SEVENTH FRAMEWORK PROGRAMME

Project Acronym:	Xperience
Project Type:	IP
Project Title:	Robots Bootstrapped through Learning from Experience
Contract Number:	270273
Starting Date:	01-01-2011
Ending Date:	31-12-2015

XPERIENCE.ORG

Deliverable Number:	D2.1.3
Deliverable Title:	Transfer of Sensorimotor Experience: Technical report or scientific publication on how to implement the developed representations of sensorimotor experience and use them within the architecture and in the final demonstration.
Type (Internal, Restricted, Public):	PU
Authors:	F. Wörgötter, T. Asfour, E. E. Aksoy, D. Schiebener, D. Kraft, N. Krüger, T. B. Jørgensen, W. Mustafa
Contributing Partners:	KIT, UGOE, SDU

Contractual Date of Delivery to the EC:	31-01-2015
Actual Date of Delivery to the EC:	31-01-2015

Contents

Chapter 1

Executive Summary

This deliverable describes the work performed in WP2.1 in year 4. In particular, it mainly addresses the technical integration of the developed methods that extract sensorimotor experiences for the final demonstration. In addition, the deliverable also includes scientific results on how to implement the separately implemented representations of sensorimotor experience.

We will start with scientific contributions from individual partners on deriving various sensorimotor experiences. Figure 1.1 shows the block diagram of the currently developed system architecture which is triggered with the visual data captured either by stereo cameras or the Kinect device. Blue, red, and green colors in the diagram indicate the contribution from the partners KIT, UGOE, and SDU, respectively. The dashed arrow highlights the data flow between currently integrated modules within the architecture. Below each module in Figure 1.1 we also indicate the corresponding task numbers as mentioned in WP2.1.

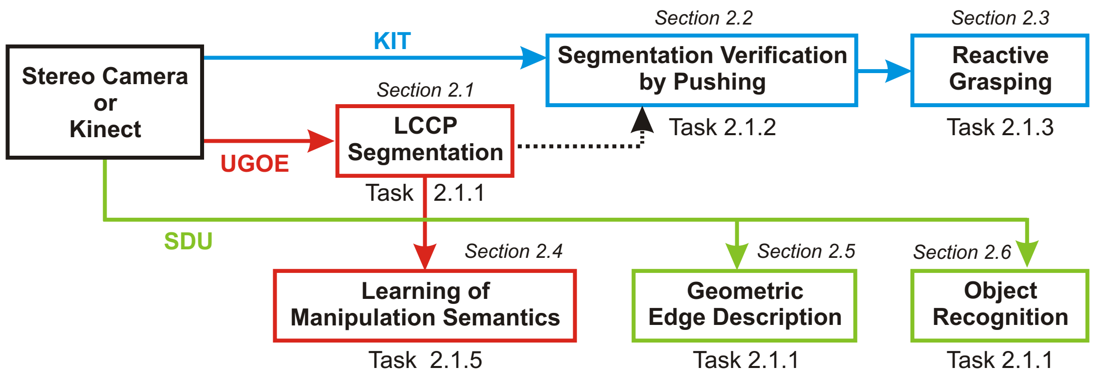


Figure 1.1: Block diagram of the currently developed modules within the system architecture.

The deliverable is organized as follows. Section 2.1 (see Fig. 1.1) introduces the recently implemented object segmentation approach based on local convexity criterion. This method employs only the depth information of the scene and leads to the decomposition of objects into various parts. In section 2.2 we continue with the verification of the primarily estimated scene segments by applying pushing actions. Such verified, but still undefined object segments are then employed for the reactive grasping task as addressed in section 2.3. In section 2.4 we introduce our incremental learning framework that extracts the semantics of observed actions from different human demonstrations.

The following sections (2.5 and 2.6) describe scientific works of a general nature that will, if at all, only be integrated in the final demo as components to enable core research. In section 2.5 we describe a new approach to extract edges from 3D point clouds as a means to faster matching for applications such as pose estimation, while section 2.6 details a system for object recognition in a robotics setup with multiple stereo or kinect cameras.

In this deliverable we aim at designing a unified structure within the software framework ArmarX that feeds the outcome of the part-based segmentation unit (section 2.1) into the segmentation verification module (section 2.2) in order to increase the efficiency of the segmentation and the subsequent reactive grasping. This process is indicated by the dashed line in Fig. 1.1. The data transferred over the dashed line is the labeled point cloud gathered from the visual sensory input. Data transfer between other modules is still subject to research.

Chapter 2

Content of the Deliverable

We here briefly present each module shown in Fig. 1.1.

2.1 Locally convex connected patches (LCCP) segmentation

Perception of objects in the scene as well as decomposition of objects into unique parts are nontrivial topics in vision-based robotic applications, which usually require high level object knowledge and limits application to known objects. We here address the problem of object segmentation by employing a bottom-up concave-convex criterion on point cloud data without employing any model fitting or learning techniques. Using a novel convexity criteria, our approach separates connected convex surface patches by concave boundaries, which leads to remarkably accurate scene segmentation. This work was published in the attached paper [SWS$^+$14].

Figure 2.1 illustrates the main algorithmic steps in our proposed segmentation method. Our method starts with capturing 3D point cloud data of the scene, for example the wooden cubes shown in Figure 2.1 A. Psycho-physical studies revealed that a human observer would consider the cubes in the upper row to form one object, however, the same cubes assembled differently in the bottom row will be perceived as two objects. To imitate the human perception we represent the scene by a graph, containing nodes and edges which represent surface patches and their neighborhood relations. We use the supervoxel algorithm by [9] to generate the surface patches. Supervoxels (see Figure 2.1 B) respect object boundaries. Their edge relation can thus be classified as being convex or concave using the convexity and sanity criteria (depicted in Figure 2.1 C-D). The sanity criterion is used to invalidate convex edges where patches are only connected by a singular point. In Figure 2.1 E black lines denote convex and red lines concave connections after considering both criteria. Our segmentation method then continues as a region growing

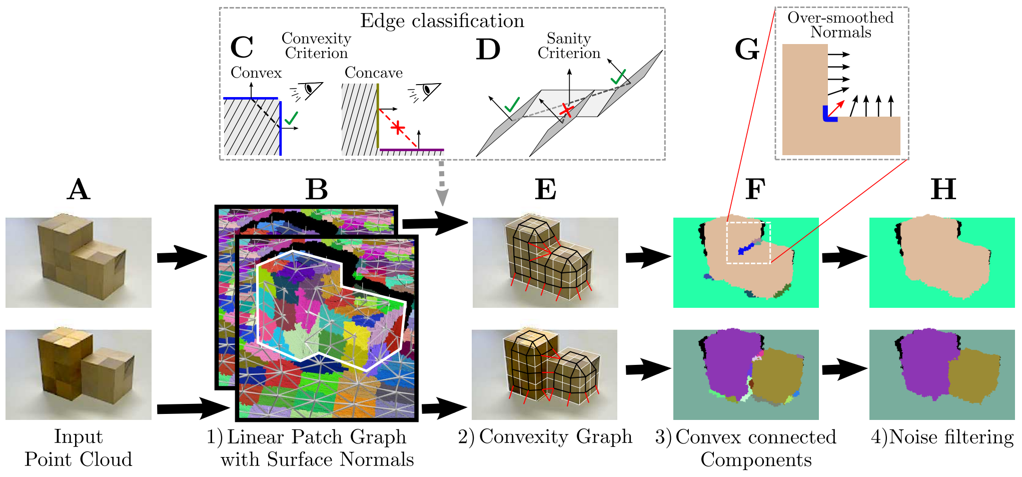


Figure 2.1: Flow diagram of the segmentation algorithm.

process in which randomly chosen seed patches propagate their label to convex connected patches (see Figure 2.1 F). New seeds are generated and propagated until all patches have been assigned a label. We finally apply noise filtering to remove small segments at the edges as highlighted in Figure 2.1 G. The final segmentation result after noise filtering is shown in Figure 2.1 H. We observe two segments for the object in the bottom row, whereas only one segment in the upper object. For more details see the attached paper [SWS+14].

2.2 Object Segmentation and Verification by Physical Interaction

As already described in D2.1.2, we have improved our approach for using physical interaction of a robot with its environment in order to discover and segment unknown objects in cluttered scenes. This method (see attached paper [SUA14]) can be applied to all kinds of rigid objects, independent of their shape or the appearance of their surface (textured, single/multicolored, ...).

As the next step, we plan to combine this approach with the LCCP segmentation presented in section 2.1. To this end, we have already integrated the LCCP algorithm into the ArmarX framework. In the next year, we intend to leverage it for the creation of object hypotheses that can then be verified by pushing. Refining the object segmentation based on their motion on a supervoxel level in addition to the RGBD-point level as it is done in the current implementation might also lead to improved segmentation quality.

2.3 Visual Collision Detection for Corrective Reactions during Grasping

We continued our line of research on reactive grasping. While this is traditionally based on the input from force or tactile sensors, those are sometimes not sensitive enough when dealing with e.g. rather light objects. If an object that the robot tries to grasp can easily be pushed away, tactile and force-based sensors are often unable to provide reliable contact detections.

To overcome this deficiency, we developed a vision-based collision detection method. It is designed for the cases in which an object does not resist to the contact with the robot's hand and thus doesn't provide significant force feedback, but starts moving instead. This motion is detected visually and used as an indicator for the collision event.

