still retains its functionality.

#### B. Testing

For a new object  $Y$  we first generate its graph representation  $\mathcal{G}^Y$  by calculating the part (nodes) and pose (edges) signatures. Edges are only drawn where parts touch (see Section IV-D). The part signatures are again transformed into SVM scores.

Since we want the function analyzer to generalize to objects with an arbitrary number of parts and graphs with different numbers of nodes and edges, we now define a similarity metric which allows all objects to be compared and at the same time leveraging on as much information as possible from the training (more specifically the part and pose signatures as well as the graph structure). As we do not know a priori which part from  $Y$  might correspond to which part from  $X$ , we need an association algorithm to allow checking all possible combinations.

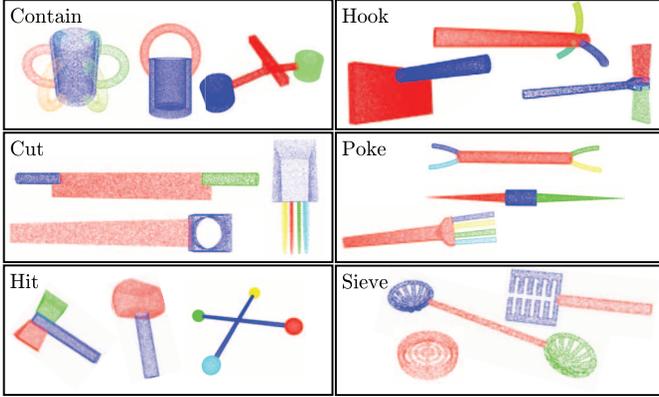


Fig. 10. Eighteen example objects from the M2 benchmark.

Let us assume object  $Y$  consists of 5 parts,  $N_Y = 5$ , and object  $X$  consists of two parts,  $N_X = 2$ . To compare both, we can assign each of the parts of  $Y$ ,  $P_Y = \{a, b, c, d, e\}$ , one of the parts of  $X$ ,  $P_X = \{A, B\}$ . A possible association (the mapping from  $P_Y$  to  $P_X$ )  $A_i$  can thus be written using a  $N_Y$ -tuple of the elements in the set  $P_X$ . The tuple  $(A, A, A, A, B)$  would for example denote the parts  $a, b, c, d$  being assigned to  $A$  and  $e$  to  $B$ . We further require the number of unique elements in the tuple

$$U(A_i) = \min(N_X, N_Y). \quad (7)$$

In the case of  $N_Y > N_X$  this corresponds to a surjective mapping, for  $N_Y = N_X$  to a bijective mapping, and in the case of  $N_Y < N_X$  to an injective mapping. For each association  $A_i$  we create an association graph  $\mathcal{G}^{A_i}$  by replacing the parts/nodes in the graph of  $Y$ ,  $\mathcal{V}^Y$ , with the associated parts/nodes of graph  $X$ . If two parts  $i, j$  in object  $X$  are touching, we add the edge  $\mathcal{E}_{ij}^X$  to the graph  $\mathcal{G}^{A_i}$ . Only if the graph has no additional or missing edges compared to the graph  $\mathcal{G}^Y$ , we call it compatible and consider it further.

This pruning is not only important for eliminating cases where required parts for a functionality are not being assigned, but it is mandatory when comparing two objects with many parts, as the number of total associations grows exponentially. For example comparing the 6 handle cup ( $N_Y = 7$ ) to any of the forks with 4 tips ( $N_X = 5$ ) shown in Fig. 10 starts with 16 807 possible associations ( $N_Y^{N_X}$ ), reduces to 2520 associations after the surjective mapping test and again reduces to 360 after checking the graph structure.

All remaining association graphs  $\mathcal{G}^{A_i}$  have now the same structure as the graph  $\mathcal{G}^Y$ , thus we next calculate association scores. For this we compare part and pose signatures separately and combine part and pose scores  $AS^{Pa}$  and  $AS^{Po}$  in a weighted sum

$$AS(A_i, Y) = \omega^{Pa} AS^{Pa}(A_i, Y) + \omega^{Po} AS^{Po}(A_i, Y) \quad (8)$$

with part and pose weights  $\omega^{Pa}$  and  $\omega^{Po}$ . For all experiments we use  $\omega^{Pa} = \frac{2}{3}$  and  $\omega^{Po} = \frac{1}{3}$ .

To calculate  $AS^{Pa}$  and  $AS^{Po}$  we L2-normalize all signatures and calculate the L2-distance between all associated part

and pose pairs from the graph  $\mathcal{G}^Y = (\mathcal{V}^Y, \mathcal{E}^Y)$  and  $\mathcal{G}^{A_i} = (\mathcal{V}^{A_i}, \mathcal{E}^{A_i})$ . The L2-distance between two signatures ranges between 0 and 2. This follows from the triangle inequality for two L2-normalized vectors  $x$ , and  $y$ :  $\|x - y\| \leq \|x\| + \|y\| = 2$ .

