Decomposing CAD Models of Objects of Daily Use and Reasoning about their Functional Parts

Moritz Tenorth[*] tenorth@cs.uni-bremen.de
Stefan Profanter[†] profante@cs.tum.edu
Ferenc Balint-Benczedi[*] balintbe@cs.uni-bremen.de
Michael Beetz[*] beetz@cs.uni-bremen.de

***Abstract*— Today's robots are still lacking comprehensive knowledge bases about objects and their properties. Yet, a lot of knowledge is required when performing manipulation tasks to identify abstract concepts like a "handle" or the "blade of a spatula" and to ground them into concrete coordinate frames that can be used to parametrize the robot's actions. In this paper, we present a system that enables robots to use CAD models of objects as a knowledge source and to perform logical inference about object components that have automatically been identified in these models. The system includes several algorithms for mesh segmentation and geometric primitive fitting which are integrated into the robot's knowledge base as procedural attachments to the semantic representation. Bottom-up segmentation methods are complemented by top-down, knowledge-based analysis of the identified components. The evaluation on a diverse set of object models, downloaded from the Internet, shows that the algorithms are able to reliably detect several kinds of object parts.**

I. Introduction

Robots that are to perform household tasks like setting a table or preparing simple meals [1] or that are to become co-workers in a factory, as laid out in different research roadmaps [2], [3], [4], will need to competently interact with different tools and other objects of daily use. Most of these items have been created for a specific purpose and are therefore composed of a set of functional parts. Often, objects are even *defined* by these functional parts: Wikipedia for example defines a spoon[1] as "a utensil consisting of a small shallow bowl, oval or round, at the end of a handle" and a bottle[2] as "a rigid container with a neck that is narrower than the body and a mouth". To apply such definitions when performing a task, a robot needs to be able to find the components they refer to and compute their geometric properties, like their positions and shapes, which is needed for manipulating them.

Competently handling everyday items requires robots to have large-scale object knowledge bases whose manual construction can be a tedious and time-consuming task. We therefore investigate methods to automate their construction from existing sources on the Internet: On the one hand, there are textual descriptions of which parts an object consists of, in Wikipedia as well as in instructions for everyday activities on pages like *wikihow.com*. This information can be imported into the robot's knowledge base [5], but is not sufficient for interacting with the real physical objects. On the other hand, there are public databases such as the Trimble/Google 3D warehouse[3] that already contain hundreds of thousands of CAD models and that are expected to grow substantially over the following years. These databases provide detailed models of the geometry of kitchen utensils, tools and appliances, but almost no semantic annotations. Integrating both information sources will enable robots to ground the textual descriptions in geometric information to competently handle the objects. To identity functional components in the monolithic CAD models, they need to be decomposed, semantically interpreted and set in relation to the textual descriptions. In addition, robots have to perceptually ground the models in their sensor data to be able to interact with them in the real world.

This paper investigates how such hybrid symbolic-geometric knowledge bases can automatically be constructed by decomposing CAD models of objects into their functional parts (Figure 1). The system consists of (a) a set of Prolog

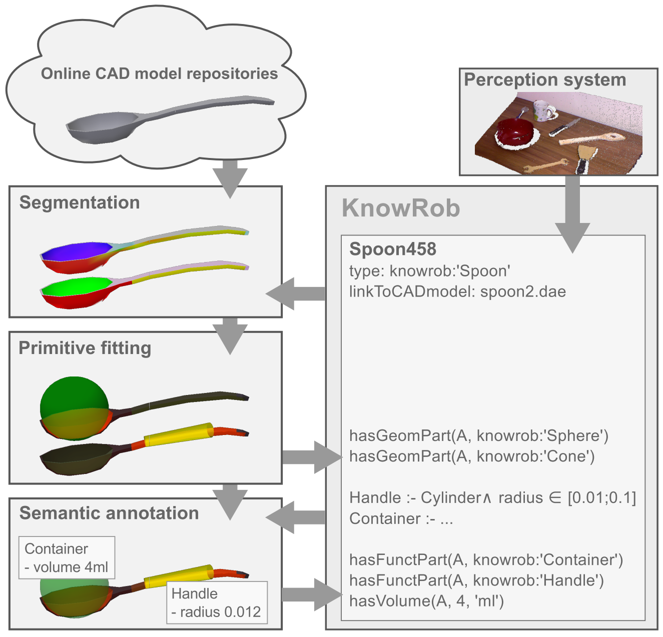
Online CAD model repositories
Perception system
Segmentation
KnowRob
Spoon458
type: knowrob:'Spoon'
linkToCADmodel: spoon2.dae
Primitive fitting
hasGeomPart(A, knowrob:'Sphere')
hasGeomPart(A, knowrob:'Cone')
Handle :- Cylinder∧ radius ∈ [0.01;0.1]
Container :- ...
Semantic annotation
Container
- volume 4ml
Handle
- radius 0.012
hasFunctPart(A, knowrob:'Container')
hasFunctPart(A, knowrob:'Handle')
hasVolume(A, 4, 'ml')

Fig. 1. Overview of the system for the semantic interpretation of geometric object models. The modules for mesh segmentation, geometric primitive fitting and semantic annotation are integrated with the robot's knowledge base such that the identified components are instantiated in the knowledge base to be available for logical inference and that symbolic knowledge can be used for advanced interpretation.

