Context Dependent Action Affordances and their Execution using an Ontology of Actions and 3D Geometric Reasoning

Simon Reich[1], Mohamad Javad Aein[1], and Florentin Wörgötter[1]

Abstract— **When looking at an object humans can quickly and efficiently assess which actions are possible given the scene context. This task remains hard for machines. Here we focus on manipulation actions and in the first part of this study define an object-action linked ontology for such context dependent affordance analysis. We break down every action into three hierarchical pre-condition layers starting on top with abstract object relations (which need to be fulfilled) and in three steps arriving at the movement primitives required to execute the action. This ontology will then, in the second part of this work, be linked to actual scenes. First the system looks at the scene and for any selected object suggests some actions. One will be chosen and, we use now a simple geometrical reasoning scheme by which this action's movement primitives will be filled with the specific parameter values, which are then executed by the robot. The viability of this approach will be demonstrated by analysing several scenes and a large number of manipulations.**

I. INTRODUCTION

From every day life we know that different scenes suggest different actions, e.g. a board, a tomato, and a knife suggests a "cutting the tomato" action. However, assessing whether or not a robot could actually do this, whether it should/could do rather something else or whether not much can be done at all given such scenes remains a difficult problem. It amounts to estimating the affordance of certain actions give the context provided by the scene. One approach to solving this problem is to analyse a scene and derive from it a symbolic representation, which can then be used to find possible actions and/or to do planning.

To achieve this, in Rosman and Ramamoorthy [1] a complex network of geometrical relations in the spatial and temporal domains is used. Via Support-Vector-Machines (SVMs) topological features and symbolic meanings are learned. In Sjoo and Jensfelt [2] patterns of functional relationships are defined, e.g. the object "work surface" with the action "manipulate". Similar, in Liang et al. [3] posture templates are applied to the input data of each frame. The resulting series of templates eventually forms a library of actions. The authors use variable-length Markov models for learning.

Staying closer to the actual motion patterns one can also break down actions into segments, using – for example – principal component analysis (PCA) as in Yamane et al. [4]. A motion sequence is here projected into a state space, which is then mapped to the first n principal components. In that reduced state space a threshold is applied and the action is divided into two parts. The same is iteratively applied to each subspace until some exit criteria is met. The resulting segments could then be interpreted as meaningful action parts.

There are also non-vision based methods available, for example in Modayil et al. [5] and Liao et al. [6], but these methods will not be discussed any further, as we are focusing on vision here.

All these approaches are problematic, because it remains difficult to smoothly link sensor signals (e.g. from scene analysis) to symbolic action concepts and then back to the signal domain for creating the trajectories needed for the execution of an action by a robot. There is a danger of too strongly focusing on the symbolic side or of remaining too close to the signal domain.

Here we focus on manipulation actions and one goal of the current study is to improve on this by introducing a deeper hierarchy of several layers between signals and symbols for analysing a scene in a given action context. We ask: What is needed to push (or pick, or cut, etc.) a certain object? Which are the *general preconditions* required for this regardless of the actual objects in the scene? And – if those hold – are also the *specific conditions* met to actually do it?

Again we build on our old framework using Semantic Event Chains (SECs) Aksoy et al. [7] but now we extend them in several ways. SECs are matrices that show how touching relation between pairs of objects change in an action. The entries of the SEC matrix are ("T") for Touching, ("N") for Not touching and ("A") for absent relation. A manipulation actions is segmented at keyframes which are moments that a touching relation changes. The original SEC framework did not much care about objects. Here, based on an older study [8], we will now incorporate (still abstract) object roles to build an object-action-linked ontology of manipulations, where these object roles define the general preconditions that need to be met to perform a certain action *at all*. On top of this we introduce a simple framework for geometric reasoning allowing the machine to check specific preconditions, too, to finally execute an action.

In this study the robot selects one object in a scene and asks – like a child during play – what could I do with it? The framework will then analyse the situation and suggest possible manipulation actions, thereby addressing the problem of context dependent affordances.

The research leading to these results has received funding from the European Communitys Seventh Framework Programme FP7/2007-2013 (Programme and Theme: ICT-2011.2.1, Cognitive Systems and Robotics) under grant agreement no. 600578, ACAT.

[1]Third Institute of Physics - Biophysics, Georg-August-Universität Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany
{sreich, maein, worgott}@gwdg.de
(The first two authors contributed equally to this paper.)

II. Method

This section divides into two parts: 1) definition of the ontology and 2) algorithm to arrive at robotic execution of manipulation actions using the ontology given an observed scene. We start with the first aspect.

A. *Ontology of Manipulation Actions*

We use all manipulation actions defined in Wörgötter et al. [8] and create a new ontology by incorporating three layers: 1) abstract object relations (SEC), 2) object topologies and also 3) action primitives. Before doing this we need to define the roles of object in a more general way.

