A Model-Based Approach to Finding Substitute Tools in 3D Vision Data

Paulo Abelha[1], Frank Guerin[1], Markus Schoeler[2]

***Abstract*— A robot can feasibly be given knowledge of a set of tools for manipulation activities (e.g. hammer, knife, spatula). If the robot then operates outside a closed environment it is likely to face situations where the tool it knows is not available, but alternative unknown tools are present. We tackle the problem of finding the best substitute tool based solely on 3D vision data. Our approach has simple hand-coded models of known tools in terms of superquadrics and relationships among them. Our system attempts to fit these models to point clouds of unknown tools, producing a numeric value for how good a fit is. This value can be used to rate candidate substitutes. We explicitly control how closely each part of a tool must match our model, under direction from parameters of a target task. We allow bottom-up information from segmentation to dictate the sizes that should be considered for various parts of the tool. These ideas allow for a flexible matching so that tools may be superficially quite different, but similar in the way that matters. We evaluate our system's ratings relative to other approaches and relative to human performance in the same task. This is an approach to knowledge transfer, via a suitable representation and reasoning engine, and we discuss how this could be extended to transfer in planning.**

I. INTRODUCTION

We are interested in service robots tackling tasks in everyday home environments. Such environments are open and unconstrained in the sense that the objects and materials available may not be known in advance (although they can be expected to be variations on known things). Humans in such environments transfer knowledge, to apply their known skills to variations on known objects, with appropriate adjustments. Humans frequently improvise, opportunistically exploiting affordances of utensils that diverge from the canonical usage. For example a knife with a sufficiently wide blade can be slipped under a piece of cake to lift it; a chopstick with a sufficiently pointy tip can be used to pierce food. Consider in more detail the following example: the required manipulation is crushing an object with force with a mallet as the usual tool (for example tenderising meat). When a mallet is not available, one can consider other possible candidates: a frying pan has a handle, a flat-base and is of a suitable size to be manipulated to crush something. Could a robot see the frying pan as a possible mallet? In order to do that, it must reason about the essential parts of the mallet, the relationship between them (handle, hitting surface, distance and angle between), and whether or not the frying pan has available parts that could serve the same function. See Fig. 1 to get a concrete idea of our system.

[1]Paulo Abelha and Frank Guerin are based at the Department of Computing Science, University of Aberdeen, King's College, AB24 3UE Aberdeen, Scotland. p.abelha/f.guerin@abdn.ac.uk

[2]Markus Schoeler at Department for Computational Neuroscience Georg-August-Universität Göttingen, Germany. mschoeler@gwdg.de

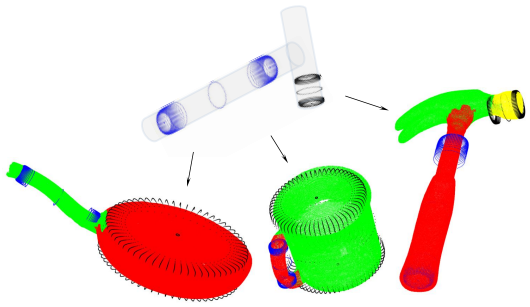

Fig. 1. Typical fits by our system for projecting an idealised model of a hammer (source) onto a hammer, mug, and frying pan (target objects). Each model part is projected into a segment of the point cloud. The task model defines how various distortions of parts, and the relationship among parts, are to be penalised.

This problem is important because it is not feasible to endow robots with knowledge of every tool they may encounter in open environments. It is possible to endow them with knowledge of a limited set of tools, and then it is important that they are able to apply this knowledge in novel situations, and to adapt to similar tools. This problem goes beyond standard paradigms in computer vision, which tend to focus on determining the single category to which an object belongs. In our setting we do not want to be limited to seeing a knife as a knife or a pan as a pan; we are looking for alternative interpretations, directed by a task need. This is a small part of the more general problem of finding suitable representations (for objects in this case) and reasoning processes that facilitate transfer of knowledge to unforeseen settings.

The key innovation we introduce to robotic vision stems from work in psychology on *projection* [1], which explains how we organise the perception of sensor data through a top down process that imposes a certain structure onto lower level data. We try to see what we want to see, and this may work to varying degrees, as shown in Fig. 1.

High-level knowledge approaches already exist to find tools in the same class, using ontologies, or using a database of models of known objects, and these are useful to find a set of tools to perform a particular manipulation. However this approach will miss tools from a different class that may possess parts which make them suitable for the particular manipulation task in question. On the other hand, low level approaches learning similarity from 3D vision features [2], [3] will struggle to capture tool affordances that rely on a particular relationship between parts, where those parts may vary considerably.

To tackle this problem it seems necessary to construct a

system connecting high level (i.e. a model) and low level (i.e. a point cloud) information, through a process of approximate matching that embodies some knowledge of what aspects of a tool are important for a particular task. In this way our system goes beyond typical model fitting in computer vision [4]. In this paper we focus purely on shape similarity (similar to [2], [5]) and neglect other aspects such as materials or weight; this is in order to conduct a clear investigation on one aspect in isolation. The scientific hypothesis tested in our paper is whether a part based geometric representation (model) of an ideal tool, together with knowledge of the importance of aspects of that model for a task and an algorithm for part fitting, can approximate human performance (for accuracy) in the task of rating the usefulness of a novel tool for a particular task.

Our system has three essential components: (i) Hand-coded models of ideal objects for tasks; this is similar to several works in this area defining how objects are composed from parts [6], [7], [8]. (ii) Task models that specify the relative importance of various parameters of the model for a particular task. (iii) A matching algorithm that tries multiple fits in the point cloud, converging toward that preferred by the task model. We evaluated our approach on 3D scans from everyday household objects, comparing our system with ground truth from performing the tasks, human ratings and also with two variants of an approach based on histogram features of shapes of parts. Our results show high accuracy. Our main limitation is that we did not discover the relative importance of parameters of the models from data by machine learning; instead we have crude hand-coded rules in the task model. A data driven approach would be superior at finding optimal parameters.