We detect the motion of the object caused by the robot's hand based on the optical flow observed during the grasp execution. The optical flow gives, for each pixel of the robot's camera image, a 2D motion vector from one image to the following. This work is to some extent inspired by [6], where the contact between a robot and an object is observed from a static external camera, and in the moment when the object starts to move, the area of nonzero optical flow caused by arm and hand spreads instantaneously to a larger image area.

In the context of robotic grasping, the situation is a bit more difficult, as the camera is mounted in the robot's head and thus itself moving during the grasping process, and in consequence there is usually a nonzero optical flow throughout the whole image all the time. What we therefore need to detect within the optical flow image is whether there is a region next to the hand, in the direction in which it is moving, where the optical flow is different from the rest of the scene (except maybe hand and arm).

To this end, we cluster the optical flow vectors by their 2D values, i.e. we get clusters of relatively similar optical flow. We track the robot hand in the camera image using forward kinematics and a particle filter that also localizes the exact pose of the fingertips which will be relevant later. We define a region next to the hand in the direction in which it moves, which corresponds in size to the object. If there is a cluster of optical flow that exists only within that region but not outside of it (with the exception of the area covered by arm and hand), this means that something there is moving relative to the scene, which strongly indicates that the hand has caused the object to move. Figure 2.2 shows the camera images of the robot, the optical flow and the clusters of similar optical flow just before and after a collision.

When a collision has been detected, this means that the current grasp attempt is probably going to fail

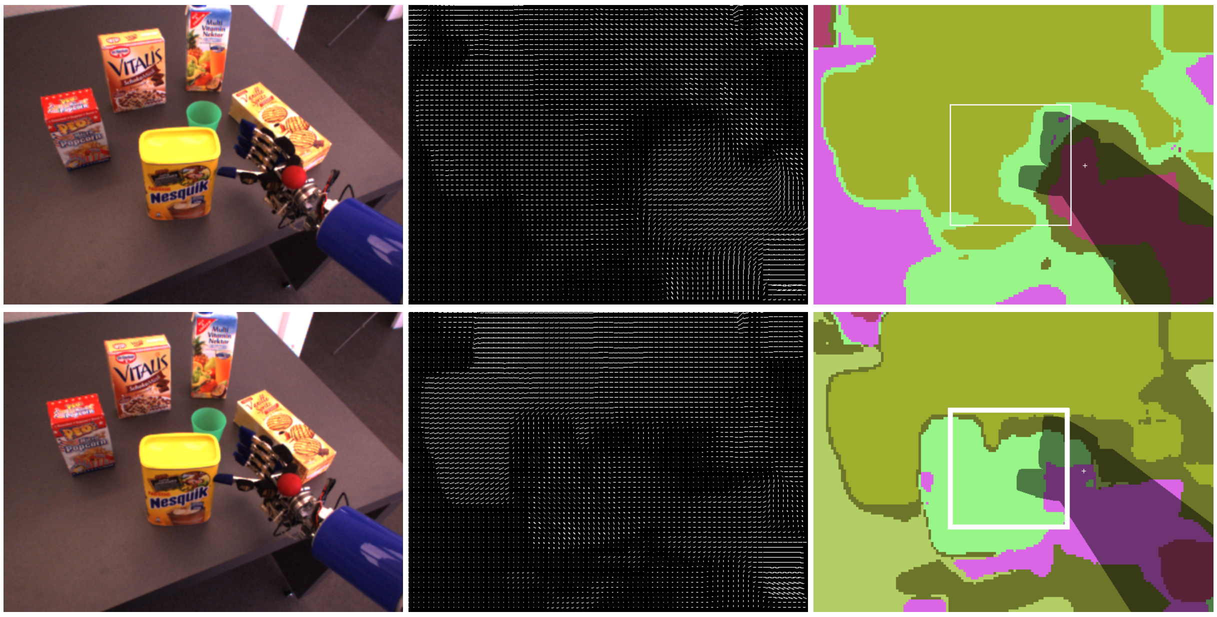


Figure 2.2: Visual collision detection in the moment when the robot's hand touches the object: The left column shows the scene from the robot's cameras immediately before and after the collision. The central column visualizes the optical flow, the right column the clusters of similar optical flow, where each cluster has been marked with a distinct color. The darker area is occupied by hand and arm and therefore ignored. The white box marks the area next to the hand where we expect a possible collision to occur. If we find a cluster of optical flow that exists mostly within this area but not outside of it, this observation indicates that the hand collided with an object and caused it to move.

and a corrective reaction is necessary. As premature collisions between hand and object are virtually always caused by one of the fingertips, we use their positions that are determined by the particle filter to decide which one is closest to the object. Based on that knowledge, we implemented three different reaction strategies. In all cases, the hand is moved a little bit back along its trajectory. The first strategy modifies the hand position by moving it into the direction from the hand center to the finger that caused the collision, projected into the plane that is perpendicular to the hand's approach vector. Thus, the fingertip which collided with the object is moved away from it. The second strategy aims at a similar effect by rotating the hand around the axis that is perpendicular to the approach vector and the direction from the palm to the colliding fingertip. The third strategy is a weighted combination of the first two correction movements. In all cases, the respective modification of the hand pose is performed and also applied to the intended grasp pose, and another grasping attempt is executed.

If another collision occurs, this can be repeated as often as necessary until the intended grasp pose has been reached without a collision. We tested the approach on grasping of known objects. Figure 2.3 shows, for the different strategies, how many corrections were needed until grasp success. For the tests, we used a set of known objects with predefined grasp configurations that all failed at the first approach. The graph shows how many of the objects were grasped after at most 1, 2, etc. reactive correction movements, depending on the strategy. As a baseline, we also implemented an uninformed strategy that just applied a small random rotation to the hand pose. As can be seen, although the random modification of the grasp pose sometimes leads to a success, it mostly still fails, while the three strategies that take into account which fingertip caused the collision are usually able to successfully complete the grasp after few correction movements. The two strategies that involve a rotation of the hand are significantly more effective than the one which modifies only the hand position.

So far, we have only tested our approach for visual collision detection in the context of corrective movements for reactive grasping of known objects. We intend to test it in combination with tactile and force-based collision detection for reactive grasping of unknown objects that were just segmented using the approach presented in the previous section.

For more details see the papers [10] and [SVA14].

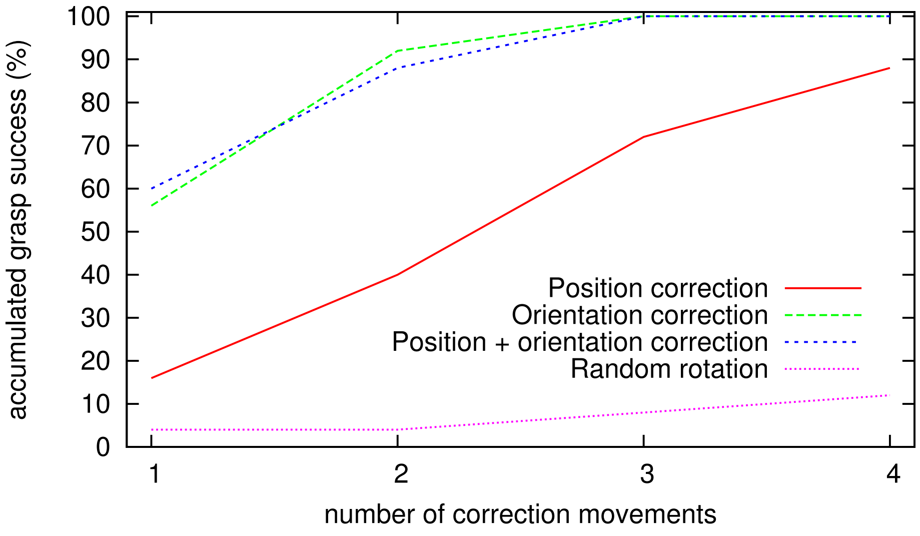


Figure 2.3: Percentage of successful grasps after a certain number of correction movements, depending on the applied strategy. E.g., for the orientation correction strategy, in 56% of the cases one corrective motion was enough, another 36% of the grasps were successful after two corrections, and the remaining 8% of the attempts required three corrective movements.

2.4 Incremental learning of manipulation semantics from human observation

Defining a generic action representation to learn the variations in trajectory, pose, and object phases is of great interest in cognitive robotics. We have recently introduced the concept of Semantic Event Chains (SECs) [1] as a novel method to encode the semantics (*meaning*) of manipulation actions (e.g. *Cutting*) independent from all variations in trajectory, pose, and object domains. SECs are derived on-the-fly from the graph representations of the consistently tracked object segments in the scene.

In this work, we use the concept of SECs as the main processing tool in order to learn the semantics of human demonstrated manipulation actions and to generate a vocabulary of such observed manipulations like *Cutting* or *Stirring*. We here aim at designing a cognitive agent that can infer and learn frequently observed spatiotemporal changes embedded in SECs in an unsupervised manner whenever a new manipulation instance is observed. This work was published in the attached paper [ATW14].