[*] Moritz Tenorth, Balint-Benczedi and Michael Beetz are with the Institute for Artificial Intelligence and the TZI (Center for Computing Technologies), University of Bremen, Germany. [†] Stefan Profanter is with Technische Universität München, Germany.

[1]http://en.wikipedia.org/wiki/Spoon

[2]http://en.wikipedia.org/wiki/Bottle

[3]http://sketchup.google.com/3dwarehouse

rules that define objects and functional object parts in terms of geometric primitives, (b) methods for segmenting CAD models into these geometric primitives, and (c) components for matching the CAD models to sensor data. Our system automatically identifies geometric primitives like planes, cylinders and spheres. Logical rules that define functional components like handles or containers can be applied to these primitives to identify higher-level concepts. The resulting components are instantiated in the robot's knowledge base and can be used to answer queries that combine semantic and geometric aspects like

- Where shall I grasp this object, where is its handle?
- Into which of these objects can I pour one liter of liquids?
- I need to pour batter onto a pancake maker, where shall I pour it?
- Which part do I need to grasp to open a bottle?

Our main contributions are (1) the approach of modeling objects by their functional parts, combining geometric and semantic information, (2) methods for automatically creating these functional object models from CAD models downloaded from the Internet, and (3) the integrated system from perception to semantic interpretation that allows robots to reason about the parts and properties of the objects in front of them.

II. Related Work

The work we present is related to affordances [6] that also deal with functional object parts and the actions that can be performed with them. There has been much work on recognizing affordances in sensor data, often focusing on graspability (e.g. [7], [8], [9], [10]). The common approach is to segment two- or three-dimensional sensor data in a bottom-up fashion and to compute properties like concavity or affordances like graspability directly from the sensor data. This has the advantage that no models are required, so that a robot can interact with unknown objects based only on their geometric properties. These approaches are however limited in that they have to deal with noisy data, can only interpret the visible parts of an object, and do not make use of higher-level knowledge. Information about the objects can help to distinguish perceptually similar objects with different functions (a roll of tape and a bowl may look similar, but only one is a container).

Model-based approaches can apply prior knowledge to improve the interpretation. Aldoma [11] presents a system that detects affordances like containment or stack-ability using CAD models of object categories. Kresse [12] describes methods for recognizing and classifying tools like spoons, spatulas or skimmers, and to use the result for parameterizing the robot's controller. These approaches are closely related to ours, though their focus is on perception and on the classification of whole-object affordances, while we concentrate on extracting knowledge from the models and on performing reasoning about object components.

While we apply state-of-the-art techniques to segment the object models into parts, we not claim to make contributions to this research area. There is a large body of work on mesh segmentation[13], [14] and geometric primitive fitting [15] using techniques like hierarchical clustering [16], [17] or machine learning [18], [19]. These methods are rarely used for robotic applications ([20] is one example) and provide only geometric, but no semantic analysis.

III. Overview

In this section, we will explain how objects are represented in our hybrid geometric-semantic knowledge base and how this internal representation can be generated.

A. Object representation

The object representation in the knowledge base needs to bridge the gap between the abstract symbolic description of object parts and the detailed geometry in the CAD models. It is based on the assumption that many functional object parts can (very roughly) be described by a combination of only a few geometric primitives, namely planes, cones, cylinders and spheres. The spoons in Figure 2, for example, consist mainly of the handle (approximately cylinder-shaped) and the bowl (approximately a sphere segment). There is evidence from psychological research that humans model and recognize objects by separating them into primitive geometrical shapes called *geons* [21].

Note that we do not require the shapes to match exactly: The spoons' handles are not straight and flat on the top, their bowls are somewhat elongated, neither surface is smooth. Yet, the primitives can be matched, and the abstraction allows to relate abstract descriptions ("a small shallow bowl [...] at the end of a handle") to the geometric information. As the examples in Figure 2 show, this also allows some generalization across concrete objects. Section VI will explain the representation of objects and their components in the knowledge base in more detail.