Defining Object Roles: Those are determined by the changes that occur following an action in the relation of an object to other objects. An action involves at least two objects: a *hand* and a *main* object. Resulting object categories (hand, main, primary, secondary, etc.) and their abstract roles are defined in table I.

We enumerate all possible actions; the resulting groups are summarized in table II, we have:

1) Actions with main support,
2) Actions without main support,
3) Actions with load,

and several actions usually exist for each group. A schematic of some sample actions from the three groups is shown in figure 1. Due to lack of space we show the full definition of the ontology only on our webpage[1]. Now we can define the layers of the ontology.

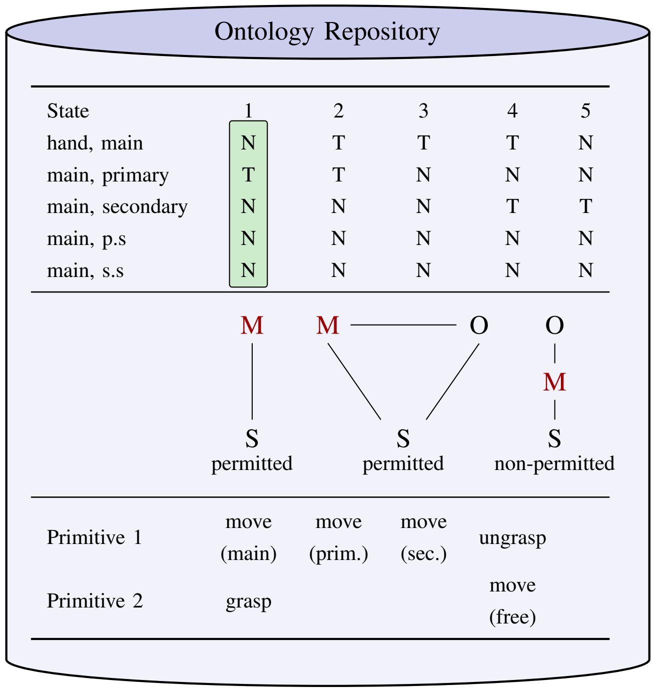
Layer 1) SEC based object relations at start: The individual graphical panels in figure 1 represent the columns of a SEC which reflect the transition of object relations and are the necessary condition for successful execution. Fig. 1b shows a *pick and place* action; its corresponding SEC is shown in figure 2. *The SEC-defined pre-conditions, to allow for an action, are encoded in the first column of a SEC.* If and only if these touching relations are not violated, the action could commence. But this is not yet sufficient.

Layer 2) Object Topologies: All actions are always performed at the main object and this will only be possible if the SEC-pre-condition hold *and if* the main object appears in the scene with certain topological connections to other objects. The middle part of figure 2 shows which topologies are permitted for *pick and place*.

Remarkably there are only three possible topological relations to which all scenes that include the main object can be reduced. To achieve this the complete connectivity graph of who-touches-whom will be reduced into those subgraphs that contain the main object. Each subgraph consists of at least the *main* object and the *support*, and, if directly touching neighbors exist, only one directly touching neighbor (figure 3). There are three cases:

1) The main object has only one touching relation. The touched object is a support, e.g. a table (see figure 3,

[1]http://www.dpi.physik.uni-goettingen.de/cns/index.php?page=ontology-of-manipulation-actions



State	1	2	3	4	5
hand, main	N	T	T	T	N
main, primary	T	T	N	N	N
main, secondary	N	N	N	T	T
main, p.s	N	N	N	N	N
main, s.s	N	N	N	N	N

Primitive 1	move (main)	move (prim.)	move (sec.)	ungrasp
Primitive 2	grasp			move (free)

Fig. 2: This figure shows one example action, pick and place, in the proposed ontology repository, which is also shown in figure 1b. It consists of three parts: the SECs (top), including the SEC pre-condition (top with green bar), topological preconditions (middle), and primitives (bottom).

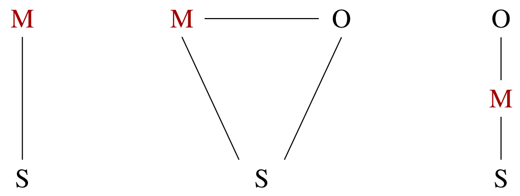


Fig. 3: All complex graph structures can be reduced to one of these three graphs. “M” is the main object; “O” depicts other objects in the scene on which we do not have any closer information. The support is “S”.

left). A real world example as shown in figure 6b is the blue plate which is on the board.

2) The main object has two touching relations. One is a support, the second one is another object, which is also touching the support (see figure 3, middle). In figure 3, the apple touches its support (green plate) and the yellow pedestal which is on the same support.
3) The main object has two touching relations. It touches its support and another object, which does not touch the support (see figure 3, right). In figure 3, the pedestal is on top the green plate and the jar is on top of the pedestal (but does not touch the green plate).