Figure 2.4 illustrates the on-line unsupervised learning framework which is triggered whenever a new manipulation sample is observed. At start, an individual manipulation is shown and the first extracted SEC sample is assumed to be the first “model” and stored in a “library”. We then encode the manipulation that follows again by a SEC. Next, we compare it with all existing SEC models in the library. For this purpose, the framework measures semantic similarities (δ) between the new SEC sample and the existing models by employing the method described in [1], which compares rows and columns of two SECs using sub-string search and counting algorithms. Computed semantic similarity values between all existing models and the new sample are stored in a matrix, called the similarity matrix (ζ_{sim}), which is then converted into a histogram ($\mathcal{H}$) representing the distribution of similarities. We apply the conventional

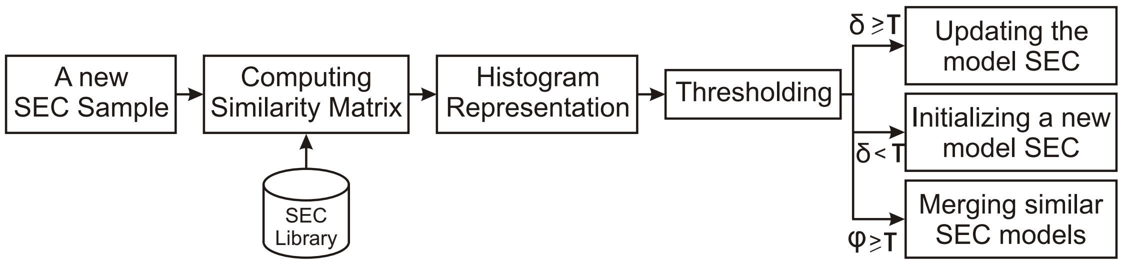


Figure 2.4: Overview of the proposed on-line learning framework.

Otsu's method [8] to the normalized histogram to distinguish low from high similarities. We take the average of the high similarities to estimate a threshold τ to classify the currently observed SEC sample against the existing models.

If similarity (δ) is higher than τ, then the new sample will be assigned to the best fitting (highest similar) model and this model will be updated with additional rows or columns that might exist in the new SEC sample. In this way, the model SECs will only consist of those rows and columns observed frequently in all type-similar manipulations. If similarity (δ) is lower than τ, the novel SEC sample will be used as a new model. In addition, we merge learned SEC models, which have higher semantic similarities (φ) than τ, as they are likely representing the same manipulation.

We benchmarked the proposed framework with a large action dataset with eight manipulations: *cutting*, *chopping*, *stirring*, *pushing*, *hiding*, *putting*, *taking*, and *uncovering*. Each manipulation consists of 15 versions demonstrated by 5 different individuals and have vast variations in terms of manipulated objects, their poses, and followed trajectories. We let the learning framework run only once through 120 manipulations by choosing a random sample at each time. To investigate the robustness of the framework, we repeated the same learning experiment 100 times independently from each other and computed differences between the incrementally learned SEC models. Among eight manipulation types in the dataset, the proposed learning algorithm extracted seven SEC models in all 100 trials by naturally merging the cutting and chopping manipulations due to high semantic similarity between each task. The threshold value τ always converged to 72% in all these 100 independent trials. These results highlight the robustness of our proposed learning framework. For more details see the attached paper [ATW14].

2.5 Geometric Edge Description and Classification in Point Cloud Data

With the recent advent of new sensor technology point clouds have become a very important representation in computer vision and especially robotics. In [JBK15] our motivation was to speed up and improve high level 3D point cloud analysis by using edges. The hypothesis was that using 3D edges to represent objects should produce a sparser and yet fairly descriptive representation, compared to using the entire surface. This general idea has shown several promising results particularly for 2D images, but also for 3D point clouds. Some applications where edges can improve algorithm speed are pose estimation and point cloud registration [2, 3]. Besides reducing run times, geometric edges can be used to improve point cloud enrichment by preserving edges.

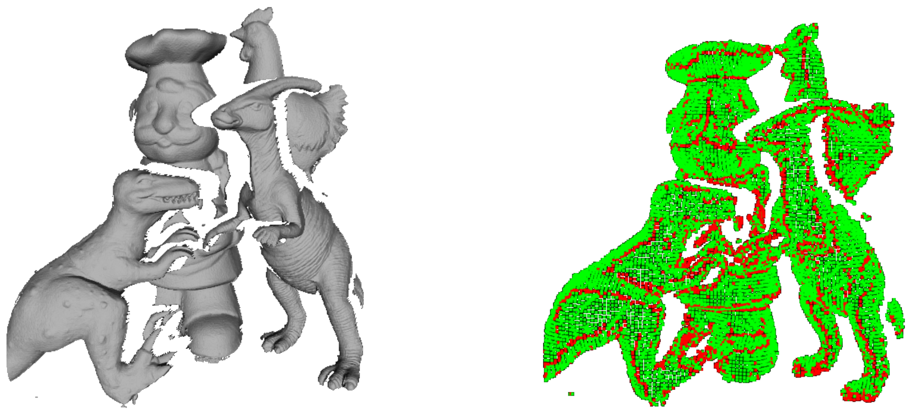
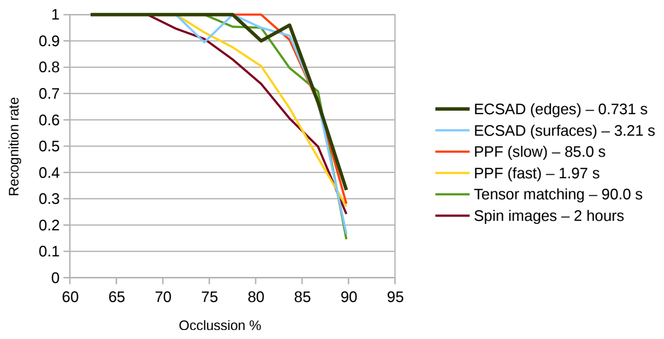


Figure 2.5: Left: A scene captured with a laser scanner [7]. Right: Edge detector response using our method (red means high confidence).

In order to investigate the benefits of edges, our first step was to develop a point cloud based edge detector. Unfortunately only a few edge detectors can be applied directly to 3D point cloud data. Therefore a new approach was developed, based on a new descriptor, machine learning and non-maximum suppression to determine sparse edges. The reason for using supervised machine learning to generate the edge models, is that it is difficult to manually define a robust edge in 3D point cloud data. Therefore we labeled a dataset with edges, and used this to generate an edge model based on random forests. Descriptors were used to generate input data for the random forest, since these produce a more structured representation compared to the raw point cloud data. In order to improve the performance of the classification scheme, a new descriptor focusing on edge description was developed. This descriptor has the added benefit of estimating the direction of the edges, which proved beneficial during the developed edge thinning scheme. After all edges in a point cloud are classified, edge response smoothing and non-maximum edge suppression is used in order to generate thinner and thereby sparser edge representations. Figure 2.5 shows the edge response of a point cloud using our method.

This edge detection approach was applied to a pose estimation problem, in order to show the benefit of using edge based descriptions compared to full surface descriptions. This showed that edges can speed up pose estimation by up to a factor of four. The result of our pose estimation is seen in Figure 2.6, where it is compared with several other methods (PPF [4], tensor matching [7] and spin images [5]). The proposed approach (ECSAD), in a surface and edge version, shows the benefits of using edges to reduce run times of algorithms. Both methods perform well in terms of occlusion vs. recognition rates, but the edge based method is substantially faster.



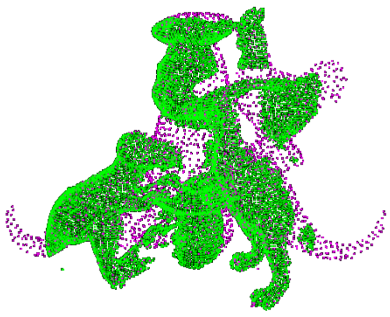


Figure 2.6: Comparison of our surface- and edge-based recognition systems with other works. Left: Pose estimation performances in terms of occlusion-recognition rates along with average running times per object. Right: Sample object alignment results. The magenta objects are the determined poses while the green points represent the scene. In this scene all objects have been correctly recognized.

2.6 Multi-View Object Instance Recognition

In [MPBK15] we introduce a fast object recognition system, coding shape by viewpoint invariant geometric relations and also coding appearance information. The system (see Figure 2.7) can observe the work space by three pairs of Kinect and stereo cameras allowing for reliable and complete object information. From these sensors, we derive global viewpoint invariant shape features and robust color features making use of color normalization techniques. The recognition system is trained using Random Forests.

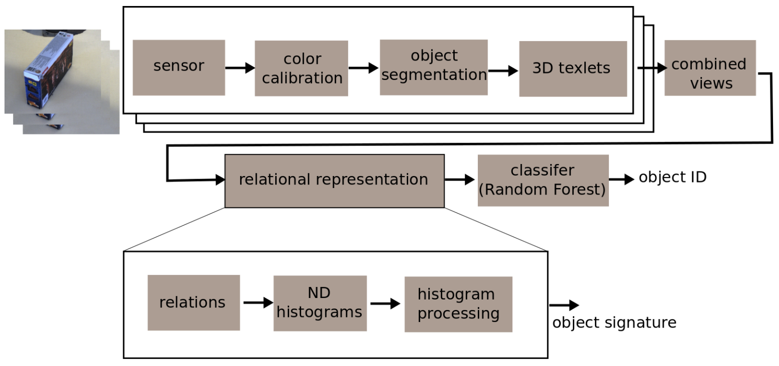


Figure 2.7: Object instance recognition.