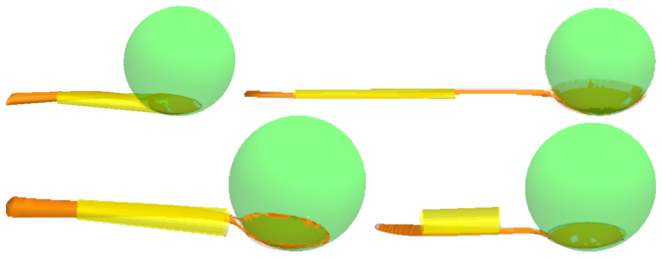
Fig. 2. Representation of objects as a set of functional components. Though the spoons are of very different shapes, they are composed of the same functional components.

B. Model acquisition

The analysis of a new object model starts with a segmentation based on the surface curvature and a fitting of geometric primitives into these segments. After this step, the basic object components are known and can be represented in the knowledge base. The symbols generated for these components remain linked to the respective segmented surface mesh, which allows to compute additional properties like their area or their volume later on. Once there is a symbolic representation of the composition of an object, logical definitions of functional parts can be applied. These Prolog rules identify the respective parts in the geometric primitives and to instantiate them in the knowledge base.

The system is realized as an extension of the KNOWROB knowledge base [22] that is available as open-source software in the ROS distribution[4]. KNOWROB is a knowledge processing system that is specifically designed to be used on and by robots, and that provides them with knowledge needed for performing everyday manipulation tasks. It is based on SWI Prolog [23] and represents knowledge in terms of OWL statements [24]. The KNOWROB ontology provides descriptions of hundreds of objects, objects parts, and their properties. Each geometric analysis method (e.g. a handle detector) is attached to the respective concept in the knowledge base it computes (in this case, the class *Handle*).

IV. Object Model Segmentation

The first interpretation step is a segmentation of the object model into parts. We apply a curvature-based segmentation technique and fit geometric primitives like planar surfaces, spheres, cones and cylinders to the segments. We apply the curvature estimation method proposed by Rusinkiewicz [25] that can be used for irregular triangle meshes (which are commonly found in models downloaded from the Internet). It computes the vertex curvatures based on a weighted average of the normal vectors of the adjacent triangles.

As a compact representation of the curvature at each vertex, we compute a color value from the curvature tensor. The hue (0 to 360 degrees) and saturation (-1 to 1) are calculated based on the mean curvature $H = (k_1 + k_2)/2$ and the Gaussian curvature $K = k_1 \cdot k_2$ as follows:

$$hue = \frac{4}{3} \,|\, atan(H^2 - K, H^2 \cdot sgn(H)) \,| \tag{1}$$

$$sat = \frac{2}{\pi}\, atan((2H^2 - K) \cdot scale) \tag{2}$$

The $scale$ parameter indicates how curved a surface may still be to be still considered planar. A smaller value leads to greater tolerance for lightly curved surfaces to be regarded as planes. The color value is used to segment the object surface. First, each vertex is annotated with the kind of primitive shape (planar surface, convex/concave sphere, convex/concave cone) it most likely belongs to. The annotation of a triangle is determined by majority voting among the annotations of its vertices. In the rare case that all three annotations are different, we prioritize planes over spheres and cones, and convex over concave annotations which we found to occur more frequently in our data set. Neighboring triangles with the same annotations are then combined to larger surfaces using a region-growing approach. The resulting segmentation is smoothed by merging very small annotations into larger neighboring surfaces (if their area is below 5% of the larger surface) and by averaging annotations of adjacent triangles.

[4]http://www.knowrob.org

V. Geometric Primitive Fitting

The result of the previous step is a segmentation into submeshes of approximately the same curvature. By matching geometric primitives, we obtain a compact and parameterizable representation as intermediate step towards the semantic interpretation.

A. Plane Fitting

A plane is defined as a two-dimensional rectangle in 3D space described by a normal vector and two values for the length of each side. The formula for a 3D plane through $(0, 0, 0)$ is $a \cdot x + b \cdot y + c \cdot z + d = 0$. We minimize the distance from each point to the plane by determining a, b, c and d such that the following equation becomes minimal

$$f(a, b, c, d) = \sum \frac{(a \cdot x_i + b \cdot y_i + c \cdot z_i + d)^2}{a^2 + b^2 + c^2} \tag{3}$$

by setting the partial derivatives with respect to d to zero. The plane normal vector is then the largest eigenvector of the resulting matrix. To determine the extent of the plane, we fit a rectangle to the vertices projected into a plane perpendicular to the normal vector. The enclosing rectangle for these points is then calculated by first computing the convex hull using the Graham Scan algorithm [26] and then calculating the minimal-area enclosing rectangle whose sides are aligned with the edges of the convex hull. Figure 6 shows examples of planes found in the models.