II. RELATED WORK

There has not been a great deal of work on finding similar/substitute tools in vision for robotics, although our work can be seen as coming under the broader umbrella of finding affordances of objects in the environment (e.g., [2],[3]). These affordance approaches are based on learning from visual features, and do not incorporate the higher level knowledge of essential parts and relationships that we encode in our system. Hence they will struggle to see the functional equivalence between objects that might be superficially quite different in appearance, although having the essential required parts in the right relationship.

Most works on affordances are not designed to deal with object parts; e.g. Myers et al. [2] just consider one part for things like sharp or concave. We can consider a relationship between a hammer handle and how far it is from the head, and at what angle (however this is not the only paper to consider multiple parts, e.g. see [7], [9], [10], [11]).

The approach of Biegelbauer et al. [7] was the starting point for the construction of our system. Biegelbauer and Vincze's system is able to quickly fit geometric models of common shapes (they use superquadrics, here explained in Section III) onto point cloud data. Although Biegelbauer and Vincze's work is not explicitly on finding similar tools, their system will fit to the most similar object when the exact object is not present. In contrast with Biegelbauer and Vincze's work we explicitly design ours to find good substitute tools, e.g. by providing knowledge on which parts of the source model are most important and less important, and what matches are acceptable.

Tenorth et al. present a closely related system [6] that defines rules for how geometric primitives are composed for various kitchen objects. However their aim is to find the canonical category of a tool, not to consider also alternative uses. While their system does combine elements of top-down and bottom-up processing, ours has a stronger top-down influence where the type of geometric primitive we project is decided top-down.

An interesting work by Fitzgerald et al. [12] is able to find analogies between tool-use situations from 2D images. However the techniques used are all 2D based and hence very different to our work and other work cited here, making comparison difficult. Fitzgerald et al.'s more recent approach [13] is a little closer to our approach at a high level as it looks at transfer from source situations to target situations where parts and relationships are explicitly represented (although their parts are objects whereas ours are parts of objects).

Schoeler and Wörgötter [5] have a system that uses 3D features to find shape signatures of parts of a tool; tools are described by a graph and can have multiple connected parts. Tools are then compared pairwise for similarity, and they show that this does find tools with similar functions. This system can also find non-standard uses of a tool, e.g. a hammer's shaft can be used to poke holes in the ground. This aspect is like our system. Shapira et al. [14] present a similar (shape features and graphs of parts) but more complex system, able to find part-based similarities in a database of 3D objects (e.g. a finger is to a human as a wing is to a plane). Our system has a different focus from these two works because of the projection idea: Our search for a substitute tool is directed: We have a model of what we want, like mallet, and we are projecting that onto whatever sensor data we are looking at, in an attempt to force it to be seen like a mallet. It is not a symmetrical similarity between two objects; while a mallet projects reasonably well onto a frying pan the projection would be poor in the other direction. During the process of projection the following features of the system are of note:

1) We can allow flexible matching; e.g., we do not always have to see the handle as the handle, we could use it as the 'business end' (similar to [5]); we do not have to insist on curvature, we can be relaxed and allow a flat surface to fit.
2) We include task relevant knowledge; e.g. specifying importance of components and tolerances for size/angle between parts. Other works sometimes have knowledge like categories of tools (e.g. ontologies), where this knowledge is separate from the 3D models, we integrate knowledge into the 3D models. Therefore we know what parts of the model we can be more relaxed about fitting, for a particular task.

3) We combine top-down pressure from the model's definition with bottom-up indications from the sensor data (e.g. changing the scale of parts to better fit the data).

III. SYSTEM DESCRIPTION

A. Overview

- Input:
  - A *source* model specifying size and shape of *grasping* and *action part* and their relationships (angles and distances);
  - A *target task* model specifying how important are the sizes, shapes and relationships from the model;
  - A set of point clouds as the *target objects* (these are segmented from ground, but not yet part-segmented).
- Output: A score value for each of the target objects in a scale from 1 to 10 representing how good that object is as a substitute tool for the task.

B. Source models and target tasks

The source model specifies the ideal shape, size and part-relationships of a tool; in this work, our source models have only two parts: grasping and action. Parts are modeled with superquadrics (Sec. III-F). We define relationships among the parts as: distance between the parts' centres; angle between both parts' main axes; and angle between grasping part's main axis and the axis that passes through both parts' centres.

We have five source models: an idealised hammer, spatula, sharp chopstick, rod for retrieving, and teaspoon. Each model consists of two superquadrics and the relationships between them. We have seven target tasks:

1) Hammer a small nail into soft wood with a swinging action (hammer source model)
2) Tenderise meat with swinging action (overrides hammer source model to define a larger size for action part)
3) Lift a pancake from a pan (spatula source model)
4) Cut soft lasagne (overrides spatula source model to define an angle of 0 between parts)
5) Pierce a potato skin (chopstick source model)
6) Retrieve an item from a narrow gap, e.g. under a fridge (retrieving tool source model)
7) Scoop sugar from a small bowl whose aperture is just big enough for the teaspoon (teaspoon source model)

C. Target task models

Target task models specify functions for scoring how well a target object is likely to perform on the target task. It scores the part parameters (biggest, medium and smallest dimensions, biggest/medium proportion, and superquadric fit) and part-relationships (distance, angle between centers and angle between z axis). For each part parameter we firstly calculate the absolute value of the difference from the ideal value of that parameter in the source model, and this is all that is done for most cases. The target task may also override the source model's ideal values in the case where one model is used for different tasks (the two hammering tasks). For some task functions (table below) an additional linear or quadratic function is applied for values below or above the ideal (penalising more or less a given feature), or cut-off to prohibit certain values. The following table shows the task functions applied to the action part:

Task	Big		Medium		Small		Distance	
Hammer nail	*quad*	*lin*	*quad*	*lin*	*quad*	*lin*	*quad*	*lin*
Tender. meat	*quad*	*lin*	*quad*	*lin*	*quad*	*lin*	*quad*	*lin*
Retrieve item					*lin*	*cutoff*	*quad*	*lin*
Scoop sugar					*lin*	*cutoff*		

In each table cell the first string specifies the function applied when the value is smaller than ideal, and the second string specifies the function when larger than ideal. In all unspecified cases we used a simple absolute difference from the ideal value. For quadratic (*quad*) functions, in addition to squaring the value we multiplied by 40 and for linear (*lin*), by 0.1. We have a cut-off function for retrieving from gap and scooping sugar because the dimensions must be suitable for the size of the gap or container: smaller values are only slightly worse, but larger values are prohibitive. For hammering, we have a quadratic scoring if they are below ideal; i.e. the hammering task heavily penalises a too small hammer head. For larger than ideal values a linear function is applied which penalises weakly.

The system starts with a source model and first performs a specified number of fittings of each part to each segment, keeping the best overall fit (Sec. III-F). After fitting, the next step is to calculate the target score for each pair (grasping part and action part) of part-to-segment-fit, considering both the source model and the target task. This process constitutes one *projection* of the source model onto the target object. In Fig. 1, we can see, for three different target objects (hammer, frying pan and mug) the best projection for source model hammer and target task of hammering a nail (grasping part is blue and action part is black).

D. Obtaining Pointclouds from Real Objects

We used an Artec Eva 3D scanner and Kinect 2. Artec works by projecting structured light (similar to Kinect 1), while Kinect 2 uses time of flight. We chose both a high-end and low-end scanner in order to see how dependent our technique is on high precision and complete visual input. We scanned (separately) 20 household objects on a flat surface; see Fig. 2. As can be seen from the figure some objects have some scan errors (see e.g. meshed spatula, where the difficulty arises from the actual object being completely black).

Artec scanning: With the Artec scanner we used real-time fusion while moving around the object completely. We subsequently applied a pipeline of algorithms to remove outliers, fill holes, and smooth (a process which could be automated). The resulting scans are almost complete, only missing a base or other patches that could only be seen from underneath. Some objects were supported with plasticine to

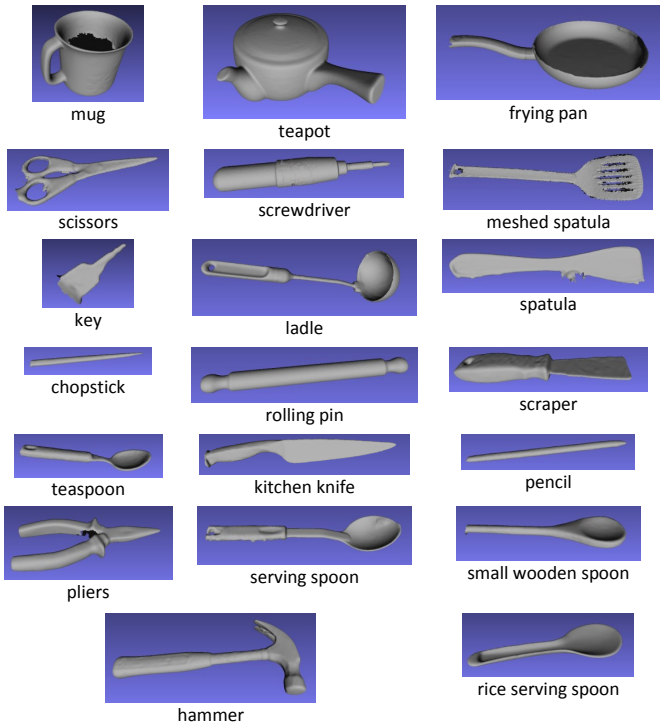

Fig. 2. Artec scans of our 20 household objects, not to scale. Black portions indicate scan errors (missing parts of the actual object).

maximise the visible surfaces, and some artefacts of this can be seen in the scans (see wooden spatula for example). For seven of the 20 objects we also inverted the objects to obtain a good scan of the hidden side and fuse the models; this was for models where we felt the base could be important. Our approach of supporting and scanning from all sides can be integrated in a real robotics setup where the robot grasps the object and rotates it, and similar work has been done [15], [16].

Kinect 2 scanning: Objects were left flat on a table with no supports in most cases; however some objects like the wooden spatula were so flat that they disappeared into the table so we needed to raise them with plasticine underneath (but we did not change the orientation). We used real-time fusion while moving over the object in a single arc from one side, over the top, to the other side. The entire process took 10 seconds. This was done to mimic a real robot analysing an object on a table. RANSAC plane removal was then applied to remove the table. These scans are of much poorer quality primarily due to Kinect interpolating and partly merging an object into the table, and also due to half the object being missing for many objects because it was resting on the table (and RANSAC removal takes more than just the plane). Nevertheless the objects have sufficient points and fragments of shape to permit superquadric fitting, as illustrated in Fig. 7. Most roughly cylindrical objects have at least half a cylinder after processing. Small objects could not be captured at all; they appear as nothing more than a slight depression in the table.