These subgraphs determine the remaining preconditions. For example, a tower structure as shown in figure 3 (right graph) is not allowed for pick and place and pushing actions.

Layer 3) Movement Primitives: SEC pre-conditions and topological pre-conditions define the first two layers of the

Object Category	Description	Relations and relation changes during the action
hand	The object that performs the action	not touching anything at the beginning and the end of action. It touches at least one object.
main	The object which is directly in contact with the hand	not touching the hand at the beginning and the end of action. It touches the hand at least once.
primary	The object from which the main object separates	initially touches the main object. Changes its relation to not-touching during the action
secondary	The object to which the main object joins	initially does not touch the main object. Changes its relation to touching during the action
load	The object which is indirectly manipulated	does not touch the hand. During the action either touches/untouches the main and untouches/touches container.
container	The object whose relation with load changes and it is not the main object	touches or untouches the load object
main support	The object on which the main object is located	touching the main object all the time
primary support	The object on which the primary object is located	touching the primary object all the time
secondary support	The object on which the secondary object is located	touching the secondary object all the time
tool	The object which is used by the hand to enhance the quality of some actions	touching the hand all the time

TABLE I: Object categories which are defined based on relation changes.

Category	Sub-Category	Example Actions
Actions with main support	Actions with hand, main and main support	push, punch, flick
	Actions with hand, main, main support and primary	push apart, cut, chop
	Actions with hand, main, main support and secondary	push together
	Actions with hand, main, main support, primary and secondary	push from a to b
Actions without main support. (These action have primary, secondary and their supports)	primary $\neq$ secondary and primary support $\neq$ secondary support	pick and place, break off
	primary $\neq$ secondary and primary support = secondary support	pick and place, break off
	primary $\neq$ secondary and primary = secondary support	put on top
	primary $\neq$ secondary and primary support = secondary	pick apart
	primary = secondary	pick and place, break off
Actions with load and container	The relation of load and main changes from N to T (loading)	Pipetting
	The relation of load and main changes from T to N (unloading)	Pour, Drop

TABLE II: Summary of ontology of actions. Actions are divided into three categories and further into sub-categories. There can be more than one action in each sub-category.

ontology. The third and last layer is a set of movement primitives, which are needed to execute the action.

For the pick and place action, the primitives are shown at the bottom of figure 2. The complete list of primitives for all actions is shown on the web page. How to fill these abstract primitives with execution relevant parameters will be described later and the process of execution of actions is then the same as in Aein et al. [9].

One primitive shall be explained in more detail: The $move(object,\ T)$ primitive sends a command to the robot to move to a pose which is determined by applying transform T to the pose of $object$. The transform T has two parts, a vector p which shows the translation, and a matrix R which shows the rotation. For example, when we want to grasp the *main* object, we perform a $move(main,\ T)$ primitive to move the robot arm end effector to a proper pose for grasping. Since we want the end effector to reach the *main* object, the vector p in this case is equal to zero. However, the rotation part R needs to be set such that the robot approaches the *main* object from a proper angle. This is necessary to avoid possible collisions with other object near the *main*.

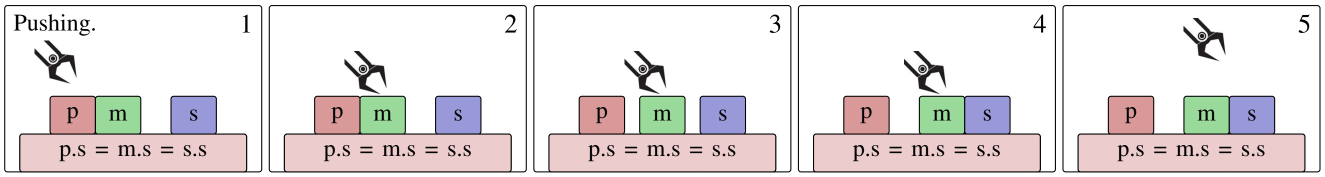
B. Algorithm for Execution-Preparation

Figure 4 shows an overview of the algorithm used for robotic execution of the above defined actions. Most components rely on existing methods and will not be described in detail.