A normalized  $AS_i^{Pa}$  and  $AS_i^{Po}$  can thus be calculated using the equations

$$AS^{Pa}(A_i, Y) = 1 - \frac{1}{2N_V} \sum_{n=1}^{N_V} \|\mathcal{V}_n^Y - \mathcal{V}_n^{A_i}\|_2 \quad (9)$$

and

$$AS^{Po}(A_i, Y) = 1 - \frac{1}{2N_E} \sum_{n=1}^{N_E} \sum_{\substack{m=1 \\ m \neq n}}^{N_E} \|\mathcal{E}_{nm}^Y - \mathcal{E}_{nm}^{A_i}\|_2. \quad (10)$$

Here  $N_V$  and  $N_E$  denote the number of nodes and edges in the graph  $\mathcal{G}^Y$ , respectively  $\mathcal{G}^{A_i}$ .

After calculating the scores,  $AS_i$ , for all associations we define their maximum as the *final object similarity* of object  $Y$  to object  $X$

$$OS_{X,Y} = \max_{A_i} (AS(A_i, Y)). \quad (11)$$

Finally, the function compatibility is calculated using the maximum object score per function: Instead of a hard-max assignment we tried other voting schemes like  $k$ -nearest neighbors voting or averaging over all object scores per function, but this decreased performance by about 3%. Our intuition is that a winner takes it all assignment works best, because we have a huge variance in the appearance of training objects, such that averaging the response of multiple objects is not meaningful. A positive side-effect: Assuming object  $X$  leads to the function score, we can easily retrieve the labels of the parts for that functionality by using the known part-labels and winning association  $A_i$  for object  $X$ . Thus, we are not only able to assess the compatibility of an object  $Y$  to a certain function, but can additionally identify the parts, which are indispensable for this. Additionally, one can assign tools and their parts multiple possible functionalities by thresholding the function scores as we show in Fig. 12.

## VI. EXPERIMENTAL EVALUATION

### A. M1 and M2 Benchmarks

For testing the generalization capabilities of our algorithm we create two benchmarks (M1 and M2) consisting of 144 models from 56 traditional classes (like saw, hatchet, sword, dumpling spoon, pizza-cutter, cleaver, drumstick, pugil stick, rapier). Models have been generated using the shape generator from the Point Cloud Library.

For this experiment we let humans assign the most probable primitive functionality (see Fig. 1) to all objects. From all the human annotations we use the function (and the part assignments) with most votes as ground-truth. For the first benchmark (M1) we limit the objects to two parts. The second benchmark (M2) drops this restriction by allowing objects to have any number and type of parts, which increases intraclass variance.

		SH <sub>4,4</sub> <sup>Ext</sup>						
		Contain	Cut	Hit	Hook	Poke	Sieve	
Part-only	Contain	89%	0%	2%	0%	0%	9%	
	Cut	1%	81%	0%	15%	3%	0%	
	Hit	0%	0%	86%	1%	13%	0%	
	Hook	5%	14%	0%	54%	27%	1%	
	Poke	1%	7%	15%	29%	48%	0%	
	Sieve	10%	0%	0%	4%	0%	87%	
Part & Pose	Contain	90%	0%	0%	0%	0%	10%	
	Cut	3%	87%	0%	4%	6%	0%	
	Hit	0%	0%	92%	4%	4%	0%	
	Hook	0%	5%	3%	78%	12%	2%	
	Poke	0%	0%	0%	7%	93%	0%	
	Sieve	8%	0%	0%	0%	0%	92%	

		SHOT-A						
		Contain	Cut	Hit	Hook	Poke	Sieve	
Part-only	Contain	78%	2%	8%	0%	4%	7%	
	Cut	0%	91%	0%	8%	1%	0%	
	Hit	0%	0%	88%	3%	9%	0%	
	Hook	0%	9%	4%	72%	14%	0%	
	Poke	0%	1%	10%	10%	76%	2%	
	Sieve	10%	3%	0%	6%	0%	81%	
Part & Pose	Contain	83%	0%	5%	0%	2%	9%	
	Cut	0%	85%	1%	10%	3%	0%	
	Hit	0%	0%	90%	5%	4%	0%	
	Hook	0%	3%	3%	84%	10%	0%	
	Poke	0%	5%	0%	1%	94%	0%	
	Sieve	9%	2%	0%	0%	3%	86%	

Fig. 11. Confusion matrices showing the importance of Pose information to the classification pipelines. It is especially important for telling hook from poke objects, as they mostly differ in the way parts are attached and aligned.

Both benchmarks consist of 72 models. Some example objects from M2 are shown in Fig. 10.

To compare to a baseline recognition system we use SHOT [35] and ESF-features [36] extracted on the full objects together with a Chi-square SVM. Since the size of objects has a very high intraclass variance we normalize the biggest L2-distance in the objects to one. Tests are performed using the following cross-validation procedure: We train the framework with a random selection of 70% of the objects and test on the remaining 30%. We repeat this for 60 different partitionings of the data and average the accuracy. We use the ground-truth segmentation for training and testing on M1 and M2 in order to measure the performance of the function recognition in isolation.

We use the following convention to name the classification pipelines:

- 1) baseline centroid SHOT classifier trained on the full object (SHOT-C);
- 2) baseline averaged SHOT classifier trained on the full object (SHOT-A);
- 3) baseline ESF classifier trained on the full object (ESF);
- 4) classifiers trained on parts ignoring pose, superscript P (e.g., ESF<sup>P</sup>);
- 5) classifiers trained on parts together with the pose signatures. Superscript PP (e.g. ESF<sup>PP</sup>);
- 6) basic shape histogram using  $X$  distance and  $Y$  angle bins (SH <sub>$X,Y$</sub> );
- 7) extended shape histogram using  $X$  distance and  $Y$  angle bins (SH <sub>$X,Y$</sub> <sup>Ext</sup>).