B. Sphere Fitting

A sphere is described by $(x - a)^2 + (y - b)^2 + (z - c)^2 = r^2$, where (a, b, c) is the sphere center and r is its radius. Our algorithm is based on [27]; a precondition is that not all the points are coplanar, which is given since otherwise the vertices would not have been classified as part of a sphere in the segmentation step. The algorithm fits a sphere to the vertices of a mesh segment by minimizing the following energy function:

$$E(a, b, c, r) = \sum_{i=1}^{m} (L_i - r)^2 \tag{4}$$

with m as the number of vertices and $L_i = \sqrt{(x_i - a)^2 + (y_i - b)^2 + (z_i - c)^2}$. It can be solved using fixed-point iteration after setting the partial derivatives with respect to a, b, c, and r to zero. Examples of fitted spheres are shown in Figure 3.

C. Cone and Cylinder Fitting

Cylinders are a special case of cones whose bottom and top radii are equal, i.e. the following algorithm can be used for both kinds of primitives. Cones are described by their generating line, their height, and the bottom and top radii. Given a cone-shaped mesh segment, we compute the generating line by iterating over all vertices, randomly selecting two other vertices, and computing the intersection points of the planes described by the position and normal vectors of the vertices. The apex (tip of the cone) is the intersection point; if there is none, the annotation is a cylinder, not a cone.

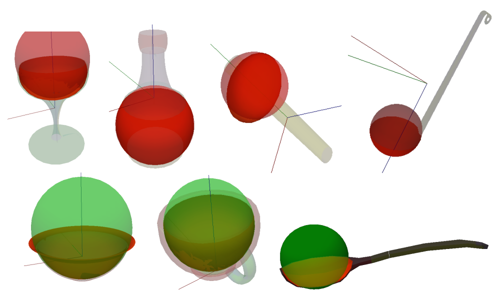
Fig. 3. Examples of convex (red) and concave (green) spheres fit to different objects. In case of thin-walled objects like the spoons, bowl, glass or cup, there are usually also the complementary inner/outer spheres.

The cone direction is computed from the mean positions of the vertices and the apex position, the opening angle by the angle between the generating line and the vectors between the apex and the vertices. The values for each set of vertices are averaged and then used to compute the top and bottom radii as the averaged distance of vertices above and below the center to the generating line. The algorithm iterates between computing the radius and correcting the direction of the generating line by comparing the direction of the radius with the vertex normals until convergence.

VI. Knowledge-based Object Representation

The primitive shapes determined in the previous step are now instantiated in the symbolic knowledge base. In the KnowRob system, objects are represented as instances of abstract object classes like *Cup* or *Spoon*. Using the *properPhysicalParts* relation, these instances are linked to the object's parts, which are themselves instances of concepts like *Cylinder* or *Sphere*. The generation of these component instances is performed automatically whenever a query involves components of an object that has not yet been analyzed. All components are cached for future queries and remain linked to the corresponding surface mesh segment that can be used to compute additional information like the dimensions, volume, diameter of the component. Table I lists which properties can be computed for the different kinds of annotations. As for the components themselves, the computation of these properties is automatically triggered whenever they are needed to answer a query. Example queries are shown in Section IX.

VII. Semantic Annotation of Object Parts

The representation of object parts in the knowledge base forms the basis to apply abstract knowledge to identify semantically meaningful higher-level concepts. In this paper, we exemplarily describe the identification of containers, handles, and supporting planes by applying logical rules to the representation of object parts.

A. Finding Containers

The kind of containers we consider are concave objects that are open on one and closed on another side. In particular, we define containers as "concave cones that have a planar surface near one end, which is perpendicular to the generating line of the cone and has approximately the same area as that end of the cone". This definition is obviously not exhaustive, but due to the composability of logical rules, it can easily be extended. Additional definitions for e.g. box-shaped containers can be added, and queries will return the union of the results of all definitions.

Planar surfaces	**Spheres**
normalDirection (vector)	*radius* (float)
objectLongSide (vector)	*volumeOfObject* (float)
objectShortSide (vector)	*areaOfObject* (float)
areaOfObject (float)	*areaCoverage* (float)
areaCoverage (float)	*ConcaveTangibleObject*
SupportingPlane (computable class)	(computable class)
Cones/cylinders	**Containers**
radius (average radius, float)	*volumeOfObject* (float)
maxRadius (float)	*longitudinalDirection*
minRadius (float)	(opening direction, vector)
volumeOfObject (float)	
lengthOfObject (float)	**Handles**
longitudinalDirection (vector)	*Handle* (computable class)
areaOfObject (float)	
areaCoverage (float)	

TABLE I
Properties defined for the different object part annotations.

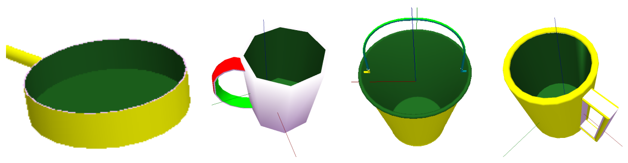
Fig. 4. Classification of containers (dark green color). Note that the container of the cup is recognized despite its hexagonal shape.