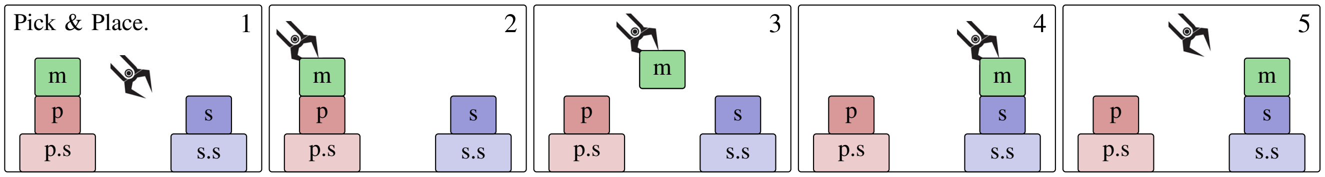
We start with (1) an RGB-D recorded scene which is (2) segmented using the LCCP algorithm [10] into different objects from which (3) a graph is created with edges between objects that touch each other. (4) Then we randomly choose one object as main. (5) The complete list of all considered manipulation actions, of which there are 29 (see Table III), is derived from Wörgötter et al. [8] (only 3 are indicated in fig. 4) and (6) for all of them we use the first layer of the ontology to check whether the main object in this scene fulfills their SEC pre-conditions. This leads to (7) a first list of possible actions and for those we check (8) with the second layer of the ontology the topological pre-conditions by which the list gets reduced. Now we can (9) use the third layer and extract from the ontology the required action primitives. This concludes the preparation stage and this information is sent to the execution engine.

C. Execution-Parameterization: Geometric Reasoning

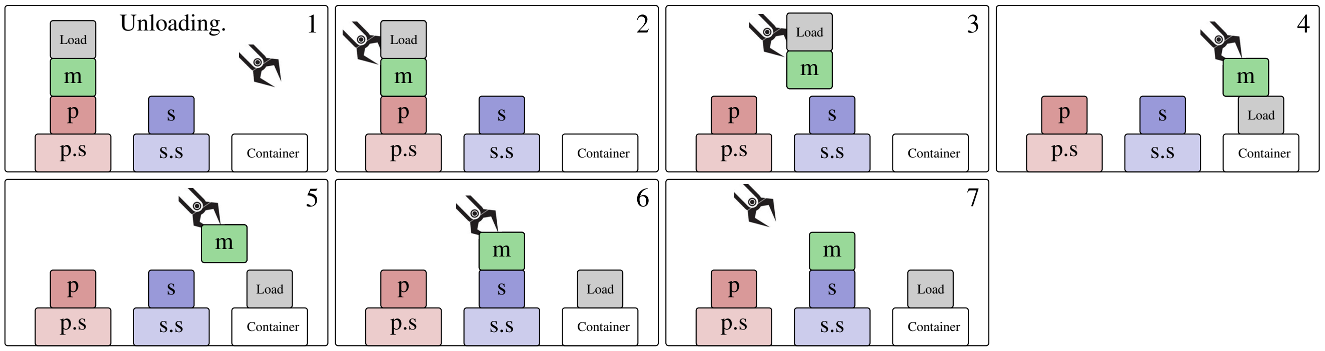
In order to execute any of the in-principle-possible actions we need to parameterize them. In general we use our action library from [9] where the required parameters are all defined. They directly map to the action primitives from stage (9) of the above described algorithm. Thus, we need to now



(a) Action 1: Pushing.



(b) Action 2: Pick and place.



(c) Action 3: Unloading.

Fig. 1: Schematic of actions in the ontology are shown for the three categories. From each category only one action is shown. The objects are marked using the following convention: h= hand, m= main m.s= main support, p= primary, p.s= primary support, s= secondary, s.s= secondary support.

consider the actual scene layout to find possible parameter ranges for these movement primitives. For this we employ geometric reasoning. The goal of this is that given an action and its main object we want to find the directions which are free to manipulate this object. These directions are directly used to define parameter ranges of the action primitives (e.g. $move(object, T)$) for action execution.

In figure 5 (top) a schematic example is shown, which consists of two cubical point clouds. The main object is shown in green and using a threshold we find all objects around it (here one). In the example the primary object is the blue cube. More objects can be taken into account by applying the algorithm iteratively; we only analyze the relative position between two objects each time.

First, we search for a local neighborhood of voxels from the main to the primary object and compute a list of distances from each point of the main object to each point of the primary. Afterwards, we create a histogram of these distances, as shown in figure 5 (top and middle). Now, we search for the maximum in the histogram and use the points in all m bins below the maximum which represent the local neighborhood. In figure 5 (middle) these points correspond to the red bars in the histogram and the red points in the two blocks. m is the only parameter of this method and depends on the point cloud quality. For smaller objects or lower

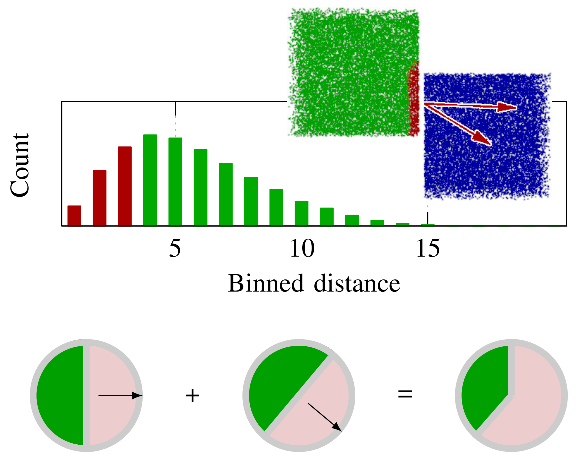


Fig. 5: Schematic example of how to perform geometric reasoning.

resolution a bigger m is needed. We have found heuristically that using all points up to the maximum usually leads to good results.