## B. Models and Scans

Furthermore, we investigate how well we can generalize across domains to polygonal models and scans from the databases 3Dcadbrowser, KIT object model web database (OMWD), tf3dm, and thingiverse. We sample equi-density random points on the faces of each model. Normals are calculated using the first three vertices of each face. For all objects we generate segmentations and ground truths using the CPC-algorithm. We use the SH<sub>4,4</sub><sup>Ext,PP</sup> classification pipeline together with all training images from the set M1 to determine the functionality of the objects.

## VII. RESULTS

### A. M1 and M2 Benchmarks

Table I summarizes the mean accuracy achieved on the M1 and M2 benchmarks. SHOT-A, SHOT-C and ESF trained on the full objects are considered the baseline classification pipelines as they do not use the concept of parts and poses. The results indicate that averaging SHOT features rather than using a single centroid based feature yields better performance. Introducing part-based classifiers increases performance significantly, adding +20% to the SHOT-A classifier, which makes this classifier comparable to some systems using Part & Pose. ESF histograms are inferior to SHOT-A histograms, since normal information is not used for the former. This confirms the findings reported by Aldoma *et al.* [39]. Part and Pose pipelines finally employ the proposed system. They show the best results in both benchmarks, improving results for the M1 benchmark by up to +12% and results for the M2 benchmark by up to +15%. The fact that we combine many different objects into one functional class leads to less detailed shape histograms (4 angular and spatial bins) being better suited for recognition than more detailed shape histograms (20 angular and spatial bins).

Using extended shape histograms improves results by an additional +1 – 2%. Comparing the confusion matrices shown in Fig. 11 we notice that pose is especially important to discern poke and hook as objects, because these functions mainly differ in how parts are aligned in respect to one another.

The M2 benchmark shows more intraclass variance, because objects differ in the number of parts. This reduces performance of the baseline classifiers, on average, by about 12% as they cannot deal with this kind of variance. The proposed Part & Pose pipelines, on the contrary, are not much affected by the increased variability and can generalize from the known parts and part-to-part relations.

### B. Models and Scans

Fig. 12 shows the segmentation and classification of several models and scans. Shown is the highest scoring function as well as the second highest if the difference is less than 2%. We empirically determined this number. Allowing a confidence difference of 5% leads to wrong secondary function assignments in the case of difficult objects (like the double bladed cut object,