Based on the abstract definition, the system searches for planes that intersect with the generating line of a concave cylinder and checks if the angle is approximately 90 degrees. For those candidates, it verifies that the intersection point between the generating line and the plane is close to the end of the cone and that the cone is open at one end. Figure 4 shows examples of containers that were identified using this definition.

B. Identifying Handles

We focus on two kinds of handles for our analysis: On the one hand, we consider (largely) cylindrical handles in a (configurable) diameter range that can be chosen depending on the robot's gripper. On the other hand, we identify convex object parts that are smaller than the robot gripper's opening range. For the former type of handle, we rank cones by a weighting function that considers the minimum/maximum radius (in our experiments 0.4–4cm) and minimum/maximum length (1–80cm) that a handle needs to have. The resulting value is further influenced by the fitting error of the cylinder and its area coverage weighted by a sigmoid function. For the second kind of handle, convex object parts are identified by combining neighboring annotations of geometric primitives if the angle between the normals of triangles along the edge is larger than 180 degrees (corresponding to a convex shape). Afterwards, a cone is fit to the convex shape and the same weighting method is applied to check whether it is suitable as handle.

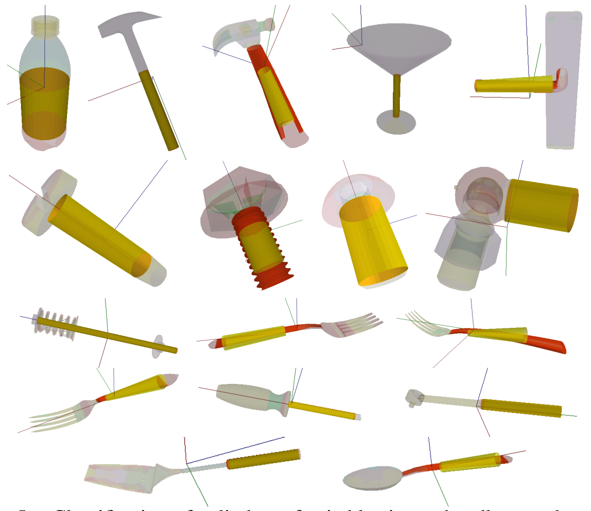

Fig. 5. Classification of cylinders of suitable size as handles, as abstractly defined in the knowledge base.

C. Locating Supporting Planes

Supporting planes, i.e. horizontal planes that objects can be put upon (Figure 6), are an important concept for robots, for example for putting down objects. Whether a plane qualifies as supporting plane depends on the object's pose; we therefore evaluate this concept only on object instances whose pose in the environment has been determined. We consider a planar surface as 'supporting plane' if its normal vector, considering the estimated current object pose, deviates by less than ten degrees from the global z-axis.

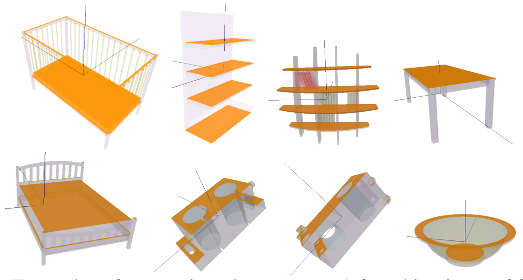

Fig. 6. Examples of supporting planes (orange) found in pieces of furniture, industrial parts and household items.

VIII. Object Recognition and CAD Model Matching

As the final step, in order to apply the results of the geometric and semantic interpretation techniques, a robot needs to be able to fit the CAD models to the perceived sensor data. The poses of the object components are described relative to the object's main coordinate system and can, once the object's pose in the environment has been determined, be transformed into global coordinates and used, for example, to identify the handle of an object. Although the model fitting to real scenes is not the focus of this work, we consider it an integral role in the context of robots finding objects. For this reason we briefly present a possible approach for fitting some of the segmented 3D CAD models to simple table top scenes (Figure 8).

A detailed processing pipeline is shown in Figure 7. For acquiring and preprocessing our scans we rely on the Point

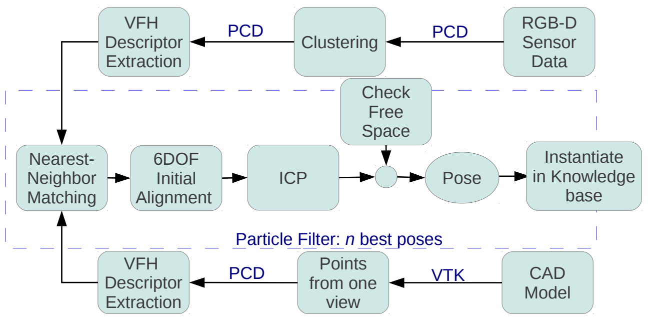
VFH Descriptor Extraction
PCD
Clustering
PCD
RGB-D Sensor Data
Check Free Space
Nearest-Neighbor Matching
6DOF Initial Alignment
ICP
Pose
Instantiate in Knowledge base
Particle Filter: n best poses
VFH Descriptor Extraction
PCD
Points from one view
VTK
CAD Model

Fig. 7. Pipeline for fitting CAD models to sensor data.