Shiny surfaces: Some parts of objects that were of highly specular chrome surfaces posed some difficulty for the Artec scanner, although it had no problem with others (e.g. the scissors had a shiny chrome surface but posed no problem). The most problematic surface was the central part of the shaft on the hammer. Fortunately it had a manufacturer's sticker on one side, and care with scanning to catch the specular surface directly (not obliquely) at the beginning helped. The standard hole filling and smoothing in the post processing stage also helped. Painting objects obviously removes the problem; we found that this is not necessary for Artec, but absolutely essential for Kinect. In Fig. 3 we compare various scans of the teaspoon. As can be seen from Fig. 3 the Kinect scan is poor even with paint. The Artec scanner had absolutely no problem with brushed metal surfaces (e.g. hammer head or kitchen knife), or with ceramic. However these needed to be painted for Kinect. For Kinect 2 we estimate that 85% of items in a typical kitchen cannot be adequately seen. Kinect is not a viable scanner for a robot dealing with regular kitchen objects. The only thing Kinect really worked well on (without painting) is plain wood (rolling pin, spatula). Nevertheless we still wanted to see how our system performs with lower quality pointclouds, so we obtained as many pointclouds as we could with Kinect, by painting problematic objects.

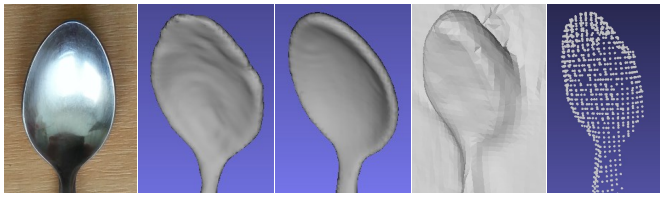

Fig. 3. **From Left**: (i) Photograph of a teaspoon. (ii)-(v) are rotated at a slight angle to highlight imperfections. (ii) Artec scan of untreated surface. (iii) Artec scan of painted surface. (iv) Kinect scan of painted spoon on table before RANSAC plane removal. (v) Kinect scan after plane removal. (Kinect scan of unpainted spoon is not possible to obtain.)

Note that while the set of objects is not huge they are highly varied. Note also that these objects are the 'test set'; there is no training set, as no learning was used, instead the source model of tool and task are handcoded.

E. Segmentation

For segmentation, we use the Constrained Planar Cuts (CPC) algorithm [17], which uses local concavities to induce cuts through the object and partition it in segments. We scale the parameters to each object size and iteratively increase the smoothing parameter in the CPC algorithm until achieving two or three segments (ignoring segments which were too small). In Figs. 1, 6, and 7, it is possible to see for each point cloud the different segments in different colours.

F. Fitting

We model parts of the source object with superquadrics, which are parametric shapes capable of representing a variety of common 3D objects (e.g. cylinder and boxes), and many shapes in between. We refer to [18, p. 79] for an in-depth explanation of superquadrics. In this work, we define a superquadric through a set Λ of eleven parameters $\lambda_1, .., \lambda_{11}$

$$\Lambda = \{a_1, a_2, a_3, \epsilon_1, \epsilon_2, \phi, \theta, \psi, K_x, K_y, p_x, p_y, p_z\}$$

Three parameters a_1, a_2, a_3 for the scale in each dimension; two parameters for shape variance ϵ_1, ϵ_2; three parameters for ZYZ Euler angles , ϕ, θ, ψ; two parameters for tapering in the x- and y-$axis$ along the z-$axis$ K_x, K_Y; and three parameters for central point p_x, p_y, p_z. Superquadrics also provide an inside-outside function $F(x, y, z) = F(x, y, z; \Lambda)$ that tells whether a given point (x, y, z) is inside ($F < 1$), on the surface ($F = 1$) or outside ($F > 1$) the superquadric. Suppose we have a set of n 3D points obtained from a range sensor, then, the superquadric that best fits this data can be obtained from the solution of the following minimization problem:

$$\min_\Lambda \sum_{i=1}^{n} (\sqrt{\lambda_1 \lambda_2 \lambda_3} (F^{\lambda_4}(x_i, y_i, z_i; \lambda_1, ..., \lambda_{11}) - 1))^2$$

This minimzation leads to a set of parameters Λ for which the superquadric has the most points close to its surface. We perform the minimization using the Levenberg-Marquadt method[1] that allows us to set lower and upper bounds on parameters. As finding a global optimum is non trivial [18], to improve fitting results, as in [7], we plant 20 random initial seed points for the centre position $(\lambda_{11}, \lambda_{12}, \lambda_{13})$ and after fitting for each point, we perform a ranked voting considering the value of the inside-outside function F, the percentage of points inside and the percentage of points on the surface; we choose the superquadric according to a small function score, a low percentage of points in the interior and a high percentage of points on the surface.

For the initial Λ of the superquadric we use values coming from the model for shape and tapering. We get the initial orientation by running PCA on the segment. The initial scale parameters come from the segment: we run PCA on the segment and for each dimension, we slice the segment in a predefined number of slices and get the median range value as the segment scale for that dimension; this helps us get rid of outlier parts of the segment. We allow a $\pm 20\%$ variation in the initial scale. In the case of non-tapered models, we give a 50% chance for having the parameter optimized; when the model does not have tapering, the tapering parameters are initialized to 0 and not optimized.

We run 10 projections in order to get a variety of different fits of source parts to target parts. Projections with similar results are clustered together so that we consider any projection in a cluster as being representative of the same unique projection. We choose the cluster with most instances and apply the target task functions to get the score.

This full process of several projections is carried out for each target object. Comparing the ratings from different objects at this stage causes difficulty due to there being orders of magnitude differences in the scores produced. We therefore apply a ranking to rank all task function results (e.g., for sizes, distance, angles). The sum of these ranks becomes the final score, and all the objects' scores are scaled to the $[1, 10]$ range.