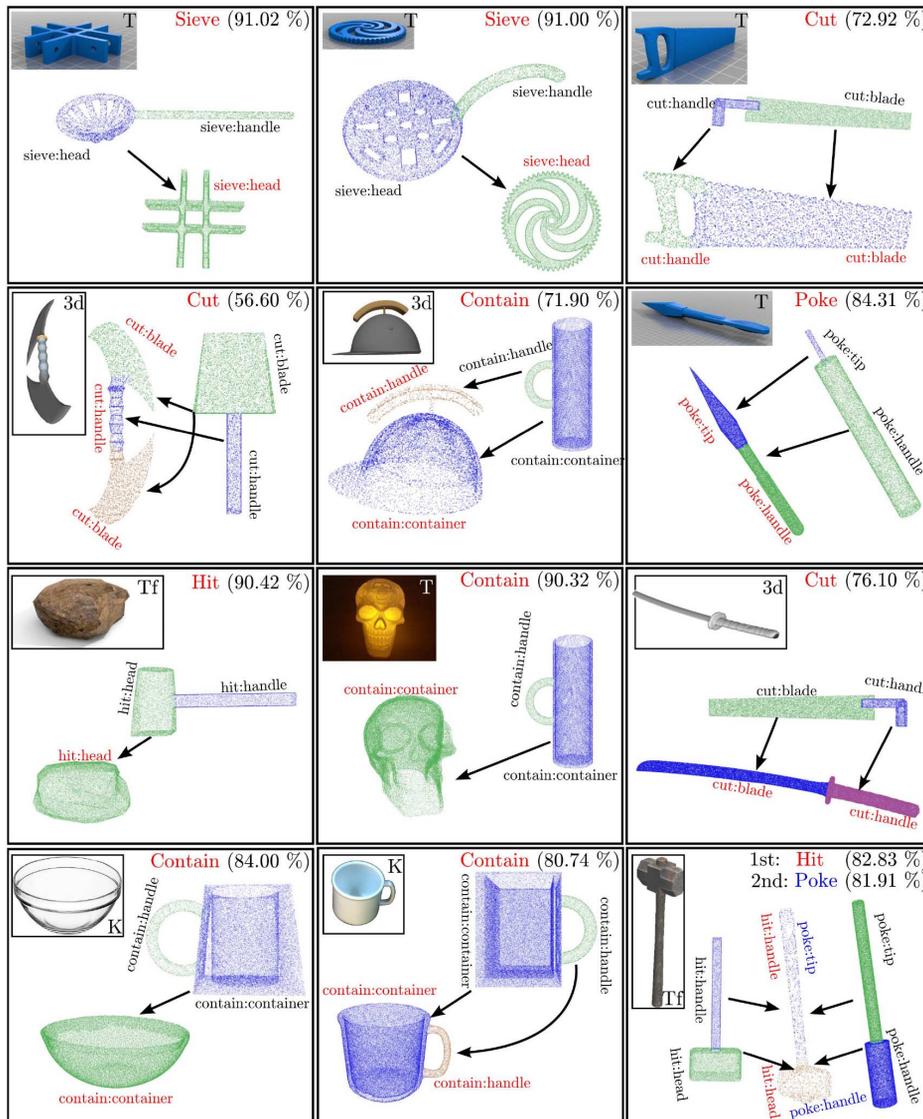


Fig. 12. Analysis of models and scans from Thingiverse (T), tf3dm (Tf), KIT OMWD (K), and 3Dcadbrowser (3d). All objects have been segmented using the CPC algorithm. The object segmentation as well as the most similar object are shown as colored point clouds. Additionally, the highest scoring function (score in brackets) as well as the part-mapping (black arrows) from the most similar object are depicted. We also include the mapping of the second highest scoring function if the score-difference is less than 2%. Training-labels are shown in black and inferred labels in red and blue.

which has a low confidence for the first function). This can be alleviated using a bigger training set.

The fact that vastly different objects can be classified using the simplistic models from our M1-dataset shows the generalization capabilities of our algorithm. This particularly crystallizes in the analysis of the hollow skull and the roman helmet models. Although these objects have been made for a different purpose they can be used as a makeshift replacement for transporting liquids. For an application of this see [41].

The hammer in the bottom right corner shows interesting results. The function classifier retrieves two high scoring functions. It can be used like an ordinary hammer for hitting. Additionally, it was labeled with the function poke, which enables an agent to use the hammer as a improvised tool to drill holes into soil for instance.

## VIII. DISCUSSION

The here suggested computer vision algorithm relies on some older ideas, like using geons for composing objects [42] and representing them as graphs [43], [44]. We had discussed above that parts have their own semantics—they are meaningful for us—an idea which goes clearly beyond a mere geometrical, geon-based representation. The bottle-neck so far had been that there were no efficient algorithms available that actually extract parts. In earlier studies we had shown that convex-concave transitions provide a very good data-driven prior for part segmentation [4], [20] and this notion is supported by a very large number of psychophysical studies, which show that humans perform part segmentation at such cutting planes [5], [45]–[50]. Therefore, we believe that the here-pursued algorithmic approach does indeed go beyond the existing older studies, which either had to rely on predefined entities [29] or on other, less meaningful features for defining an object graph.

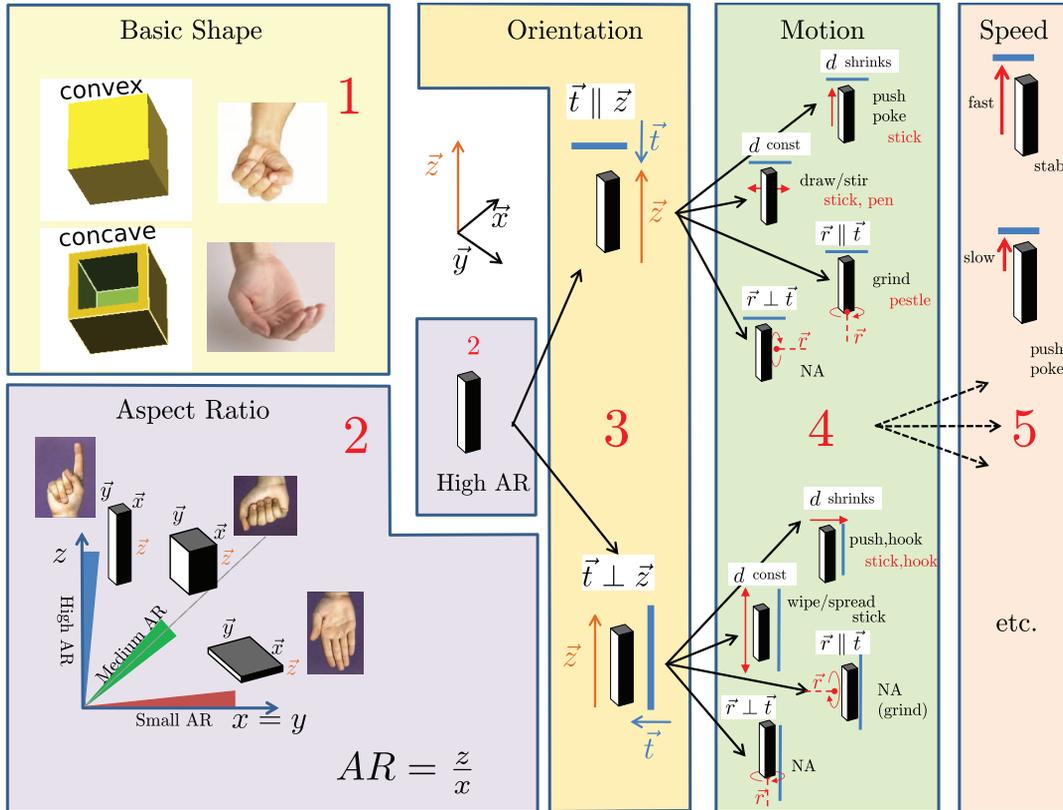


Fig. 13. Ontology of simple tools (reduced to the manipulator) derived from hand shapes and functions. Knowing the target surface (blue line) with normal  $\vec{t}$  as well as the way the manipulator acts upon it, allows us to infer the function of the manipulator in this action. Red labels in 4 denote example objects/tools to fulfill the function.  $\vec{r}$  refers to the axis of rotation and  $d$  to the distance of manipulator to target along the normal  $\vec{t}$ . For further explanation see text.