Cloud Library (PCL) [28]. In an offline phase we generate synthetic partial views of the CAD model from known poses and extract VFH descriptor [29] for each. During execution, after a successful segmentation of the object, we find the best k matches from the trained models using nearest-neighbors matching. The best model is selected using a combination of RANSAC and ICP as in [11]. This is aided by the "visibility scoring" described in [30] to reject fits that would mean that large parts of the model end up in front of or behind the scanned points. After the successful fit of the CAD model in the real scene, identifying the previously extracted semantically interesting object parts comes down to a simple coordinate transformation.

IX. Evaluation

Our evaluation includes different aspects: First, we quantitatively evaluate how well the segmentation and geometric primitive fitting identifies functional object components. We then present the whole pipeline from sensor data to functional models in the knowledge base and show queries that can be answered based on this information.

A. Identification of functional parts

We evaluated the system of a set of 337 object models downloaded from the Google 3D warehouse and the 3D-Net database (`http://3d-net.org`). The diverse set includes kitchen tools, silverware, cooking vessels, pieces of furniture as well as industrial parts. For each of the models, we generated the top three annotations by area or quality (in case of handles) and manually annotated the results regarding two criteria:

a) Are position, orientation & scale correct? For each annotation type and model, this score between 0 and 1 indicates how many of the top three annotations are correctly fit (Quality of fit).
b) Are the most relevant object parts among the top three components? This score between 0 and 1 rates which amount of the most relevant object parts are among the top three annotations (Relevance).

While b) is a partly subjective measure, it is important to verify that the most relevant functional parts (from a human point of view) have been detected. We concentrate on the three most significant components of each type to avoid issues with small annotations that may be correct from a geometric perspective, but do not carry much semantic

Annotation type	# annotations	Relevance	Quality of fit
Cone	938	0.8549	0.7833
Plane	825	0.7660	0.9082
Sphere	666	0.8739	0.6997
Container	291	0.3829	0.5381
Handle	480	0.5100	0.7735

TABLE II
EVALUATION OF THE OBJECT SEGMENTATION REGARDING QUALITY OF FIT AND COVERAGE OF RELEVANT PARTS.

information (e.g. cylinders fit to rounded edges). Table II lists the results of the evaluation. Examples of the matches can be found in the figures throughout the paper. One can see from the results that the primitive annotations (upper rows) are better recognized than the composed ones (containers and handles). The reasons are that the latter ones require all of the underlying primitive annotations to be correct, and that handles depend on the object's scale. We have not adjusted the scale of the downloaded models, but random samples among failed examples showed that often the model's scale was not correct so that the "handle" did not fit the given dimensions (i.e. did not fit into the robot's gripper).

B. Semantic queries about observed object scenes

The main contribution of this paper is to enable robots to reason about the functional parts of objects in front of them. We therefore evaluate the whole system from perception of objects to semantic queries about the scene. Figure 7 introduced the pipeline for matching CAD models of objects to sensor data, for computing their poses and for instantiating the objects in the knowledge base. Once these object instances are created, we can ask Prolog queries about their parts and properties that are answered based on the functional object model (which is generated on the fly from the CAD models associated with the objects). Figure 8 shows examples of scenes observed by the robot (top row), the aligned CAD models (second row), and selected functional components in the bottom rows. These functional components are the results of semantic queries that will be explained in more detail in the following sections.

a) Selecting appropriate containers: When pouring liquids into a container, a robot needs to select a container that is larger than the volume of stuff to be poured into it. This kind of common-sense knowledge can be formulated as a rule and can be used to select among different containers in a scene, in this case between the cup and the cooking pot. For example, the robot can ask for objects that have a container with a volume of at least 1 liter (0.001 m^3) as part:

```
?- owl_has(Obj, kr:properPhysicalParts, C),
   owl_individual_of(C, kr:'Container'),
   rdf_triple(kr:volumeOfObject, C, V),
   V > 0.001.
Obj = kr:'pot1',
C = kr:'ContainerArtifact_FqDosfsb',
V = 0.00293
```

b) Determining which surface to pour batter on: When pouring batter onto the pancake maker, the robot should use the largest area on the top of the pancake maker, i.e. the largest supporting plane. The same holds in general for pouring something onto something, e.g. oil onto a pan, so we can formulate the following rule:

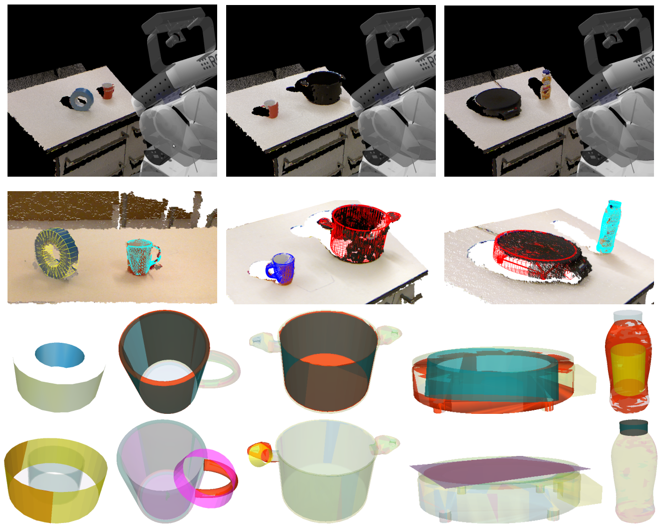

Fig. 8. Top row: View of the PR2 robot watching the tabletop scenes. Second row: CAD models fit to point clouds perceived by the robot. Bottom rows: Segmented CAD models; the results of the Prolog queries are highlighted.

```
pour_onto(Obj, Part) :-
    findall(A-P,
      (rdf_triple(kr:properPhysicalParts, Obj, P),
       rdfs_instance_of(P, kr:'SupportingPlane'),
       rdf_triple(kr:areaOfObject, P, A)), Planes),
  keysort(Planes, PlanesAsc),
  last(PlanesAsc, _-Part).

?- pour_onto(kr:'maker1', Part).
Part = kr:'FlatPhysicalSurface_UosqOAfb'.
```

c) Finding grasping points: For many objects, their handles are the preferred grasping points. Knowledge about the composition of objects and the positions of their handles can therefore be used for determining where to grasp them.

```
grasp_point(Obj, GraspPoint) :-
  rdf_triple(kr:properPhysicalParts, Obj, Handle),
  rdfs_instance_of(Handle, kr:'Handle'),
  annotation_pose_list(Handle, GraspPoint).

?- grasp_point(kr:'pot1', P).
P = [ 0.001, 0.062,-0.9980,-0.173,
     -0.998, 0.062, 0.0019,-0.109,
      0.062, 0.996, 0.0628, 0.115,
      0.000, 0.000, 0.0000, 1.000] ;
```

d) Identifying bottle caps: Virtually all bottles and many other kinds of containers like jars are closed with a cylindrical screw cap that is close to the top of the object. The following Prolog rule defines a cap as the topmost cylinder and enables a robot to identify this important object part:

```
bottle_cap(Obj, Cap) :-
    findall(Z-P,
      (rdf_triple(kr:properPhysicalParts, Obj, P),
       owl_individual_of(P, kr:'Cone'),
       objpart_pos(P, [_,_,Z])), ConePos),
    keysort(ConePos, ConePosAsc),
    last(ConePosAsc, _-Cap).

?- bottle_cap(kr:'pancakemix1', Cap).
Cap = kr:'Cone_vcRxyUbK'.
```

X. Conclusions

In this paper, we presented a system that enables robots to use CAD models of objects as knowledge source and to perform logical inference about object components that have automatically been extracted from these models. The system includes several algorithms for mesh segmentation and geometric primitive fitting that are integrated into the robot's knowledge base as procedural attachments to semantic representations. The bottom-up segmentation is complemented by a top-down, knowledge-based analysis of the resulting components to determine semantically meaningful components based on abstract, symbolic specifications in the knowledge base. The evaluation on a diverse set of object models, downloaded from the Internet, shows that the algorithms are able to reliably detect the different kinds of object parts, and that these models can be turned into a useful knowledge resource for autonomous robots. We expect that the system will help robots to better apply common-sense knowledge by grounding abstract symbols in geometric object models. The functional models complement the original object CAD models that remain available for perception and visualization purposes.

Acknowledgments

We would like to thank Zoltan-Csaba Marton, Emal Sadran and Maximilian Wenger for their help on the model fitting part for real scenes. This work is supported in part by the EU FP7 Projects *RoboHow* (grant number 288533) and *RoboEarth* (grant number 248942).

References

[1] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth, "Robotic Roommates Making Pancakes," in *11th IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, October, 26–28 2011.

[2] J. Hollerbach, M. Mason, and H. Christensen, "A Roadmap for US Robotics – From Internet to Robotics," Computing Community Consortium (CCC), Tech. Rep., 2009.