[1]Using the implementation in MATLAB R2015a's Optimization Toolbox

IV. EXPERIMENTAL EVALUATION

The task was to rate 20 objects (from point clouds) for their usefulness in each of seven tasks. We compared six different sets of results:

1 Human rating. We asked humans to rate the usefulness of the 20 test objects for each of the tasks on a 1 to 10 scale as follows: 1-5 Does not work; 6-8 Could work, but not the best; 9-10 Works best. We decided to have ten numeric values in order to capture such distinctions as a tool that almost works, versus one that is utterly useless. Humans only got to see 2D rendered point cloud images. For some objects we included two images where one perspective did not make the shape clear. Images of a human performing the task on a target object were also provided. We also explained the action for example specifying the swinging type of action is important or else the human may consider alternate actions like thumping downwards while gripping the screwdriver with base pointing downwards, which do not correspond to the task relationships coded in our system. In order to focus on shape and not materials we told humans to assume all objects were made of solid metal. We averaged the ratings across the humans.

2 Ground truth. We took the physical objects and target items such as a nail and wood, and tried the objects on the tasks[2] in order to rate them. Since we had told the humans that objects were made of metal we sometimes used metal substitutes such as a long flat spanner in place of the wooden spatula. We compared each system rating through the formula $10 - |system - groundtruth|$, thus reflecting, on a scale of 1 to 10, how much the system deviates from ground truth.

3 Random rating. Random ratings serve as a baseline. Fig. 5 shows the approximately normal distribution from 300 complete sets of random ratings. Scoring higher than 7 is extremely unlikely by random answering ($p < 0.008$).

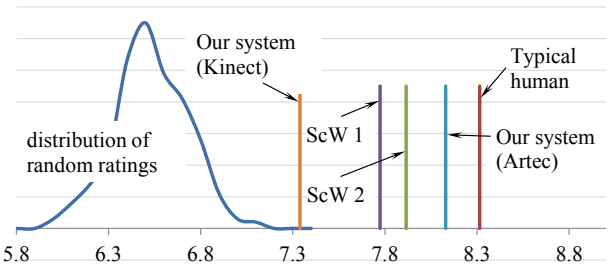

Fig. 5. Average accuracy (scale 1 to 10) of the systems evaluated.

4 Our system. For each target object for each task we ran ten projections with three iterations in each. In order to reduce running time (without much loss in accuracy), during each run the point cloud is downsampled to 2500 points per segment.

5 System ScW 1. Schoeler and Wörgötter's system [5] can be run in two ways. ScW 1 is the term we will use for the system trained on a labelled dataset M1 of 72 synthetic point

[2]For tenderising meat we hammered rough plasticine as a substitute. No animals were harmed in the making of this paper.