We show that our system can achieve high performance already with a very low number of training samples, which is crucial for user acceptance and that the use of multiple views is crucial for performance. The system was evaluated on a dataset of 100 objects recorded under three lighting conditions. This indicates that our approach can be used in controlled but realistic contexts that require—besides high reliability—fast processing and an intuitive and easy end user experience.

Chapter 3

Conclusion

The described modules in this deliverable build the basis for the acquisition of sensorimotor experience needed to demonstrate structural bootstrapping in the context of the final demonstration. In particular, the deliverable describes advanced methods for visual object segmentation, feature extraction from 3D point clouds, pose estimation, visual collision detection as well as incremental learning of manipulation semantics from human observation.

References

[1] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter. Learning the semantics of object-action relations by observation. *The International Journal of Robotics Research*, 30(10):1229–1249, 2011.

[2] Anders Glent Buch, Jeppe Barsøe Jessen, Dirk Kraft, Thiusius Rajeeth Savarimuthu, and Norbert Krüger. Extended 3D line segments from RGB-D data for pose estimation. In *Scandinavian Conference on Image Analysis (SCIA)*, pages 54–65. Springer, 2013.

[3] Changhyun Choi, Alexander JB Trevor, and Henrik I Christensen. RGB-D edge detection and edge-based registration. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1568–1575, 2013.

[4] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3D object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 998–1005, 2010.

[5] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, 1999.

[6] G. Metta and P. Fitzpatrick. Grounding vision through experimental manipulation. *Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences*, 361(1811), 2003.

[7] Ajmal S Mian, Mohammed Bennamoun, and Robyn Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1584–1601, 2006.

[8] Nobuyuki Otsu. A Threshold Selection Method from Gray-level Histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

[9] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter. Voxel cloud connectivity segmentation - supervoxels for point clouds. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.

[10] David Schiebener, Julian Schill, and Tamim Asfour. Discovery, segmentation and reactive grasping of unknown objects. In *Proc. 12th IEEE-RAS International Conference on Humanoid Robots*, Osaka, Japan, 2012.

Attached Articles

[ATW14] E E. Aksoy, M. Tamosiunaite, and F. Wörgötter. Model-free incremental learning of the semantics of manipulation actions (in press). *Robotics and Autonomous Systems (RAS)*, 2014.

[JBK15] Troels Bo Jørgensen, Anders Glent Buch, and Dirk Kraft. Geometric edge description and classification in point cloud data with application to 3D object recognition. In *VISAPP International Conference on Computer Vision Theory and Applications*, 2015. (accepted).

[MPBK15] Wail Mustafa, Nicolas Pugeault, Anders Glent Buch, and Norbert Krger. Multi-view object instance recognition in an industrial context. *Robotica*, 2015. (Accepted with minor revision).

[SUA14] D. Schiebener, A. Ude, and T. Asfour. Physical interaction for segmentation of unknown textured and non-textured rigid objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, 2014.

[SVA14] D. Schiebener, N. Vahrenkamp, and T. Asfour. Visual collision detection for corrective movements during grasping on a humanoid robot. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, Madrid, 2014.

[SWS^{+}14] S. Stein, F. Wörgötter, M. Schoeler, J. Papon, and T. Kulvicius. Convexity based object partitioning for robot applications. In *International Conference on Robotics and Automation ICRA*, June 2014.

Model-Free Incremental Learning of the Semantics of Manipulation Actions

Eren Erdal Aksoy[a,*], Minija Tamosiunaite[a], Florentin Wörgötter[a]

[a] *Georg-August-Universität Göttingen, BCCN, Department for Computational Neuroscience Inst. Physics-3, Friedrich-Hund Platz 1, D-37077 Göttingen, Germany*

Abstract

Understanding and learning the semantics of complex manipulation actions are intriguing and non-trivial issues for the development of autonomous robots. In this paper, we present a novel method for an on-line, incremental learning of the semantics of manipulation actions by observation. Recently, we had introduced the Semantic Event Chains (SECs) as a new generic representation for manipulations, which can be directly computed from a stream of images and is based on the changes in the relationships between objects involved in a manipulation. We here show that the SEC concept can be used to bootstrap the learning of the semantics of manipulation actions without using any prior knowledge about actions or objects. We create a new manipulation action benchmark with 8 different manipulation tasks including in total 120 samples to learn an archetypal SEC model for each manipulation action. We then evaluate the learned SEC models with 20 long and complex chained manipulation sequences including in total 103 manipulation samples. Thereby we put the event chains to a decisive test asking how powerful is action classification when using this framework. We find that we reach up to 100% and 87% average precision and recall values in the validation phase and 99% and 92% in the testing phase. This supports the notion that SECs are a useful tool for classifying manipulation actions in a fully automatic way.

*Corresponding author
Email address: `eaksoye@physik3.gwdg.de` (Eren Erdal Aksoy)

1. Introduction

One of the main problems in cognitive robotics is how to recognize and learn human demonstrations of new concepts, for example learning a relatively complex manipulation sequence like cutting a cucumber. Association-based or reinforcement learning methods are usually too slow to achieve this in an efficient way. They are therefore most often used in combination with supervised learning. Especially the Learning from Demonstration (LfD) paradigm seems promising for cognitive learning ([1, 2, 3, 4, 5]) because humans employ it very successfully. The problem that remains in all these approaches is how to represent complex actions or chains of actions in a generic and generalizable way allowing inferring the essential “meaning” (semantics) of an action irrespective of its individual instantiation.

In our earlier studies we introduced the “Semantic Event Chain” (SEC) as a possible descriptor for manipulation actions [6, 7]. The SEC framework analyzes the sequence of changes of the *spatial relations* between the objects that are being manipulated by a human or a robot. Consequently, SECs are invariant to the particular objects used, the precise object poses observed, the actual trajectories followed, or the resulting interaction forces between objects. All these aspects are allowed to change and still the same SEC is observed and captures the “essence of the action” as demonstrated in several action classification tests performed by us [6, 7, 8, 9].

In this paper, we address the problem of on-line, incremental learning of the semantics of manipulation actions observed from human demonstrations. We use the concept of SECs as the main processing tool to encode manipulations in a generic and compact way. Manipulations are continuous in the temporal domain but with event chains we discretize them by sampling only decisive key time points. Those time points represent topological changes between objects and the hand in the scene which are highly descriptive for a given manipulation. Our main intent here is to design a cognitive agent that can infer and learn frequently observed spatiotemporal changes embedded in SECs in an unsupervised manner whenever a new manipulation instance occurs. The learning phase is bootstrapped only with the semantic similarities between SECs, i.e. the encoded spatiotemporal patterns, without using any prior knowledge about actions or objects. Since we use computer vision methods to derive event chains, our approach for incremental learning of semantics is highly grounded in the signal domain. To the best of our knowledge, this is the first attempt to apply reasoning at the semantic level, while being fully grounded at the signal level, to learn manipulations with an unsupervised method. Note, here – on purpose – we do not include any object- or other information to show the power of our methods to fully automatically and in an unsupervised way extract action and object information. Clearly, in praxis, it will often make sense to include whatever additional knowledge is available to further ease action understanding.

The paper is organized as follows. We start with introducing the state of the art. We next provide a detailed description of each processing step; extraction of SEC representations and learning model-SECs for each observed manipulation. In the next section, we discuss experimental results from the proposed framework, which also includes validation and testing of the learned models. We finally conclude with a discussion.

2. State of the Art

Learning from Demonstration (LfD) has been successfully applied both at the control [1, 2, 10] as well as the symbolic level [3, 4, 5]. Although various types of actions can be encoded at the control level, e.g. trajectory-level, this is not general enough to imitate complicated actions under different circumstances. On the other hand, at the symbolic level, sequences of predefined abstract action units are used to learn complex actions, but this might lead to problems for execution as many parameters are left out in a symbolic representation. Although our approach with SECs is a symbolic-level representation, SECs can be enriched with additional decisive descriptors (e.g. trajectory, pose, etc.) and do not use any assumption or prior knowledge in the object or action domain. Ideas to utilize relations to reach semantics of actions can be found as early as in 1975. For instance, [11] introduced the first approach about directed scene graphs in which each node identifies one object. Edges hold spatial information (e.g. LEFT-OF, IN-FRONT-OF, etc.) between objects. Based on object movement (trajectory) information, events are defined to represent actions. The main drawback of this approach is that the continuous perception of actions is ignored and is substituted instead by idealized hand-made image sequences. This approach, however, had not been pursued in the field any longer as only now powerful enough image processing methods became available from which object and action information can be extracted.