[3] R. Bischoff and T. Guhl, Eds., *Robotic Visions to 2020 and Beyond – The Strategic Research Agenda for Robotics in Europe*. European Robotics Technology Platform (EUROP), 2009. [Online]. Available: http://www.robotics-platform.eu

[4] Bicchi, A., et al., "Research Roadmap," EURON – European Robotics Network, Tech. Rep. DR.1.3, 2007. [Online]. Available: http://www.euron.org/miscdocs/docs/year3/DR.1.3.pdf

[5] M. Tenorth, D. Nyga, and M. Beetz, "Understanding and Executing Instructions for Everyday Manipulation Tasks from the World Wide Web," in *IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, USA, May 3–8 2010, pp. 1486–1491.

[6] J. J. Gibson, *The Theory of Affordances*. John Wiley & Sons, 1977.

[7] N. Dag, I. Atil, S. Kalkan, and E. Sahin, "Learning affordances for categorizing objects and their properties," in *20th International Conference on Pattern Recognition*, ser. ICPR '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 3089–3092.

[8] M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. A. Sucan, "Towards reliable grasping and manipulation in household environments," in *Proceedings of RSS 2010 Workshop on Strategies and Evaluation for Mobile Manipulation in Household Environments*, 2010.

[9] M. Nieuwenhuisen, J. Stückler, A. Berner, R. Klein, and S. Behnke, "Shape-primitive based object recognition and grasping," *ROBOTIK 2012*, 2012.

[10] H. Kjellström, J. Romero, and D. Kragic, "Visual object-action recognition: Inferring object affordances from human demonstration," *Computer Vision and Image Understanding*, vol. 115, no. 1, pp. 81 – 90, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S107731421000175X

[11] A. Aldoma, F. Tombari, and M. Vincze, "Supervised learning of hidden and non-hidden 0-order affordances and detection in real scenes," in *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN, USA, May 14–18 2012.

[12] I. Kresse, U. Klank, and M. Beetz, "Multimodal autonomous tool analyses and appropriate application," in *11th IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, October, 26–28 2011.

[13] M. Attene, B. Falcidieno, and M. Spagnuolo, "Hierarchical mesh segmentation based on fitting primitives," *The Visual Computer*, vol. 22, no. 3, pp. 181–193, 2006.

[14] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis, "3D Mesh Segmentation Methodologies for CAD Applications," *Computer-Aided Design & Applications*, vol. 4, no. 6, pp. 827–841, 2007.

[15] R. Bénière, G. Subsol, G. Gesquière, F. Le Breton, and W. Puech, "Recovering primitives in 3d cad meshes," in *Proceedings of SPIE*, vol. 7864, 2011, p. 78640R.

[16] M. Attene, S. Katz, M. Mortara, G. Patané, M. Spagnuolo, and A. Tal, "Mesh segmentation - a comparative study," in *IEEE International Conference on Shape Modeling and Applications (SMI)*. IEEE, 2006.

[17] M. Garland, A. Willmott, and P. S. Heckbert, "Hierarchical face clustering on polygonal surfaces," in *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, ser. I3D '01. New York, NY, USA: ACM, 2001, pp. 49–58.

[18] F. Tombari and L. D. Stefano, "Automatic semantic segmentation of 3d urban scenes," in *3D Imaging, Modeling, Processing, Visualization and Transmission Conference (3DIMPVT 2011)*, May 16-19 2011.

[19] F. Tombari, L. D. Stefano, and S. Giardino, "Online Learning for Automatic Segmentation of 3D Data," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[20] S. Lee, K. D. Yoo, J. W. Kim, and M. J. Lee, "Surface patch primitive based object modeling from cad data," *Applied Mechanics and Materials*, vol. 162, pp. 179–183, 2012.

[21] I. Biederman, "Recognition-by-components: a theory of human image understanding," *Psychological review*, vol. 94, no. 2, p. 115, 1987.

[22] M. Tenorth and M. Beetz, "KnowRob – Knowledge Processing for Autonomous Personal Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 4261–4266.

[23] J. Wielemaker, T. Schrijvers, M. Triska, and T. Lager, "Swi-prolog," *Theory and Practice of Logic Programming*, pp. 67–96, 2012.

[24] W3C, *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. World Wide Web Consortium, 2009, http://www.w3.org/TR/2009/REC-owl2-syntax-20091027.

[25] S. Rusinkiewicz, "Estimating curvatures and their derivatives on triangle meshes," in *2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT 2004)*, 2004, pp. 486–493.

[26] R. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Information processing letters*, vol. 1, no. 4, pp. 132–133, 1972.

[27] D. Eberly, "Least squares fitting of data," 2008. [Online]. Available: http://www.geometrictools.com/Documentation/LeastSquaresFitting.pdf

[28] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011, pp. 1–4.

[29] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010.

[30] O. M. Mozos, Z. C. Marton, and M. Beetz, "Furniture Models Learned from the WWW – Using Web Catalogs to Locate and Categorize Unknown Furniture Pieces in 3D Laser Scans," *Robotics & Automation Magazine*, vol. 18, no. 2, pp. 22–32, June 2011.