After identifying the neighboring points we compute the normals of those and cluster the normals via k-means clustering. Now, each cluster points from the main object to the primary; these clusters show the “blocked” sides and

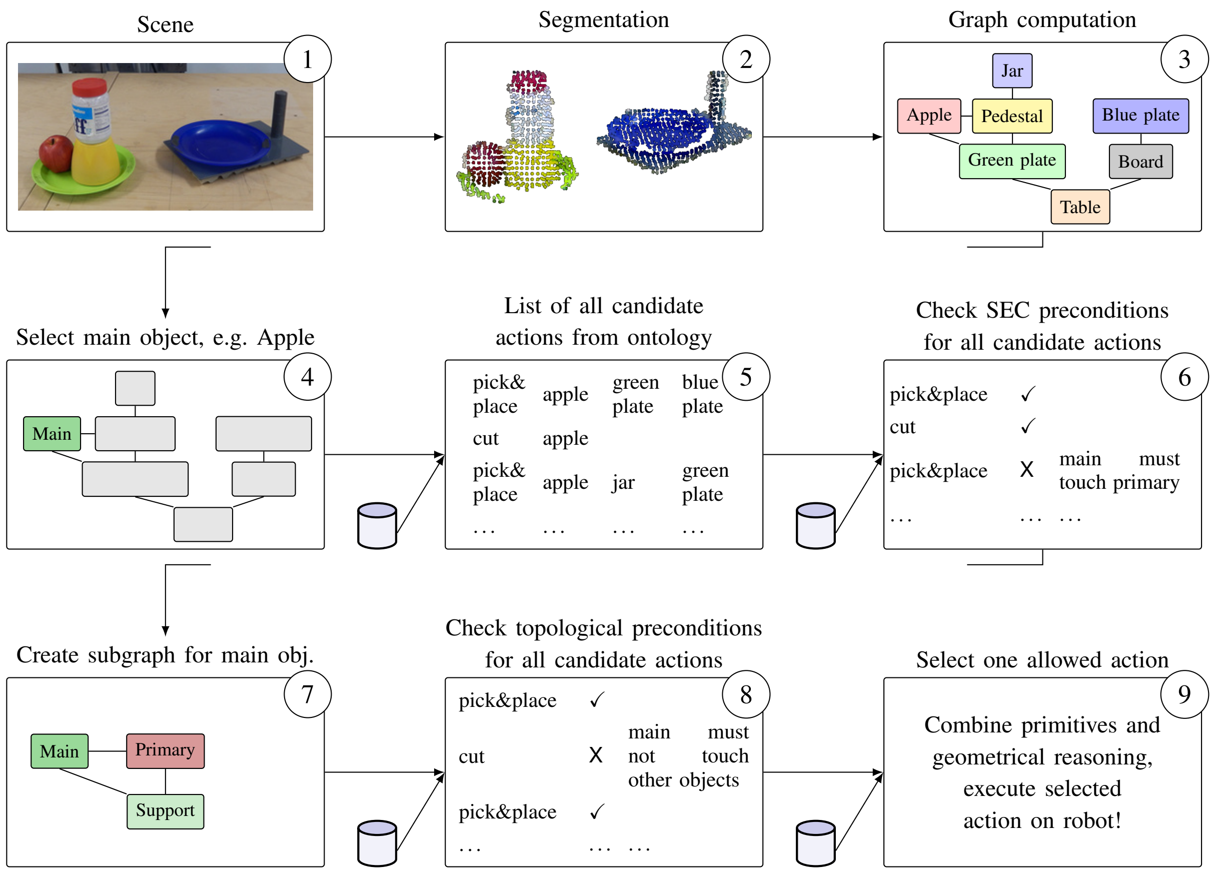

Scene
1
Segmentation
2
Graph computation
3
Jar
Apple
Pedestal
Blue plate
Green plate
Board
Table
Select main object, e.g. Apple
4
Main
List of all candidate actions from ontology
5
pick& place
apple
green plate
blue plate
cut
apple
pick& place
apple
jar
green plate
...
Check SEC preconditions for all candidate actions
6
pick&place ✓
cut ✓
pick&place X main must touch primary
...
Create subgraph for main obj.
7
Main
Primary
Support
Check topological preconditions for all candidate actions
8
pick&place ✓
cut X main not must touch other objects
pick&place ✓
...
Select one allowed action
9
Combine primitives and geometrical reasoning, execute selected action on robot!