Still there are only a few approaches attempting to reach the semantics of manipulation actions in conjunction with the manipulated objects [12, 13, 14, 15, 16, 17, 18]. The work in [12] is one of the first approaches in robotics that uses the configuration transition between objects to generate a high-level description of an assembly task from observation. Configuration transitions occur when a face-contact relation between manipulated and stationary environmental objects changes. The work presented in [13] represents an entire manipulation sequence by an activity graph which holds spatiotemporal object interactions. The difficulty is, however, that very complex and large activity graphs need to be decomposed for further processing. In the work of [14], segmented hand poses and velocities are used to classify manipulations. A histogram of gradients approach with a support vector machine classifier is separately used to categorize manipulated objects. Factorial conditional random fields are then used to compute the correlation between objects and manipulations. Visual semantic graphs (inspired from our scene graphs) were introduced in [15] to recognize action consequences based on changes in the topological structure of the manipulated object. These visual semantic graphs were further employed together with a context-free manipulation action grammar in [19] to design a cognitive architecture for human manipulation action understanding. In [16] activity trees were presented to recognize actions using a minimal action grammar. The work in [17] suggested a method for hierarchical estimation of contact relationships (e.g. *on* and *into*) between multiple objects. Such contact relations were then employed to execute different daily tasks with robots. Abstract hand movements, such as *moving*, *not moving* or *tool used*, were extracted together with the object information in [18] to further reason about more specific action primitives (e.g. *Reaching*, *Holding*). Recent works such as [20] modeled human activities by employing the human skeleton information as well as roles of manipulated objects. In the modeling process they used the human skeleton information, object segments and their tracks. Likewise, the work in [21] introduced a Bayesian model by using hand trajectories and hand-object interactions while monitoring observed manipulations. In [22] hierarchical models of manipulations were learned with weak supervision from an egocentric perspective without using depth information. Although all those works to a certain extent improve the classification of manipulations and/or objects, none of them extracts key events of individual manipulations and learns a descriptive semantic model in a fully unsupervised manner to represent different manipulation tasks independent from the manipulated objects and their tracks.

In this sense, to our best knowledge, our work is the first study to evaluate and learn the semantics of manipulations in an incremental and model free manner. The concept of semantic event chains has been successfully utilized and extended by others [23, 24, 25, 26, 27, 28] not only to represent manipulation actions but also to replicate them by robots. The work in [23] presented active learning of goal directed manipulation sequences, each was recognized using semantic similarities between event chains. Our scene graphs were represented with kernels in [24] to further apply different machine learning approaches. Additional trajectory information was used in [25] to reduce noisy events occur in SECs. Others [26, 27, 28] showed execution of various manipulations with different robots by using the key spatiotemporal points provided by SECs.

3. Method

In this method section we will present the core algorithmic components where are complex details will only be given in the Appendix. This should make reading easier, while still everything is present to implement this algorithm if desired.

3.1. Data Acquisition

In this work, we investigate eight different manipulation actions: *Pushing*, *Hiding*, *Putting*, *Stirring*, *Cutting*, *Chopping*, *Taking*, and *Uncovering*. Fig. 1 (a)

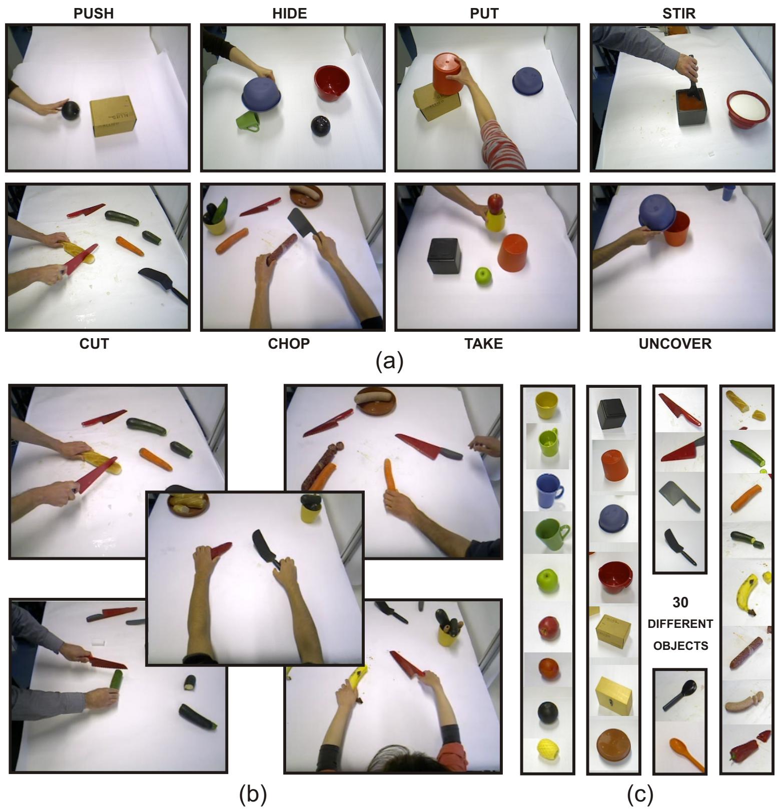


Figure 1: Eight different real action scenarios: *Pushing*, *Hiding*, *Putting*, *Stirring*, *Cutting*, *Chopping*, *Taking*, and *Uncovering*. (a) A sample original frame for each manipulation. (b) A sample frame from each demonstration of the *Cutting* action performed by 5 different individuals. (c) 30 different objects manipulated in all 120 manipulation demonstrations.

shows a sample frame for each manipulation action. All movies used in this study can also be found at `www.dpi.physik.uni-goettingen.de/~eaksoye/MANIAC_DATASET`. The *Pushing* action shows how a hand can move objects around randomly. In the action of *Hiding*, some objects are made invisible by covering them with other objects. In the *Putting* action objects are taken from the supporting background and put on top of each other. The *Stirring* action represents a scenario in which a spoon is used to stir some liquid in a bucket. In the *Cutting* action, a hand is cutting vegetables by moving a cutting tool back and forth. In the *Chopping* action, a cutting tool follows a straight trajectory to divide vegetables into parts. The *Taking* action represents a scenario where some objects are taken down and put on the supporting background. In the *Uncovering* action some objects are becoming visible after moving occluding objects away.

We recorded 15 different versions for each of these manipulations by asking 5 different individuals to demonstrate each manipulation 3 times with different objects in various scene contexts. Fig. 1 (b) depicts a sample frame from each individual demonstration of the *Cutting* action to give an impression of the differences in demonstrations. There are in total 30 different objects manipulated in all 120 demonstrations. All manipulated objects are shown in Fig. 1 (c).

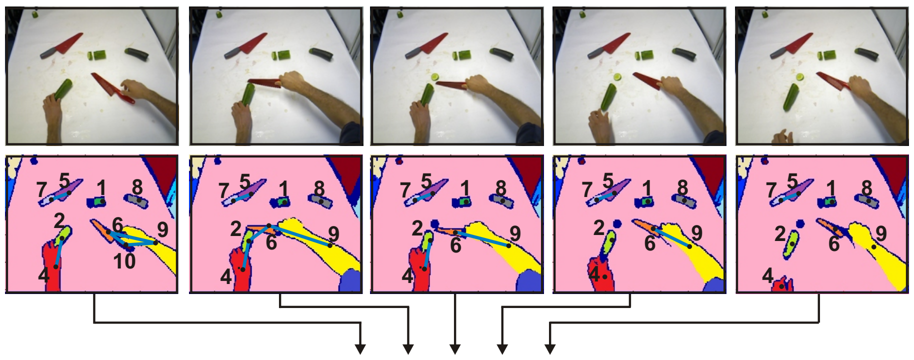
All manipulations were recorded with the Microsoft Kinect sensor which provides both color and depth image sequences. Colored objects are preferred to cope with the intrinsic limitations of the Kinect device. The central goal in these demonstrations is to learn a common archetypical SEC model for each manipulation including all possible variations in trajectory, pose, velocity, and object domains.

3.2. Segmentation and Tracking

The recorded image sequences are first pre-processed by a real-time image segmentation procedure to uniquely identify and track objects (including hands) in the scene. The segmentation algorithm is based on the color and depth information fed from the Kinect device and uses phase-based optical flow [29] to track segments between consecutive frames. Data transmission between these different pre-processing sub-units is achieved with the modular system architecture described in [30]. Segmentation and tracking approaches are described in detail elsewhere [31, 32], therefore, details are omitted here.

3.3. Extracting Semantic Event Chains (SECs)

Each consistently segmented image is represented by a graph: nodes represent segment centers and edges indicate whether two objects touch each other or not. By using the depth information we exclude the graph node for the background segment, i.e. supporting surface, since it does not play any crucial role in the dynamics of the manipulation. By using an exact graph matching technique, the framework discretizes the entire graph sequence into decisive main graphs. A new main graph is identified whenever a new node or edge is formed or an existing edge or node is deleted. Thus, each main graph represents a "key frame" in the manipulation sequence. All extracted main graphs form the core skeleton of the SEC, which is a matrix where rows are spatial relations (e. g.



$$\begin{array}{c} 9,6 \\ 4,2 \\ 6,2 \end{array} \begin{pmatrix} A & N & T & T & T & T & N & N & A \\ T & T & T & T & T & N & N & A & A \\ N & N & N & T & N & N & N & N & N \end{pmatrix}$$

Figure 2: SEC representation for a sample *Cutting* action where a hand is cutting a cucumber with a knife. Each column corresponds to one *key frame* some of which are shown on the top with original images, respective segments (colored regions), and main graphs. Rows are spatial relations between object pairs, e. g. between the hand (9) and knife (6) in the first row. Possible spatial relations are N, T, and A standing for *Not touching*, *Touching*, and *Absence*.

touching) between object pairs and columns describe the scene configuration at the time point when a new main graph has occurred.