Converting the problem of traditional classification, with its recent trend to become more fine-grained, to the problem of classifying super-categories (here: functional categories) we are able to show that the semantics of a whole tool can arise from the “understanding” of the composition from its parts. The word “understanding” relates here to the supervised training of our system by some labeled data. Here it is important to emphasize that our training set is very small and that—by this—we can extract the “essence” of tools, which allows us to generalize to even unknown classes like saws, as soon as knives are known. This is nontrivial. Just consider, for example, the class “saw” and think of the very wide variety of items in this class such as those which you could buy in a D.I.Y.-shop. Thus, compared to very tunable algorithms like Deep Convolutional Neural Networks, which need huge amounts of training data to achieve some kind of generalization, our algorithm only needs very few training samples. Moreover, our method can assign multiple functionalities to one object which allows for bootstrapping makeshift replacements for tools (like a hammer used for drilling holes, or a hollow skull being used to transport liquids).

However, some limitations and potential future work exists: First, we largely ignore size of objects, both for the part-signatures as well as for the pose-signatures. At this point we are interested in the general compatibility to tool-functions without looking at the target properties (e.g., target’s size, material, and so on). When executing an action, the target properties will become important. For this, one would likely need to add a second layer of reasoning about tool-target compatibility. For example:

If the target is soft (e.g., soil), the tool can be made out of wood. If the target is hard (e.g., wood), the tool needs to be made of metal or stone. If target is 5 liters of liquid, the container must have at least this volume. Still, for assessing general function-compatibility (not target compatibility), size can be ignored in our opinion. For example, take a barrel and a cup. Both objects largely differ in their size, still they are both used to contain a liquid. Therefore, ignoring size largely increases generalization performance of the algorithm.

Second, we compare an object to all other instances and use a classifier only for generalizing part signatures. While this has advantages, like allowing assignment of single parts to multiple other parts, or by providing more insight into the function of the algorithm (by for example showing the most similar training objects, Fig. 12), this shifts most of the algorithm’s complexity to the testing stage. We are currently investigating possibilities to use a classifier also at the function assignment stage, which would provide a remarkable speed-up at testing time.

Third, if we deal with very complicated objects, like a saw where the blade is hidden, our algorithm fails as long as such a saw is not in the training set. However, even we, as humans, would not be able to assess such tool’s function without having seen a similar saw before (or reading the label).

## IX. BOOTSTRAPPING TOOL USE - AN EXTENSION

In the current study we have shown our results on classifying objects (tools) by considering their parts and part relations. We have motivated our approach by the use of the human

hand, which we consider as a possible structure for bootstrapping the understanding of tool function. That first part of the study was much motivated by the speculation that cognitive ability to use ones hand in different ways allows learning to transfer this knowledge to unknown objects, which are then recognized as tools. To support the conjecture of a possible cognitive hand-to-tool transfer we are, here, analyzing primary tools by also incorporating tool-dynamics. For this we design a tool-ontology and argue that its complexity remains quite limited.

Recognition of tool function is in our system—like in many others—fundamentally a supervised process. We used “labeled part data” for training where we claimed that there are only very few basic tool functions existing, which are (all) related to the shapes and the function of our hands. A handle is a common entity among all objects we analyzed so far. In contrast, the manipulator, or tool-end (the actual part interacting with a target) is more discriminative and determines most of the functionality. Interestingly, the handle shape together with the way a handle is attached and aligned to the manipulator—determined by the pose-signature in the previous sections—just determines the direction and motion a tool can be applied best. This concept is supported by the fact that first prehistoric tools only consisted of the manipulator without an attached handle. A handle was added later to make the tool more usable by allowing for example a longer lever or an improved grasp.

But, if one knows how the manipulator is being used (e.g., the trajectory and motion relative to the target), the handle is actually not important anymore for the recognition of the tool function. Fig. 13 shows how one could include this information in an ontology of primordial tools using hand-shape and function as a reference.

We distinguish five levels (red numbers). The first three levels are strictly geometrical, where level 1 and 2 directly refer to the hand shape and level 3 to the arrangement of the hand (or manipulator) relative to the target object. Levels 4 and 5 take the movement patterns into account, too. Hence, here we move from a pure object-guided (hand shape guided) ontology to the final one which uses manipulator shape, arrangement and the actual action pattern for tool classification. All this remains very reductionistic and we think *less is not possible*. We show in the end that such an ontology contains only 32 entries for different possible simple tools and how concept of wheels naturally emerges.

Level 1 asks about the basic hand shape: Is it convex or concave? In the figure we just show the convex branch to explain the next levels.

Level 2 addresses the aspect ratio (“AR”). For simplicity we set  $x = y$  (the coordinate system is given in the center of the figure) and plot the possible aspect ratios. Hence this plot exceeds the hand size to show by ways of the three colored areas (blue, green, red) roughly which aspect ratios are existing (a very limited range) when considering regular tools ranging from: (high AR) borers to sabers, from (medium AR) small to large hammers and from (small AR) paddles, spades to cleavers.