	Source task↓	Chopstick	Frying pan	Hammer	Kitchen knife	Key	Ladle	Mesh spatula	Mug	Pencil	Pliers	Rice spoon	Rolling pin	Scissors	Scraper	Screwdriver	Serving spoon	Sm. wd. spoon	Wood spatula	Teapot	Teaspoon	**Tool Average**
Our Artec results	Hammer nail	9	8.6	10	7.5	9.3	8.2	6.2	9.4	9.2	7.9	9.1	8.9	6.1	9.5	9.5	6.4	7.3	8	8.5	6.5	**8.26**
	Tender. meat	10	9	8.8	8.6	7.6	9.8	3.3	9.9	9.4	9.2	7.5	6.9	7.2	7.8	9.9	5.6	9	7.3	8.2	8.7	**8.17**
	Lift pancake	8	10	6.8	6.7	10	7.8	10	9	7.8	8.3	6.9	9.7	7.4	7.6	4	9.1	9.2	8.6	6.8	9.6	**8.16**
	Cut Lasagne	7.6	7.1	8.4	9	6	8.9	9.2	9.1	5.9	8.7	8.9	8.7	9.3	9.1	9.5	9.8	9.9	9.1	6.5	7.6	**8.41**
	Pierce potato	10	10	5.6	7.5	5.5	10	6.8	8.4	9.4	9.5	5.6	6.4	7.1	5.9	8.5	8.5	8.9	6.2	9.7	8.9	**7.92**
	Retrieve item	6.5	6	6.7	9	6.6	8.9	8	8.8	6.2	8.9	9	6.3	8.3	9.2	9	8.5	5.2	7.8	7.4	8.3	**7.73**
	Scoop sugar	9.3	7.4	9.6	9.6	9	7.9	8.5	8	9.3	6.3	7	6.4	9.2	9.2	5.5	9.5	6.6	9	7.5	10	**8.24**
	Task average	8.6	8.3	8	8.3	7.7	8.8	7.4	8.9	8.2	8.4	7.7	7.6	7.8	8.3	8	8.2	8	8	7.8	8.5	**8.13**
Our Kinect 2 results	Hammer nail		9.8	10	6.5						9.3	7.8	7.8	3.3	9.3	7.3	5.3	5.8	7.3		9	**7.54**
	Tender. meat		9	7.6	6.9						9.5	9	6.7	8.9	8.6	8.1	9.9	9.8	9.6		8	**8.58**
	Lift pancake		9.9	7.2	8						8.3	8.5	5.5	8	8	5.9	7.6	8.6	7.8		9.5	**7.92**
	Cut Lasagne		5.6	4.7	7.3						6.4	9	8.6	8.6	7.6	8.6	6.8	9.6	10		6.4	**7.64**
	Pierce potato		10	8.9	6						7.6	5.2	9.2	9.6	3.9	8	1	8.1	4.2		5.3	**6.69**
	Retrieve item		6	6.7	6.4						8.9	7.2	4	5.9	6.8	7.2	5	6.5	6.7		4.6	**6.3**
	Scoop sugar		6.6	4.9	8.7						3	8.6	2.1	8	2.6	9	9	9.6	9.6		5.2	**6.7**
	Task average		8.1	7.1	7.1						7.6	7.9	6.3	7.5	6.7	7.7	6.4	8.3	7.9		6.9	**7.34**
ScW 1 (from set)	Hammer nail	9.9	3	3.8	8.8	10	9.6	8.5	8.8	9.3	5.8	7.7	6	10	6.9	9	8.3	9.7	7.5	9.6	10	**8.11**
	Tender. meat	8.9	3	4.8	7.8	9	9.4	9.5	7.8	9.7	8.8	7.7	9	9	5.9	7	8.3	8.7	7.5	9.6	9	**8.02**
	Lift pancake	5.2	9.7	7.5	8.7	4.1	9.5	4.9	10	4.9	6.1	8.5	7.3	9.3	6	4.6	8.9	7.9	9.6	6.3	7.3	**7.31**
	Cut Lasagne	8.8	9.7	8.5	9.7	8.1	9.5	5.9	10	9.1	5.1	8.5	7.3	9.3	7	6.6	9.9	8.9	8.4	6.3	9.7	**8.32**
	Pierce potato	9.7	6.8	9.9	5.7	7	9	8.3	9.8	10	9.1	6.9	5.5	7.1	5.7	9.6	6.3	7.5	6.5	6.9	8.6	**7.79**
	Retrieve item	8.3	9.2	9.9	5.7	5	2	5.7	9.8	8	9.1	9.9	5.5	9.1	9.7	9.4	9.7	9.5	5.5	5.9	8.4	**7.76**
	Scoop sugar	7	4.5	9.7	7.3	6.6	8.1	8.9	1	6	9.9	7.8	6.7	8.3	9	9	8.1	3.5	9.1	6.2	5	**7.09**
	Task average	8.3	6.6	7.7	7.7	7.1	8.2	7.4	8.2	8.1	7.7	8.1	6.8	8.9	7.2	7.9	8.5	8	7.7	7.3	8.3	**7.77**
ScW 2 (matrix model)	Hammer nail	7.1	3.7	10	9.6	9.3	9.2	8.8	7	7.3	8.7	7	9.8	7.9	7.7	9.2	8.3	6	7.4	5.5	9.6	**7.96**
	Tender. meat	6.1	3.7	9	8.6	8.3	9.8	9.8	6	6.3	8.3	7	6.8	8.9	6.7	8.8	8.3	7	7.4	5.5	9.4	**7.59**
	Lift pancake	7.2	7.9	8.6	9.5	5	8.4	10	9.5	6.9	8	9.9	9.9	10	9.5	7.5	9.5	8.9	9.1	10	8.1	**8.68**
	Cut Lasagne	7.7	9.7	9.9	4.8	5	9	4	10	7.9	5.8	6.5	5	8.2	9.7	9.8	8.8	9.5	6.8	9.3	8.7	**7.81**
	Pierce potato	10	9.1	9.9	4.1	5.2	9.4	10	10	7.9	4	9.6	3.3	5.2	6.1	9.8	8.8	9.5	7.8	9	9.8	**7.92**
	Retrieve item	9.6	8.3	9.9	4.9	5.2	4.2	5.4	10	8	4	8.3	3.1	8.3	10	9.1	8.7	8.4	5.5	8.9	8	**7.4**
	Scoop sugar	9.9	5.7	8.6	7.4	9.2	8.3	8.4	10	9.5	8.6	9.9	2.8	7.7	7.2	9	6.9	5.2	7.9	8.8	10	**8.05**
	Task average	8.2	6.9	9.4	7	6.7	8.3	8.1	8.9	7.7	6.8	8.3	5.8	8	8.1	9	8.5	7.8	7.4	8.1	9.1	**7.92**

Fig. 4. Results showing the various systems' accuracies on a scale from 1 to 10. Each cell is an accuracy measured by comparison of the system's rating with the ground truth for a task/tool combination. Heavier shading indicates a less accurate result. Blanks appear for objects Kinect could not capture.

clouds with affordances of cut, sieve, poke, hook, cup, hit. We mapped our tasks to some of these as follows {Hammer nail, Tenderise meat} → hit; {Lift pancake, Cut Lasagne} → cut; {Pierce potato, Retrieve item} → poke; {Scoop sugar} → cup. Given an affordance such as 'hit', ScW 1 was able to provide a numeric similarity value for each of our 20 objects, stating how similar it was to the closest example 'hit' object from M1. We had to normalise these values before we could use them as a rating of the target object, because some target objects had (relatively) extremely low scores due to dissimilarity from M1. We took the average of all affordance ratings for each target object, minus the total set average, as a measure of 'object deviance'. We then added object deviance to that object's values (to bring them closer to the overall average). Finally we scale the values into the $[1, 10]$ range. The comparison of our system with ScW 1 and 2 is somewhat unfair because ScW is not designed for our task, and ScW 1's training set is only very suitable for a subset of our tasks (hammering, cutting, retrieving). Nevertheless it is by far the best system we could find for this type of task, and the closest competitor.

6 System ScW 2. This is Schoeler and Wörgötter's system with no training set. Instead it uses one of our best exemplars for each task, and compares that with the other 19 objects to generate ratings (guaranteeing a perfect score when comparing with itself). This cross comparison of everything we call the matrix model. Again we had to normalise values based on each target object's average (as above); mug and teapot had particularly low values before normalising. For 'lift pancake' we created an ideal tool point cloud because we felt none of the 20 were quite perfect for the task, so ScW 2 compared this perfect tool with the 20 objects in this case.

Fig. 6 shows some selected fittings from our system, illustrating the projection mechanism where sometimes parts have a function imposed on them by the source (e.g., in inversion) and sometimes the model needs to adapt to the reality of the point cloud (adapting angle, proportion, scale).