Fig. 2 depicts the SEC representation with some sample *key frames* including original images, respective segments (colored regions), and corresponding main graphs for one of the *Cutting* action demonstrations. For instance, the first row represents the spatial relations between graph nodes 9 and 6 which are hand and knife, respectively. Note that, although the whole demonstration sample has approximately 1000 frames, it is now represented by a 3×9 matrix.

Possible spatial relations are *Not touching (N)*, *Touching (T)*, and *Absence (A)*, where N means that there is no edge between two segments, i.e. graph nodes corresponding to two spatially separated objects, T represents objects that touch each other, and the absence of an object yields A. In the event chain representation, all pairs of objects need to be considered once, however, static rows which do not contain any change from N to T or vise versa are deleted as being irrelevant. For instance, the relation between the left and right hand is always N and never switches to T to trigger an event, therefore, the respective row is ignored in the event chain. In Appendix A we introduce a de-noising process to cope with spurious spatial (rows) and/or temporal (columns) information propagated from noisy segmentation and tracking.

We note that there is no object recognition module included to identify graph nodes, i.e. segments, in the SEC framework. Event chains purely rely on spatial relational changes between segments in the temporal domain. The SEC extraction explained briefly in this section has been described in detail in [7].

3.4. Learning of Model SECs

The learning approach described next is an on-line unsupervised method to cluster observed SEC samples and to derive an archetypal SEC model for each cluster based on the semantic similarities between event chains. Each learned SEC model can then be used to describe a manipulation action.

Fig. 3 shows an overview of the proposed framework. The learning phase is triggered when a new manipulation experiment is observed; for example, a

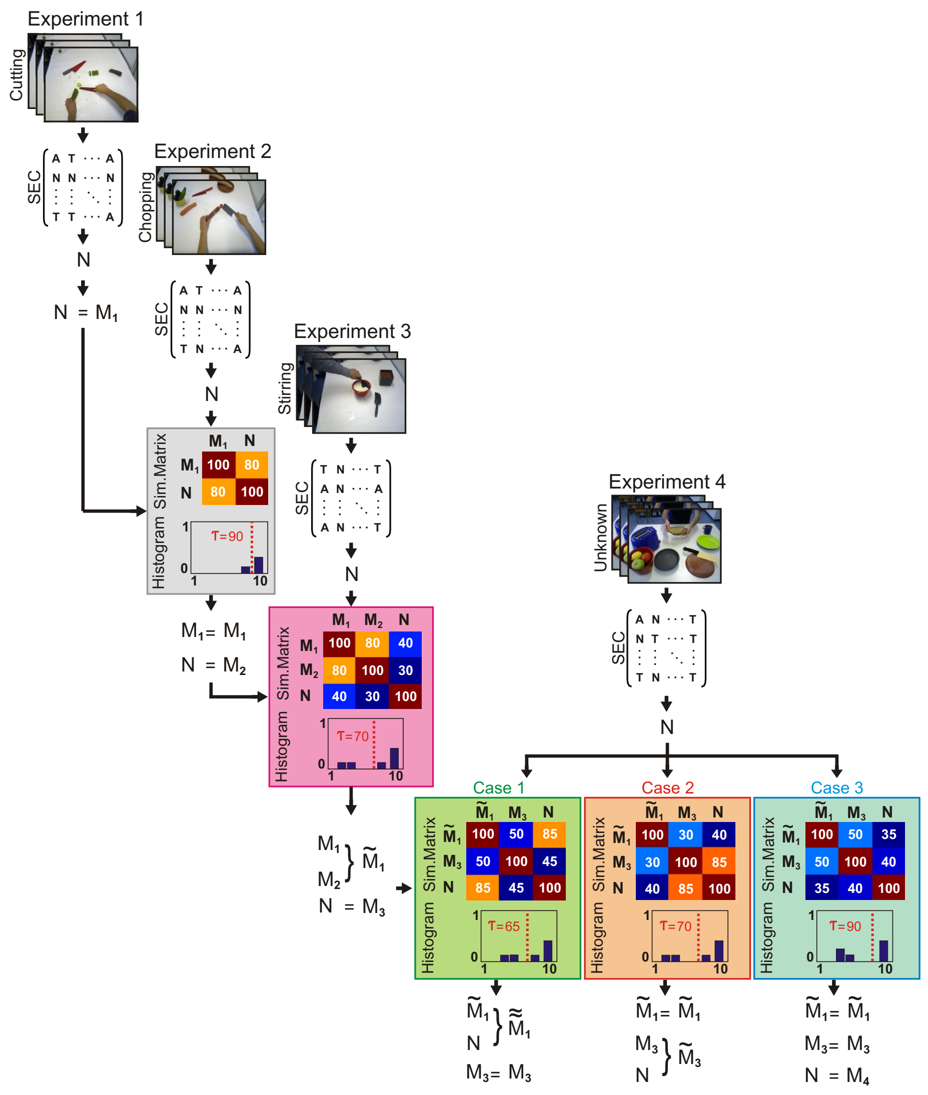


Figure 3: Overview of the proposed on-line learning framework.

Cutting manipulation sample is introduced as the first experiment in Fig. 3. The new observed sample is represented by an event chain to be compared with the already learned SEC models. If there is no model existing, as in the case for this very first manipulation observation, the currently observed SEC sample N is directly assumed as a new model M_1. Once a new manipulation example is acquired, e.g. a *Chopping* sample as the second experiment in Fig. 3, the framework measures semantic similarities between the new SEC sample N and the known model M_1 in the spatiotemporal domain. We provide a detailed explanation of the similarity measure in Appendix B.

Semantic similarity values between the known models and the new sample are stored in a matrix, called the similarity matrix ($\zeta_{semantic}$), which is then converted into a histogram ($\mathcal{H}$) representing the distribution of similarities. We apply the conventional Otsu's method introduced in [33] to the normalized version of the histogram to further compute a threshold τ. See section 3.4.1 for the details of the derivation of $\mathcal{H}$ and τ from $\zeta_{semantic}$. The gray box in Fig. 3 depicts extracted $\zeta_{semantic}$ and $\mathcal{H}$ in which the red dashed line indicates τ computed between the first two experiments.

Threshold τ is used for two purposes: First, we merge already learned SEC models which have higher semantic similarities than τ. Second, we compare the currently observed SEC sample with the so far existing models. If the comparison yields a higher similarity than τ, then the best fitting (highest similar) model will be refined with the new SEC sample. Otherwise, a new model will be created based on the SEC sample.

The comparison of the first two experiments N and M_1 shown in the gray box in Fig. 3 yields 80% semantic similarity which is less than τ estimated as 90% (See Appendix B). Therefore, the *Chopping* sample N is considered as a new SEC model M_2. We repeat the same procedure, i.e. computing $\zeta_{semantic}$, $\mathcal{H}$, and τ, once the next sample N, which is a *Stirring* experiment in this case, is observed. As depicted in the purple box τ drops below 80% which allows us to update M_1 with M_2 yielding $\widetilde{M_1}$. As the *Stirring* demonstration still has less similarities with any of the known models, a new model M_3 is initialized with N.

The threshold value is required to better assess the obtained semantic similarities between models and the observed sample. Therefore, whenever a new observation is available, the entire process of estimating a new τ by determining $\zeta_{semantic}$ and $\mathcal{H}$ is repeated to decide on the fly whether the current SEC sample belongs to one of the already learned manipulation models or whether it represents a new manipulation. This is summarized with the fourth experiment introduced as an *Unknown* demonstration in Fig. 3, the fate of which depends on three possible cases. Case 1 and 2 are respectively standing for the processes of refining the models $\widetilde{M_1}$ and M_3 with N, whereas Case 3 is representing the initialization of a new model M_4.

In the following, we will describe how to compute the threshold and update a learned model with a new SEC sample.

3.4.1. Computing the Threshold

Let $\mathcal{M}$ be a set of learned SEC models at any observation time as

$$\mathcal{M} = \{m_1, m_2, \cdots, m_n\} \quad , \tag{1}$$

where n is the total number of existing models. Semantic similarity values between all learned models are stored in a matrix as

$$\zeta_{semantic} = \begin{bmatrix} \varphi_{1,1} & \varphi_{1,2} & \cdots & \varphi_{1,n} \\ \varphi_{2,1} & \varphi_{2,2} & \cdots & \varphi_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{n,1} & \varphi_{n,2} & \cdots & \varphi_{n,n} \end{bmatrix} \quad , \quad 0 \leq \varphi_{i,j} \leq 100 \quad and \quad \varphi_{i,j} = \varphi_{j,i} \quad ,$$

where $\varphi_{i,j}$ holds the semantic similarity between models m_i and m_j and is computed as described in Appendix B.

Semantic similarity matrix $\zeta_{semantic}$ is then converted into a histogram $\mathcal{H}$ representing the distribution of similarities as

$$\mathcal{H} = \{h_k : \ k \in [1, \cdots, \lambda]\} \quad , \tag{2}$$

$$h_k = \frac{1}{\eta} \sum_{i=1}^{n} \sum_{j=i}^{n} \delta_{i,j} \quad , \tag{3}$$

$$\delta_{i,j} = \begin{cases} 1 & \text{if } \frac{\varphi_{i,j}}{\phi} \text{ is at bin } k \\ 0 & \text{else} \end{cases} , \tag{4}$$

where λ is the total number of bins each has a size of ϕ which is chosen as 10 in our experiments and η is the normalization factor. Note that, since the similarity matrix $\zeta_{semantic}$ is symmetric, only half of the matrix is processed, thus, the value of j changes from i to n in Eq. (3) and η is defined as $n(n+1)/2$.