We use now an object with high aspect ratio (*High AR* in Fig. 13–2) to explain the next three levels. The same arguments hold for the other aspect ratios, too.

Level 3 addresses the relative orientation between hand shape (or tool) and target object (target surface  $\vec{t}$ ) indicated by blue.

Using our elongated tool with the tip pointing against the target surface, hence in a parallel way to the normal ( $\vec{t} \parallel \vec{z}$ , Level 3, top) could mean it is a poker. Using it in a perpendicular arrangement ( $\vec{t} \perp \vec{z}$  Level 3, bottom) might make it a tool for spreading something out.

In the 4th level we consider the relative motion between tool and target surface during tool operation and whether we have a rotational component in the movement. If the distance  $d$  gets smaller and we have a parallel arrangement ( $\vec{t} \parallel \vec{z}$ ) then this might mean that this is a pusher, poker, or stabber (Level 4, very top). If, in the same arrangement,  $d = const$ , then this is possibly a drawing or stirring tool (Level 4, one-down from the very top). Already at this level several configurations do not make much sense anymore or are only very rarely found (indicated with “NA = not applicable”). Hence, not all slots in this ontology are filled.

Level 5 finally asks about the dynamics of the movement, mainly: Is it fast or slow? The two examples shown allow distinguishing push/poke from stab.

Note that at level 4 we observe a few rotation examples. Most indeed work using your hands but it would be much more efficient if one uses a tool with an axis. Thus, this ontology suggests introducing wheel-like structures. It certainly goes too far to think that there had been a mental transfer from hand function all the way to the design of rotational tools. However, the desire to arrive at a better functionality in these rotation-cases might well have stimulated inspiration and ingenuity leading to the wheeled tools we have nowadays.

Thus, it seems there are some mental transfer processes that are easy and could indeed have taken place this way during the evolution of hominids: fist to hammer, finger to borer (stick), etc. Others are clearly unrealistic (grinder to wheeled grinder). But this is not the main point. Central to our argumentation is that the here presented ontology leads to a very small system of primary tools, which can easily be stored and remembered by real or artificial agents. Thus, manual tagging of training examples for tools based on their parts and part relations—as performed in this study—requires not much effort and only a total of 32 entries.

Fig. 14 shows that we have found indeed only 32 different actions with their simple tools when using this ontology. Note that the same tool (e.g., “stick” can appear in different actions (e.g., “push” or “draw”). Also, there are certainly many variants of tools and tool names existing of which we tried to only include common examples.

Tools are ordered by their basic shape (convex versus concave, top) as well as their aspect ratio as defined in Fig. 13. The tools beneath the triple separating lines in every section are those that require circular or turning movement patterns. Hence, those are generally wheeled tools. Note that there are far fewer tools existing in the concave branch, most of which are containers.

Let us consider one comparison and refer this back to the definition of the ontology in Fig. 13. The tuple: (Action shovel, Tool shovel) in the concave section third from above means that here:

- a usually *fast* (level 5);
- *forward movement* is performed (level 4,  $d$  shrinks);
- *against* the target surface (level 3,  $\vec{t} \perp \vec{z}$ );

	Convex (Level 1)			Concave (Level 1)		
	Action	Level 3	Tool	Action	Level 3	Tool
		Level 4			Level 4	
		Level 5			Level 5	
Small AR (Level 2)	T	Paddle	Paddle	Sieve, lift	Sieve	
		Hit	Rug beater	Sieve, shake	Sieve	
		Spread	Butter knife, spatula	Shovel	Shovel	
		Chop	Cleaver, axe, sword	Rake	Rake	
		Cut	Knife, sword			
	R	Mix	Mixer	Empty	Shovel	
		Paddle/mix parallel* circular	Blade(s) of an agitator			
		Paddle/mix perp.* circular	Blade(s) of a water mill			
		Grind parallel* circular	Grinding/millstone used flat			
		Cut/grind perp.* circular	Circular saw, angle grinder			
Medium AR (Level 2)	T	Push	Hammer	Fill	Cup, ladle	
		Hit	Hammer			
		Grind	Grinder, pestle			
	R	Grind parallel* circular	Grinding/millstone used flat (fat)	Pour	Cup	
		Grind perp.* circular	Grinding stone sed upright			
High AR (Level 2)	T	Push poke	Stick	Fill	Test tube	
		Stab	Rapier, dagger			
		Draw, stir	Stick, pen			
		Push	Stick			
		Whip	Cane			
		Wipe, spread, hook	Stick, hook			
		Wipe, spread	Stick			
	R	Bore, drill	Drill	Pour	Test tube	

\*perpendicular versus parallel refers to the orientation of the disk like tool relative to the target surface

Fig. 14. All entries for the tool ontology showing actions as well as the related tools. We grouped entries according to their *Basic Shape* (Level 1), *Aspect Ratio* (Level 2), and to the property if motion is translational *T* or rotational *R*. A total of 32 actions and their tools are found.