Fig. 4: The steps of our proposed framework for scene affordances and execution are summarized here. Starting with a real world scene (1), we perform object segmentation (2) and graph calculation (3). Afterwards, a list of candidate actions is produced (5). The possibility of performing these actions is investigated in two steps by using the preconditions inside the ontology (6-8). We get the primitives from the ontology and send them to the execution engine. In case of $move(object)$ primitives, we perform the proposed geometric reasoning to get the parameters.

if inverting this we get the "free" directions. This means each cluster divides the space with into one "free" and one "blocked" part. Adding up all "blocked" parts we can compute the "free" access angles. An example is shown in figure 5 at the bottom.

K-means clustering needs the number of clusters ("k") as input parameter. As underfitting is harmful to our algorithm, but overfitting is not, we set k to rather higher values (usually $k = 8$ yields good results in real world examples).

The results of this type of reasoning on real scenes will be shown in Section III.

III. EXPERIMENTS

A. Setup and Experiments

We tested the algorithm in a ROS based system [11]. For vision sensors a Microsoft Kinect for 3D data and a high resolution Nikon DSLR camera are used. We used [12] for object recognition and pose estimation. For model tracking [13] is used. Our robot is a Kuka LWR arm which executes actions as described in Aein et al. [9]. Figure 6 shows three scenes are used for testing:

1) A cup is next to a box and a cup is on top of a pedestal.
2) The scene that we used in previous sections.
3) A cluttered kitchen scene with many objects.

A movie showing the execution of selected actions can be found in the supplementary material.

B. Results

Using these scenes, we analyse first the effect of the top two layers of the ontology asking: Given a main object, which actions are in principle permitted. Next, we will consider the third ontology layer and perform geometric reasoning on some examples to show how actual action parameterization can be performed and finally we will perform some actions with the robot.

1) Action Affordances: The results of action affordances for the three scenes are calculated by using the preconditions in the ontology and analysis of subgraph structures. The results are summarized in Table III. Each column shows the possibility of performing different actions in the ontology for a specific selection of *main*, *primary* and *secondary* objects.

Here, we can see some limitations of the SEC domain. Some actions require additional high level object knowledge (e.g. stirring or levering) and are marked with "n"; for example stirring is always denied as it requires a liquid and a container shaped object (non-permanent objects pose a big

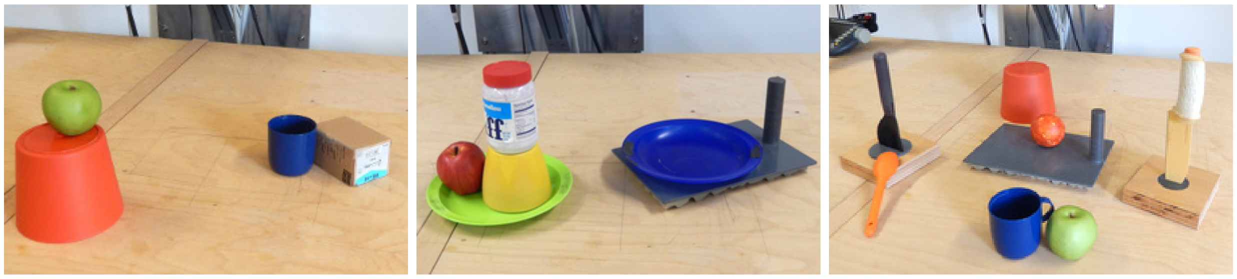

(a) Scene 1. (b) Scene 2. (c) Scene 3.

Fig. 6: These three scenes are used to test the algorithms.

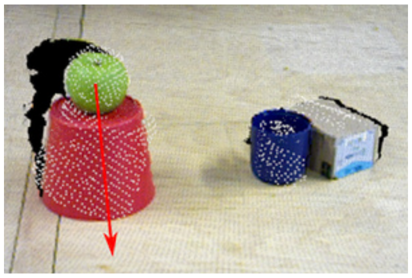


Fig. 7: Qualitative results for the geometrical reasoning method. The algorithm is applied to the object pair apple and red pedestal. For graphical purposes only the largest cluster is shown with a red arrow.

problem for SECs or planning in general). These properties cannot be measured in the SECs domain. One could argue that also cutting, kneading, or scooping needs additional high level object knowledge, but on the touching relations level these preconditions can be ensured.

2) 3D Geometrical Reasoning: Qualitative results of geometric reasoning are shown in figures 7, 8, and 9. These results show that by processing the low level point clouds one can detect the blocked and free directions of a given object. Some limitations can be found for scene 2 in figure 8a.