The normalized histogram $\mathcal{H}$ is now used to calculate the required threshold using the conventional Otsu's method introduced in [33]. For this purpose, we compute zero- and first-order cumulative moments of the normalized histogram at each bin as

$$\omega(k) = \sum_{i=1}^{k} h_i \quad , \tag{5}$$

$$\mu(k) = \sum_{i=1}^{k} i h_i \quad . \tag{6}$$

The total mean value of the histogram is calculated as

$$\mu_T = \sum_{i=1}^{\lambda} i h_i \quad . \tag{7}$$

The variance of the histogram separability is then given by

$$\sigma_B^2(k) = \frac{[\mu_T \omega(k) - \mu(k)]^2}{\omega(k)[1-\omega(k)]} \quad . \tag{8}$$

Otsu's method yields a threshold value k^* for that bin at which the variance σ_B^2 is maximal; that is,

$$k^* = \underset{1 \leq k < \lambda}{\arg\max} \; (\sigma_B^2(k)) \quad . \tag{9}$$

The threshold k^* separates the histogram into two distinct regions. The left side of k^* indicates low semantic similarity between models in $\mathcal{M}$, and vice versa. As we are seeking for a threshold τ to group similar manipulations, we take the average of the similarities falling into the right side of k^* as

$$\tau = \frac{1}{\eta_r} \sum_{i=k^*}^{\lambda} h_i \quad , \tag{10}$$

where η_r is the normalization term which is the total number of similarity values on the right side of k^*.

3.4.2. Updating Model SECs

Once the highest semantic similarity between a novel SEC sample and any of the known models is higher than the threshold τ, the one model with *highest* similarity to the new SEC is now updated with this new SEC sample. To update a model, the learning procedure just needs to search for all common rows and columns observed in the new SEC sample.

Each model is initially created by assigning weight value of 1 for each row. Once a new SEC sample is observed, weights of each row in the model that match to a row in the new SEC are incremented. This way existing common rows between the matched model and the novel sample are receiving increasing weights. In the case of having additional rows in the new SEC sample, the model is extended by these rows, each of which is initiated again by giving them a weight of one. As the next step, we search for the common temporal information embedded in the columns of the event chains by employing a procedure very similar to that applied for extracting common rows. Finally, the model SEC consists of only those rows and columns observed frequently in the observed new SEC samples. A detailed explanation of the model updating procedure is given in Appendix C.

4. Results

In this section, we will first show experimental results from our proposed incremental learning framework. We will then continue with enrichment of each learned SEC model with object information. Next, validation and testing processes of the learned models will be given.

4.1. Learning

We apply the incremental learning and clustering framework described above to 8 different manipulation actions each of which has 15 versions, yielding in total 120 samples, as introduced in section 3.1. Manipulation tasks have vast variations in terms of manipulated objects, their poses, and followed trajectories as depicted in Fig. 1. The framework is first tracking each segment in the scene and extracting the corresponding SEC representation from a randomly observed manipulation sample. While observing more samples, different SEC models are learned or updated based on the threshold value.

When we let the framework run only once through 120 manipulation tasks by randomly choosing a sample at each time, it learns 22 model event chains. Fig. 4 (a) shows the final computed semantic similarity matrix $\zeta_{semantic}$ between each of the learned models. Low similarities between models indicate how distinct those models are. The corresponding histogram representation $\mathcal{H}$ with derived thresholds k^* and τ is depicted in Fig. 4 (b). The threshold k^* separates the histogram into two distinct regions as depicted with the gray shade and τ is then calculated as 72 from Eq. (10). In Fig. 4 (c), we can see the complete behavior of τ during the learning cycle with 120 observation samples.

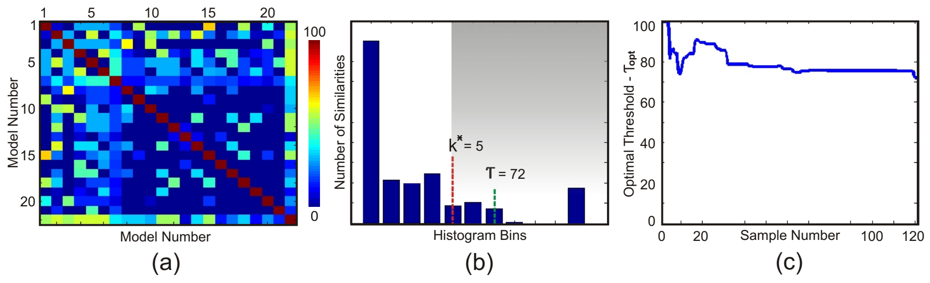


Figure 4: Thresholding. (a) Semantic similarity matrix $\zeta_{semantic}$ computed between 22 learned SEC models. The scale bar on the right indicates the similarity values in percent. (b) Respective histogram representation $\mathcal{H}$ with extracted k^* and τ values. The threshold k^* separates the histogram into two distinct regions as depicted with the gray shade. (c) Development of τ during the observation of 120 samples. τ is initiated with 100 and after updating with Eq. (10) it starts to converge to 72.

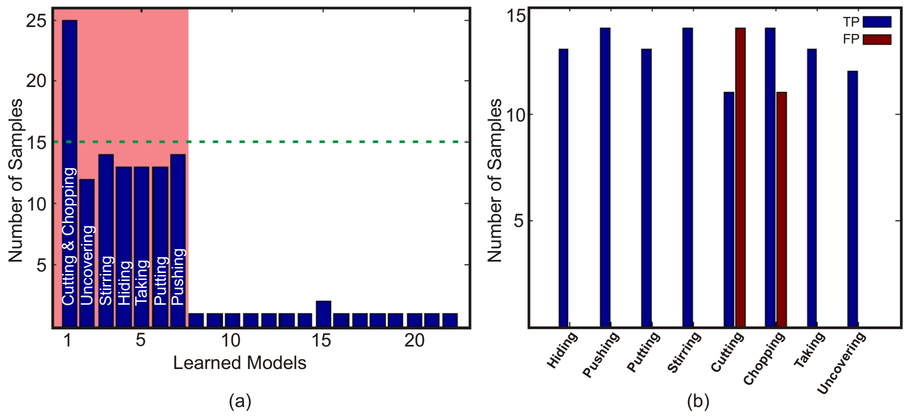


Figure 5: Number of learned models and clustering accuracy of observed samples. (a) Learned 22 SEC models with corresponding number of trained samples. The green dashed line indicates the actual sample numbers as the ground truth. (b) Number of true and false positive samples clustered in learned models with respect to the ground truth.

It is initiated with 100 and after updating with Eq. (10) at each observation it starts to converge to approximately 72.

Fig. 5 (a) depicts all learned models with corresponding number of observation samples employed for updating each. The green dashed line indicates the actual sample numbers as the ground truth. Although the framework learns in total 22 models, only 7 of them, those in the red box, contain more than 10 samples and the rest hold at most 2 samples. Recalling the fact that the training set has 8 manipulations, we can state as one central result that 7 of them are indeed found with high numbers of examples each. *Cutting* and *Chopping* models are merged, though, but we will explain below that this actually "makes more sense" than the naively (by us) assumed ground truth. Furthermore, we observe that only few demonstrated samples have either enormous variations or noise, i.e. less semantic similarities than τ with any other models, which leads to the generation of the additional models outside the red box. As mentioned, our framework produces a single model representing the *Cutting* and *Chopping* manipulations together due to having high semantic similarities. It is because both manipulations have the same fundamental action primitives, i.e. similar columns in the event chains, and the only differences are mostly in the followed trajectories and velocity of the movements which are not captured by SECs. See Fig. B.14 in Appendix B as an example of high semantic similarities between the *Cutting* and *Chopping* tasks. Thus, Fig. 5 (a) shows that without using any human intervention the proposed learning framework can automatically retrieve the demonstrated 8 manipulation types two of which are naturally merged.

As addressed in section 3.4, all manipulation samples used for updating the same SEC model will have the same cluster label. In Fig. 5 (b), we show the number of true and false positive samples falling into the same model with respect to the ground truth. Except for the *Cutting* and *Chopping* manipulations, none of the given manipulation samples is wrongly clustered. This means, for instance, a given *Stirring* demonstration is used only for updating the *Stirring* model, but not for the *Pushing* model, etc. However, since we have now only one SEC model for the *Cutting* and *Chopping* manipulations, samples from both manipulations will be used for the same model. As the ground truth expects two different models, high false positives are observed for both.

Fig. 6 shows how the clustering results for all 120 manipulation samples are varying from observation to observation. Colors encode the cluster labels and the ground truth for each cluster is given on the left. Note that time is progressing from left to right, thus the first observed sample is the one depicted in cyan in the *Chopping* manipulation. As a consequence of merging models with high semantic similarity, some clusters will merge once new observations become available. Black ellipses depict when a sample switches from one cluster to another. For instance, cyan clusters observed for the *Chopping* samples in the beginning are turned into red clusters originally created for the *Cutting* task. At the sample number 120 in the very right hand side we therefore observe 7 different colored clusters each from one learned model. This figure illustrates that for some manipulations types the model is immediately converging to the optimal solution, whereas for other models certain number of samples are required. Noisy clusters, which belong to the noisy models shown outside the red box in Fig. 5 (a), are indicated by black dots.