- with a *shovel-shaped* (Level 2, small AR);
- *concave* tool (level 1):  
where the tool is pushed into something, like a pile of sand, to shovel it up. Now you could continue this movement staying in the same ontological class throwing the content off in a similar forward motion (e.g. continuing the shoveling motion). Or by comparison you could use the tuple (Action “empty,”<sup>4</sup>Tool shovel) where this is the movement by which the filled shovel is turned to empty it and for which a different branch in the ontology exists.

## X. CONCLUSION

This paper was meant to give some speculative food for thought about cognitive development and tool ontologies, but at the same time we tried to provide a rather more solid algorithmic basis for the possible underlying processes. The here developed framework for providing object graphs based on their parts and part constellations generalizes to all objects, which can be segmented into their parts. Several algorithms exist by now that achieve the latter to quite a high degree of accuracy, which supports the viability of this approach. In general, we advocate the idea that object recognition might be much facilitated when considering part combinations and we have used simple tools to show how this might work. Just by the fact that a part can also be considered an object and since one can combine same parts to many different objects there are much less “parts” than “objects” in the world. One could indeed hope that this approach might be more successful and possibly

<sup>4</sup>Means: to empty something.

“brain-like” than brute-force training of deep-learning classifiers where all this variance needs to be put into the training set. In contrast to our proposed method those will continue to fail as soon as human invention designs (artistic) objects, which humans—but not such machines—can easily classify.

## REFERENCES

- [1] C. V. Ward, M. W. Tocheri, J. M. Plavcan, F. H. Brown, and F. K. Manthi, “Early pleistocene third metacarpal from kenya and the evolution of modern human-like hand morphology,” in *Proc. Nat. Acad. Sci.*, 2014, vol. 111, no. 1, pp. 121–124.
- [2] T. Torigoe, “Comparison of object manipulation among 74 species of non-human primates,” *Primates*, vol. 26, no. 2, pp. 182–194, 1985.
- [3] F. Guerin, N. Kruger, and D. Kraft, “A survey of the ontogeny of tool use: From sensorimotor experience to planning,” *IEEE Trans. Autom. Mental Develop.*, vol. 5, no. 1, pp. 18–45, Mar. 2013.
- [4] M. Schoeler, J. Papon, and F. Wörgötter, “Constrained planar cuts - object partitioning for point clouds,” presented at the IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2015.
- [5] I. Biederman, “Recognition-by-components: A theory of human image understanding,” *Psychol. Rev.*, vol. 94, no. 2, pp. 115–147, 1987.
- [6] I. Biederman, “Human object recognition: Appearance vs. shape,” in *Shape Perception in Human and Computer Vision*, ser. Advances in Computer Vision and Pattern Recognition, S. J. Dickinson and Z. Pizlo, Eds. Berlin, Germany: Springer-Verlag, Jan. 2013, pp. 387–397.
- [7] I. Biederman, “The neural coding of parts and relations in object recognition,” presented at the ECCV Workshop Parts Attrib. (ECCVW), 2012.
- [8] W. A. Richards and D. D. Hoffman, “Codon constraints on closed 2d shapes,” *Comput. Vis., Graphics, and Image Process.*, pp. 265–281, 1985.
- [9] D. D. Hoffman and W. A. Richards, “Parts of recognition,” *Cognition*, vol. 18, pp. 65–96, 1984.
- [10] M. Behrmann, M. A. Peterson, M. Moscovitch, and S. Suzuki, “Independent representation of parts and the relations between them: Evidence from integrative agnosia,” *J. Exp. Psychol.: Human Perception Perform.*, vol. 32, no. 5, pp. 1169–1184, 2006.