We expect that we can compute the normals of the point cloud, but at corners, e.g. at the border of object point clouds, this assumption is not always met and the resulting access angles are slightly off. Another problem can be seen in scene 3. In figure 9c, the relations between the orange spoon and the black spoon in the spoon holder (black spoon and spoon holder are recognized as one object) form one unexpected cluster downwards, all others point towards the spoon. Careful examinations show that there actually are some points belonging to the spoon base below the orange spoon and that the arrow downward is justified. However, the resulting access angle is very small.

3) Action Execution: The results of action execution are presented in the video attachment of the paper. The execution of three different actions is shown: “pushing”, “pick and place”, and “put on top”.

IV. CONCLUSION

The goal of this study was to address the problem of affordances given the scene context. We specifically wanted to create a system that can look at objects in a scene and suggest actions which are very likely possible. For this we first defined a novel and hopefully quite complete ontology of manipulation actions which considers objects, too, but still from a rather abstract viewpoint. The main point here is that this allows generalizing the same action across quite different scenes. Combined with geometrical reasoning this system can analyse scenes and suggest and perform many actions.

Thus, essentially the here-suggested system acts like a multi-layered planner with several levels of pre- and post-conditions. This may indeed ease robotic planning problems by allowing the system to check all conditions in a hierarchy and to finally profit from the geometrical link to the actual scene layout.

Of course, situations may exist that cannot be correctly disentangled this way. The resulting permitted movement directions are always based on parts of the 3D space that had been derived from straight direction vectors. Hence if there is a complex shaped object that hooks-around some other object this type of geometric reasoning will fail. Also, if objects are topologically linked (physically connected) in complex ways to other objects the approach will fail. Our system does not attempt to solve all these problems. Rather, like a child after some experience, here we have arrived at a system that produces *very reasonable* suggestions about how to modify its world using different manipulations. This is the main strength of this approach. We have here a quite powerful bottom-up decision framework, which does not rely on high-level knowledge but could be extended by this (for example using learned models of some aspects of the world) without problems.

REFERENCES

[1] B. Rosman and S. Ramamoorthy, “Learning spatial relationships between objects,” *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1328–1342, 2011. [Online]. Available: http://ijr.sagepub.com/content/30/11/1328.abstract

[2] K. Sjoo and P. Jensfelt, “Learning spatial relations from functional simulation,” in *Intelligent Robots and Sys-*

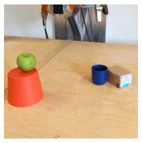
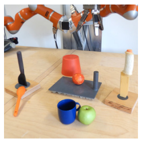
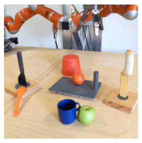
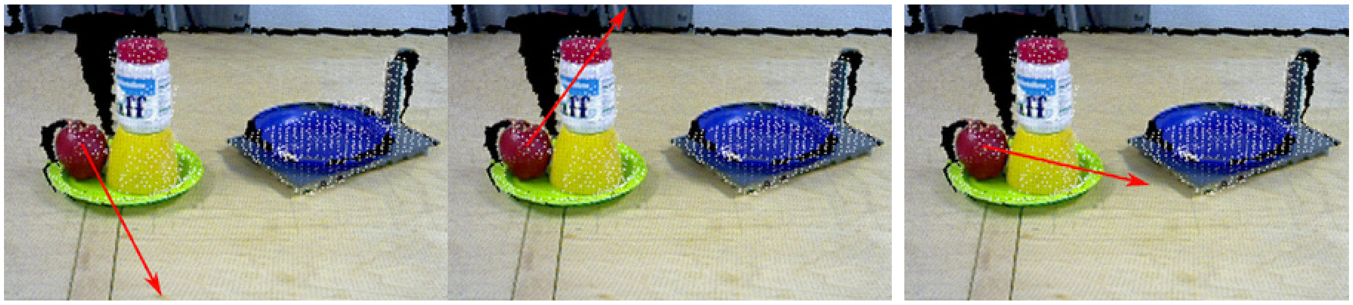
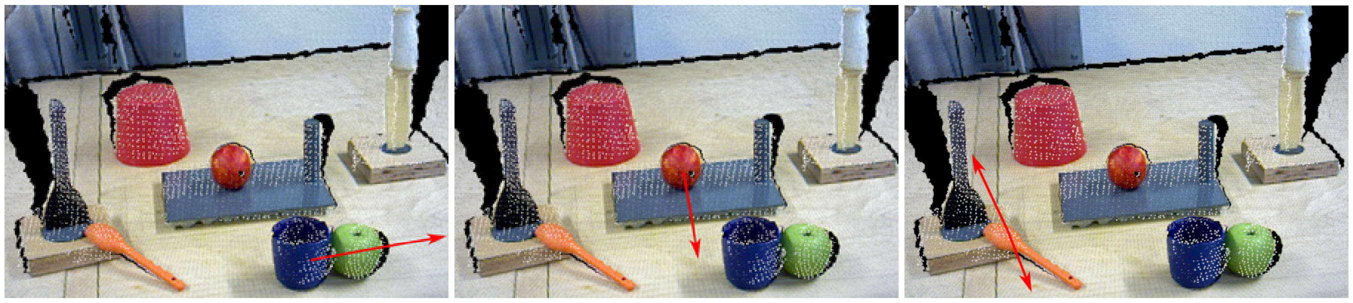
		Scene 1	Scene 2	Scene 2	Scene 3	Scene 3
						