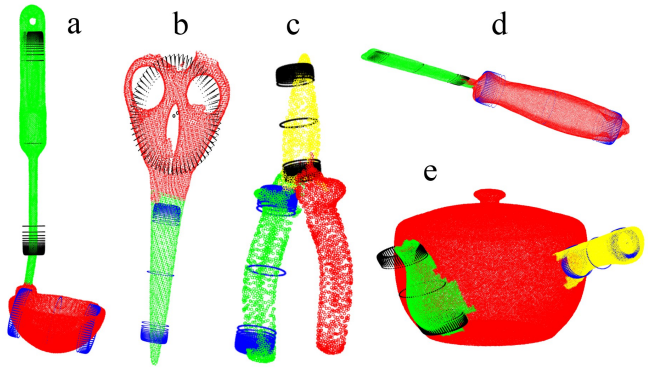

Fig. 6. Typical fits resulting from our system (with Artec scans) where the black fitted superquadric is the action part and the blue is the part to be grasped. This illustrates: (**a**) Inversion to use the long part of the spatula to retrieve from a gap. (**b**) Inversion of the scissors for hammering shows a weakness of the approach; it thinks the scissors handle is useful to hammer a nail because it does not know that the gaps cause a problem. (**c**) Fitting in an object with three segments. (**d**) Significant distortion of size to almost perfectly fit the scraper head. (**e**) Distortion of angle to try to use a teapot to pierce (this earns a low score).

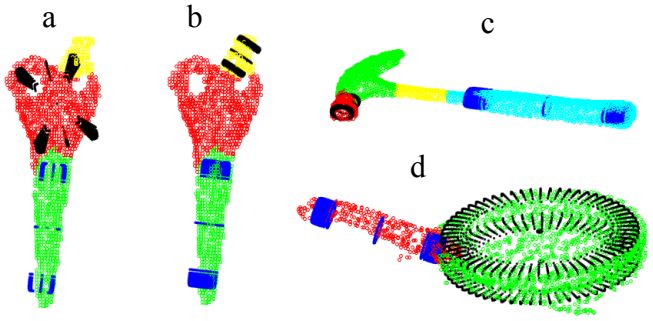

Fig. 7. Typical fits resulting from our system (with Kinect scans) where the black fitted superquadric is the action part and the blue is the part to be grasped. (**a**) Inversion in the scissors to grab it by the blade and use it to lift pancake (**b**) Inversion of the scissors for using small part to be able to scoop sugar (**c**) Fit in the hammer for hammering nail (**d**) Fit in the frying pan for tenderising meat

Figs. 5 and 4 show the full results. The running time of our system is approx. 54 seconds for one projection with three iterations, and 3 seconds to fit a single part to a segment on a modest Intel i5-3470 3.20 GHz (quad-core). Superquadric fitting is amenable to parallelisation e.g., in a GPU, which could cut the time dramatically.

A. *Discussion of Results and Future Work*

All the systems perform significantly better than random. Furthermore answering randomly and scoring 7 would not be the same as the algorithms that additionally output which part should be the handle to grasp and which part is the business end.

Kinect performs relatively poorly but still above chance on average. It is not meaningful to compare it with the other methods because many objects could not be included due to Kinect scanning troubles. In general it performs well on tasks requiring objects that are large, and very poorly on pierce, retrieve and scoop, which require small object parts that it cannot see.

ScW 1 performs very well on the hammering and cutting tasks where it has a suitable dataset of similar objects. It is not surprising that ScW 1 performs poorly on 'Lift pancake' and 'Pierce potato' because it has no suitable items in its training set. It performs poorly on 'Scoop sugar' because it does not consider scale, which is critical. At a deeper level ScW's similarity notion is not affected by the task under consideration, whereas humans do consider the target task when judging tool similarity, and in this way our approach may be more human-like. A further problem for both ScW systems is that they do not consider importance of parts for a task, so e.g. they consider a good similarity in the handle as equally important to similarity in the head.

It is curious that ScW 2 performs so well given that it has no training. This may be because our 20 objects contain exemplars that are very good and very bad for every task, and similarity to the best exemplar is a very good method of rating, also the presence of poor tools helps to calibrate the lower end of the scale appropriately. ScW 2 performs astonishingly well on the lifting task, probably because we gave it a hand-crafted ideal tool to compare with. Overall the ScW system is remarkably good at a task it was not designed for (and for this reason our comparison is somewhat unfair).

V. FUTURE WORK AND CONCLUSIONS

We now consider some future improvements that could be made. Our task model rules are crude in that they apply a quadratic, or linear or flat function to parameters. A better approach would be to use machine learning from a good dataset to do fine-grained tweaking of weights on function parameters. This means learning in what ways a component of the source model can be allowed to morph its shape (or change scale) while still being effective. Learning could also be used for a more ambitious extension. Currently our source models are hand-coded. We see this as an important first step. It allows us to test our hypothesis about what is a good representation for a source model. Having completed this we can tackle learning of a source model, given a set of geometric primitives to be used in the modelling, and a training set of objects for each function.

Given the success of the ScW system, especially on some tasks we could use a mixture of experts approach to use the technique that gives the best result for the task at hand (similar to [19]). Looking at a 'best of' just across our task averages could give an average of 8.21. There is also possibility for a closer integration, for example incorporating shape signatures in our projection approach to have extra bottom-up information to guide the initial orientation and suggested distortion of the model.

This paper's idea of projecting a source model of a tool into target point clouds is probably only part of the story of how humans do the task. Simulation is another important part. It emerged from human subjects' commentary of the task that they do simulate in some detail: grasping the tool, and approaching the target object, with a table surface present as well, and playing through the motions in performing the task. We suspect that humans do a two

step process in selecting a substitute tool: first they scan the environment looking at a large number of present potential tools and rapidly assessing them based on their parts and the relationship among them (this process may be similar to what our system does); secondly, having identified a potentially useful tool they engage in a more time-consuming simulation process to get a more accurate assessment of the tool's potential for the task.