- [11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.* 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012, pp. 1097–1105, Curran Associates, Inc..
- [12] N. Ahuja and S. Todorovic, "Connected segmentation tree: a joint representation of region layout and hierarchy," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2008, pp. 1–8.
- [13] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, vol. 33, no. 5, pp. 898–916, May 2011.
- [14] P. Viola and M. Jones, "Robust real-time face detection," *Int. J. Comput. Vis. (IJCV)*, vol. 57, no. 2, pp. 137–154, 2004.
- [15] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.
- [16] P. Arbelaez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik, "Semantic segmentation using regions and parts," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2012, pp. 3378–3385.
- [17] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGB-D images," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2012, pp. 746–760.
- [18] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2013, pp. 564–571.
- [19] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski, "CAD-model recognition and 6DOF pose estimation using 3D cues," in *Proc. 2011 IEEE Int. Conf. Comput. Vis. Workshops (ICCV Workshops)*, 2011, pp. 585–592.
- [20] S. Stein, M. Schoeler, J. Papon, and F. Wörgötter, "Object partitioning using local convexity," presented at the IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2014.
- [21] B. Yao and L. Fei-Fei, "Modeling mutual context of object and human pose in human-object interaction activities," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2010, pp. 17–24.
- [22] H. Pirsiavash and D. Ramanan, "Detecting activities of daily living in first-person camera views," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2012, pp. 2847–2854.
- [23] C. Desai, D. Ramanan, and C. Fowlkes, "Discriminative models for static human-object interactions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, 2010, pp. 9–16.
- [24] A. Farhadi and M. A. Sadeghi, "Phrasal recognition," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, vol. 35, no. 12, pp. 2854–2865, Dec. 2013.
- [25] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.
- [26] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman, "Discovering object categories in image collections," presented at the IEEE Int. Conf. Comput. Vis. (ICCV), 2005.
- [27] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2011, pp. 1385–1392.
- [28] D. Huber, A. Kapuria, R. Donamukkala, and M. Hebert, "Parts-based 3D object classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2004, p. II82.
- [29] M. Tenorth, S. Profanter, F. Balint-Benczedi, and M. Beetz, "Decomposing cad models of objects of daily use and reasoning about their functional parts," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2013, pp. 5943–5949.
- [30] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *Int. J. Comput. Vis. (IJCV)*, vol. 61, no. 1, pp. 55–79, 2005.
- [31] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Trans. Comput.*, vol. 22, no. 1, pp. 67–92, Jan. 1973.
- [32] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang, "Contextual part analogies in 3d objects," *Int. J. Comput. Vis. (IJCV)*, vol. 89, no. 2, pp. 309–326, 2010.
- [33] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2009, pp. 951–958.
- [34] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter, "Voxel cloud connectivity segmentation - supervoxels for point clouds," presented at the IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2013.
- [35] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2010, pp. 356–369.
- [36] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, 2011, pp. 2987–2992.
- [37] W. Mustafa, N. Pugeault, and N. Krüger, "Multi-view object recognition using view-point invariant shape relations and appearance information," presented at the IEEE Int. Conf. Robot. Autom. (ICRA), 2013.
- [38] L. A. Alexandre, "3D descriptors for object and category recognition: A comparative evaluation," presented at the Workshop Color-Depth Camera Fusion Robot. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2012.
- [39] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-dimensional object recognition and 6 DOF pose estimation," *IEEE Robot. Autom. Mag.*, vol. 19, no. 3, pp. 80–91, 2012.
- [40] L. Torresani, M. Szummer, and A. Fitzgibbon, "Efficient object category recognition using classemes," in *Proc. 11th Eur. Conf. Comput. Vis.: Part I*, 2010, pp. 776–789, ser. European Conference on Computer Vision (ECCV).
- [41] A. Uderzo and R. Gosciny, *Astérix et les normands*, p. 17, 1967.
- [42] I. Biederman, "Recognition-by-components: A theory of human image understanding," *Psychol. Rev.*, vol. 94, no. 2, p. 115, 1987.
- [43] S. Biasotti, S. Marini, M. Spagnuolo, and B. Falcidieno, "Sub-part correspondence by structural descriptors of 3D shapes," *Comput.-Aided Design*, vol. 38, no. 9, pp. 1002–1019, 2006.
- [44] H. Laga, M. Mortara, and M. Spagnuolo, "Geometry and context for semantic correspondences and functionality recognition in man-made 3D shapes," *ACM Trans. Graph.*, vol. 32, no. 5, pp. 150:1–150:16, 2013.
- [45] E. Rubin, *Visuell wahrgenommene Figuren. Copenhagen: Gyldenalske Boghandel 1915 Reprinted as: Figure and Ground*, D. C. Beardslee and M. Wertheimer, Eds. Princeton, NJ, USA: Van Nostrand, 1958.
- [46] J. J. Koenderink and A. J. van Doorn, "The shape of smooth objects and the way contours end," *Perception*, vol. 11, no. 2, pp. 129–137, 1982.
- [47] D. D. Hoffman and W. A. Richards, "Parts of recognition," *Cognition*, vol. 18, no. 13, pp. 65–96, 1984.
- [48] M. L. Braunstein, D. D. Hoffman, and A. Saidpour, "Parts of visual objects: An experimental test of the minima rule," *Perception*, vol. 18, pp. 817–826, 1989.
- [49] A. D. Cate and M. Behrmann, "Perceiving parts and shapes from concave surfaces," *Attention, Perception, Psychophysics*, vol. 72, no. 1, pp. 153–167, 2010.
- [50] M. Bertamini and J. Wagemans, "Processing convexity and concavity along a 2-D contour: Figure-ground, structural shape, and attention," *Psychonomic Bulletin Rev.*, vol. 20, no. 2, pp. 191–207, 2013.



**Markus Schoeler** studied physics at the University of Würzburg, Würzburg, Germany. He received the M.Sc. degree in physics for his research from the IOF-Fraunhofer Institute, University of Jena, Jena, Germany, in 2010. Currently, he is working toward the Ph.D. degree in computer vision at the University of Göttingen, Göttingen, Germany.

Among his research interests are bottom-up object and scene partitioning, robot vision, object classification, and object cognition.



**Florentin Wörgötter** has studied biology and mathematics at the University of Düsseldorf, Düsseldorf, Germany. He received the Ph.D. degree for work on the visual cortex from the University of Essen, Essen, Germany, in 1988.

From 1988 to 1990, he was engaged in computational studies with the California Institute of Technology, Pasadena, CA, USA. Between 1990 and 2000, he was a Researcher at the University of Bochum, Bochum, Germany, where he was investigating the experimental and computational neuroscience of the visual system. From 2000 to 2005, he was a Professor of Computational Neuroscience with the Psychology Department, University of Stirling, U.K., where his interests strongly turned towards Learning in Neurons. Since July 2005, he has been the Head of the Computational Neuroscience Department at the Bernstein Center for Computational Neuroscience, Inst. Physics 3, University of Göttingen, Germany. His current research interests include information processing in closed-loop perception-action systems, sensory processing, motor control, and learning/plasticity, which are tested in different robotic implementations.