Main Object		cup	apple	yellow pedestal	orange	apple
Primary Object		box	yellow pedestal	green plate	board	cup
Secondary Object		red pedestal	blue plate	blue plate	cup	board
1	punch	✓	✓	✓	✓	✓
2	flick	✓	✓	✓	✓	✓
3	poke	✓	✓	✓	✓	✓
4	chop	-	-	-	✓	-
5	bore	✓	✓	✓	✓	✓
6	cut	-	-	-	✓	-
7	scratch	✓	✓	✓	✓	✓
8	scissor-cut	-	-	-	✓	-
9	squash	✓	✓	-	✓	✓
10	draw	✓	✓	✓	✓	✓
11	push	✓	✓	-	✓	✓
12	stir	n	n	n	n	n
13	knead	✓	✓	-	✓	✓
14	rub	✓	✓	✓	✓	✓
15	lever	n	n	n	n	n
16	scoop	✓	✓	-	✓	✓
17	take down	-	-	-	✓	-
18	push down	-	-	-	✓	-
19	rip off	-	-	-	✓	-
20	break off	n	n	n	n	n
21	uncover by pick and place	n	n	n	n	n
22	uncover by pushing	n	n	n	n	n
23	put on top	-	✓	-	✓	-
24	push on top	-	-	-	-	-
25	put over	n	n	n	n	n
26	push over	n	n	n	n	n
27	grasp	✓	✓	-	✓	✓
28	push apart	✓	✓	-	-	✓
29	push together	-	-	-	-	-

TABLE III: Results of the action affordances for different scenes and objects. The different scenes are also depicted in figure 6. Objects corresponding to the computed affordances are listed below the table heading. Please note that we cannot check the preconditions for some actions, e.g. stirring, knead which are related to the material of objects. These actions are denoted with "n"; they require high level object knowledge. For example you need a liquid and a container object for stirring. This knowledge is not provided in the SEC domain.



(a) Scene 2: Apple and green plate. (b) Scene 2: Apple and jar. (c) Scene 2: Apple and yellow pedestal.

Fig. 8: Qualitative results for the geometrical reasoning method. For graphical purposes only the largest cluster is shown with a red arrow.



(a) Scene 3: Blue cup and apple.
(b) Scene 3: Orange and board.
(c) Scene 3: Orange spoon and black spoon.

Fig. 9: Qualitative results for the geometrical reasoning method for a cluttered scene. For graphical purposes only the largest cluster is shown with a red arrow. Please note the two red arrows in (c). Here, the two largest clusters were depicted.

tems (IROS), 2011 IEEE/RSJ International Conference on, Sept 2011, pp. 1513–1519.

[3] Y.-M. Liang, S.-W. Shih, S.-W. Shih, H.-Y. Liao, and C.-C. Lin, “Learning atomic human actions using variable-length markov models,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, no. 1, pp. 268–280, Feb 2009.

[4] K. Yamane, Y. Yamaguchi, and Y. Nakamura, “Human motion database with a binary tree and node transition graphs,” *Autonomous Robots*, vol. 30, no. 1, pp. 87–98, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10514-010-9206-z

[5] J. Modayil, T. Bai, and H. Kautz, “Improving the recognition of interleaved activities,” in *In submission*, 2008.

[6] L. Liao, D. Fox, and H. Kautz, “Location-based activity recognition using relational markov networks,” in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, ser. IJCAI’05. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005, pp. 773–778. [Online]. Available: http://dl.acm.org/citation.cfm?id=1642293.1642417

[7] E. E. Aksoy, A. Abramov, J. Dörr, N. Kejun, B. Dellen, and F. Wörgötter, “Learning the semantics of object-action relations by observation,” *The International Journal of Robotics Research September*, vol. 30, pp. 1229–1249, 2011.

[8] F. Wörgötter, E. E. Aksoy, N. Krüger, J. Piater, A. Ude, and M. Tamosiunaite, “A simple ontology of manipulation actions based on hand-object relations,” *IEEE Transactions on Autonomous Mental Development*, 2012.

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

[10] S. C. Stein, M. Schoeler, J. Papon, and F. Worgotter, “Object partitioning using local convexity,” in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014, pp. 304–311.

[11] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.

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

[13] J. Papon, T. Kulvicius, E. E. Aksoy, and F. Wörgötter, “Point cloud video object segmentation using a persistent supervoxel world-model,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, November 3-8 2013, pp. 3712–3718.