In conclusion, this work has made an advance in how effectively an artificial system can recognise tool objects that have the potential to be used as tools in creative ways. The system can return a suggestion for (i) which tool to use, (ii) which is the grasping part, and how its axis is aligned, (iii) which is the head and in which orientation it should be brought to the target. We see this as one example use of a generic analogical machinery that is used in many aspects of cognition. Our larger enterprise is to bridge the gap between sensorimotor robotics and higher levels of cognition traditionally under the umbrella of symbolic AI. Ultimately this should address what Barsalou advocated with his 'perceptual symbols' [20]. We intend to further develop our current work on projecting tool concepts to objects in sensorimotor data, especially to include a simulation ability to try out candidate tools. Subsequently we intend to tackle robot planning steps: to look at a visual scene, where a robot has a task to perform (a step in a plan) and must select a skill and associated representation to impose on the situation. The robot will have a library of manipulation skills and associated idealised scenes (tool, object to work on, work surface), and will select from this library a suitable skill and perspective to apply in the current situation. This entails a projection that can distort the robot's perception and lead it to see the real-world scene in a way that makes a particular manipulation straightforward, by transferring existing knowledge. For example the robot could view grains or seeds as liquid and borrow schemas for pouring and scooping liquids, or the robot could view a pancake as a flat rigid cylinder, or as a foldable cloth, depending on the task requirements. In the longer term we believe that the same analogical matching should be valuable in language processing, for example to understand metaphors, such as "the speaker was meandering", or to automatically generate news stories where it is desired that a certain perspective is imposed. This would be important for robots communicating naturally with humans.

ACKNOWLEDGMENT

Paulo Abelha is supported by the Brazilian agency CAPES through the program Science without Borders. Thanks to the University of Aberdeen's ABVenture Zone for equipment use. We got a lot of very helpful advice and assistance from the following people: Severin Fichtl, Dirk Kraft, Norbert Krüger, Karthik Varadarajan, Markus Vincze, Florentin Wörgötter.

REFERENCES

[1] B. Indurkhya, *Metaphor and Cognition*. Dordrecht, The Netherlands: Kluwer Academic Publishers., 1992.

[2] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[3] E. Ugur and J. Piater, "Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, 2015, pp. 2627–2633.

[4] W. Wohlkinger, A. Aldoma, R. Rusu, and M. Vincze, "3dnet: Large-scale object class recognition from cad models," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 5384–5391.

[5] M. Schoeler and F. Worgotter, "Bootstrapping the Semantics of Tools: Affordance analysis of real world objects on a per-part basis," *IEEE Transactions on Autonomous Mental Development*, vol. pp, no. 99, pp. 1–1, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7293623

[6] 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 *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, nov 2013, pp. 5943–5949.

[7] G. Biegelbauer, M. Vincze, and W. Wohlkinger, "Model-based 3D object detection," *Machine Vision and Applications*, vol. 21, no. 4, pp. 497–516, dec 2008.

[8] J. Krivic and F. Solina, "Part-level object recognition using superquadrics," *Computer Vision and Image Understanding*, vol. 95, no. 1, pp. 105 – 126, 2004.

[9] K. Varadarajan and M. Vincze, "4D space-time mereotopogeometry-part connectivity calculus for visual object representation," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, Aug 2014, pp. 4316–4321.

[10] E. Ugur, H. Çelikkanat, E. Sahin, Y. Nagai, and E. Oztop, "Learning to grasp with parental scaffolding," in *11th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2011), Bled, Slovenia, October 26-28, 2011*, 2011, pp. 480–486.

[11] O. Kroemer, E. Ugur, E. Oztop, and J. Peters, "A kernel-based approach to direct action perception," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 2605–2610.

[12] T. Fitzgerald, K. McGreggor, B. Akgun, A. K. Goel, and A. L. Thomaz, "A visual analogy approach to source case retrieval in robot learning from observation," in *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[13] T. Fitzgerald, A. K. Goel, and A. L. Thomaz, "A similarity-based approach to skill transfer (extended abstract)," in *Women in Robotics Workshop at the Robotics: Science and Systems Conference. Rome, Italy*, 2015.

[14] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang, "Contextual Part Analogies in 3D Objects," *International Journal of Computer Vision*, vol. 89, no. 2-3, pp. 309–326, sep 2010. [Online]. Available: http://link.springer.com/10.1007/s11263-009-0279-0

[15] D. Kraft, R. Detry, N. Pugeault, E. Baseski, F. Guerin, J. Piater, and N. Kruger, "Development of object and grasping knowledge by robot exploration," *Autonomous Mental Development, IEEE Transactions on*, vol. 2, no. 4, pp. 368–383, Dec 2010.

[16] K. Welke, J. Issac, D. Schiebener, T. Asfour, and R. Dillmann, "Autonomous acquisition of visual multi-view object representations for object recognition on a humanoid robot," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 2012–2019.

[17] M. Schoeler, J. Papon, and F. Wörgötter, "Constrained planar cuts - object partitioning for point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition CVPR*, 2015.

[18] A. Jaklic, A. Leonardis, and F. Solina, "Superquadrics and their Geometric Properties," in *Segmentation and Recovery of Superquadrics*, M. A. Viergever, Ed. Dordrecht: Kluwer Academic Publishers, 2000, ch. 2, pp. 13–39. [Online]. Available: http://www.springer.com/computer/image+processing/book/978-0-7923-6601-0

[19] D. Nyga, F. Balint-Benczedi, and M. Beetz, "PR2 looking at things - ensemble learning for unstructured information processing with markov logic networks," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 3916–3923.

[20] L. Barsalou, "Perceptual symbol systems," *Behavioral and Brain Sciences*, vol. 22, pp. 577–660, 1999.