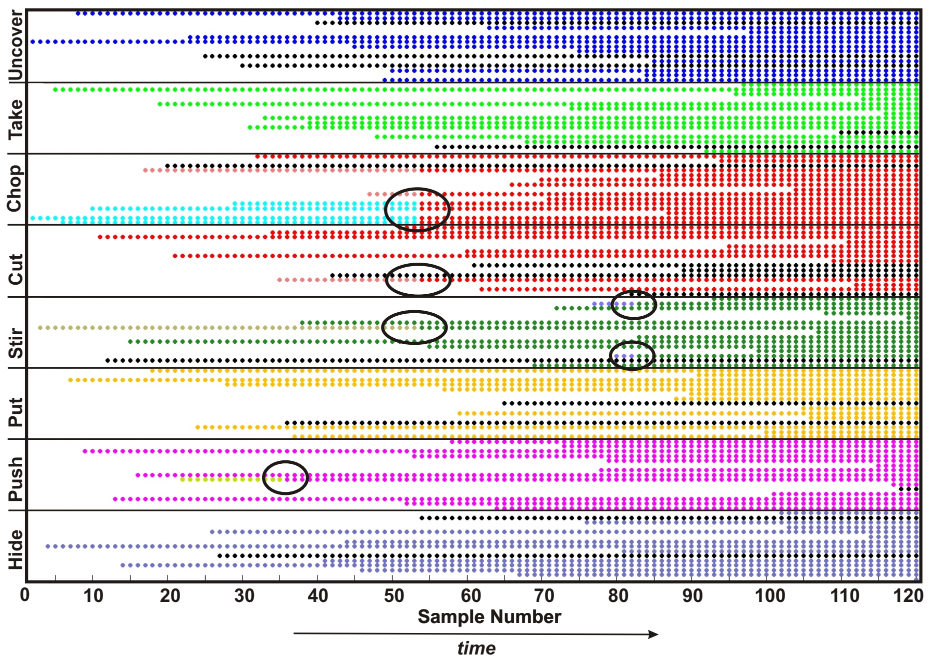
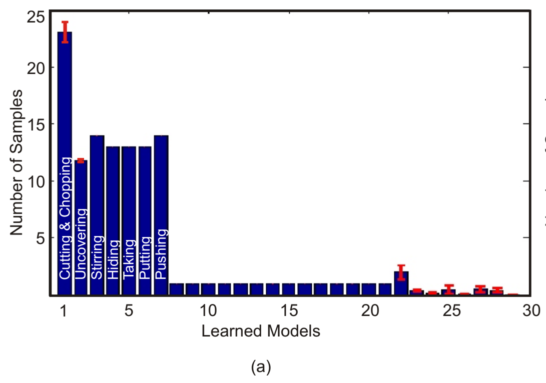


Figure 6: Clustering result of 120 manipulation samples. Colors encode the cluster labels and the ground truth for each cluster is given on the left. Noisy clusters are indicated in black. Black ellipses depict when a sample switches from one cluster to another.

To investigate the robustness of the framework, we repeat the same learning experiment explained above 100 times independently from each other and compute differences between the learned models. In each trial, the framework produces at least 21 and at most 23 various models. However, when we compare all these models extracted in 100 trials, we see indeed 29 different ones, the distribution of which is shown in Fig. 7 (a). Among those 29 models, it is again the same 7 models introduced in Fig. 5 (a) which have high number of samples. Furthermore, as indicated in Fig. 7 (b) we still do not obtain any false positives among the clustered samples except for the *Cutting* and *Chopping* manipulations due to the same reason as clarified above. Note that the red bars depict the standard error of the mean for those which are not zero. Fig. 7 consequently proves that the learning approach is always converging to the same 7 models no matter in which order the manipulation samples are provided.



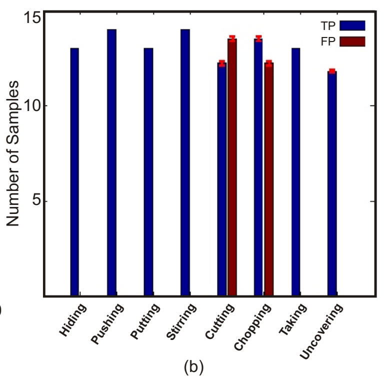


Figure 7: Total number of learned models and clustering accuracy after 100 independent trials. (a) Learned 29 SEC models with corresponding number of trained samples. (b) Number of true and false positive samples clustered in learned models with respect to the ground truth. Red bars depict standard error of the mean for those which are not zero.

We can now take a close look at some of those 7 SEC models explored from demonstrated manipulation actions. Fig. 8 shows models for the *Cutting & Chopping*, *Stirring*, and *Uncovering* manipulations with all derived states introduced in Eq. (C.2) and the transition probabilities between each. States and arrows given in red color correspond to the most commonly observed event chain columns and their transitions with the highest probabilities as described in Eqs. (C.3) and (C.4), respectively. On the left side of each model, we also show weight values ($\mathcal{W}$ from Eq. (C.1)) for each row in the states. It can be seen that in all 3 models some rows are quite commonly obtained in the trained samples since their weights are close to 1, whereas this is not the same for the state transitions. For instance, in the *Cutting & Chopping* model, there exist three more states given in gray color which are particularly observed in the second half of the action and cause drop of some state transitions to 0.28. This is because even though each subject grasps a tool and cuts or chops an object in the same temporal order, they leave the scene in different orders; for example, one subject first removes the hand supporting the object to be cut and then withdraws the hand holding the cutting tool whereas another subject either does it the other way around or removes both hands at the same time.

Another reason of having extra states, thus smaller transition probabilities, is the noise propagated from the segmentation and tracking components as observed in the *Stirring* model. Nevertheless, we can now extract all these variations that occurred due to the nature of manipulation or noise and pick the most often observed states, i.e. states in red, as a representative model for each manipulation action. Note that the learning process never ends and is open to refine models incrementally whenever new samples are provided, just like the assimilation process that happens in humans [34].

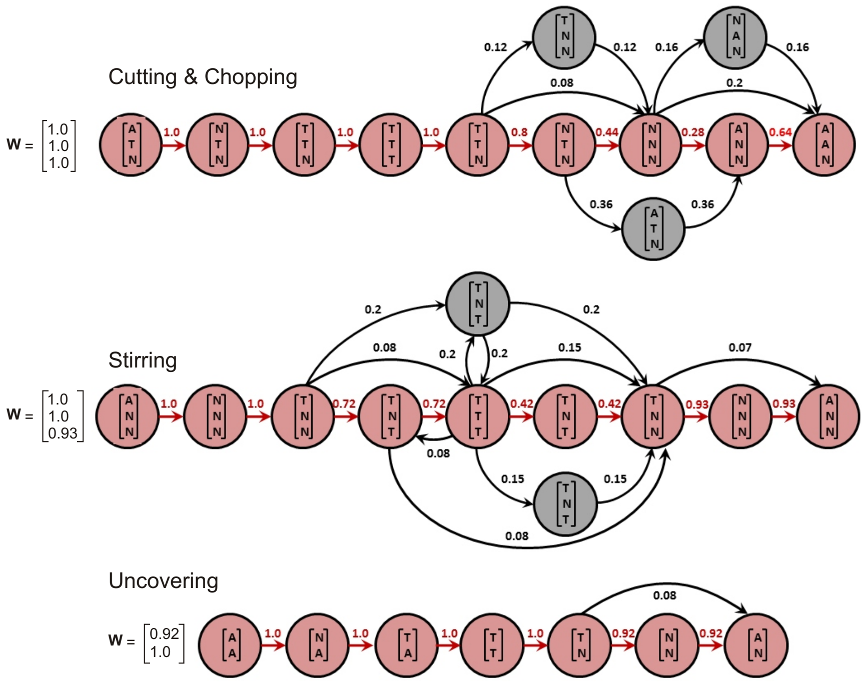


Figure 8: Complete learned SEC models for the *Cutting & Chopping*, *Stirring*, and *Uncovering* manipulations. Each state corresponds to one SEC column and arrows represent the transition probabilities from one state to the next. Those in red color correspond to the most commonly observed states, their transitions having the highest probabilities. Weight values $\mathcal{W}$ on the left indicate how often each row in the states is obtained in the trained samples.

4.2. Enriching Learned Model SECs

In this section, we will show how learned SEC models can be enriched with additional object information.

During the updating process of model SECs, we determine correspondences between rows of event chains as explained in section 3.4.2. Since each row in an event chain holds relational changes between segments in the scene, the row correspondences can also be used to calculate matchings between segments in two event chains. We refer the interested reader to [7] for details of the segment matching method. We now use this technique to extract segments, i.e. objects, that play the same role in different versions of the same manipulations observed during the learning phase.

Fig. 9 (a) shows the learned *Cutting* & *Chopping* model, columns of which are the states indicated in red in Fig. 8. The framework now estimates which segment is used as a main tool and which one as an object to be cut or chopped in each observation. As explained in Appendix A, we refer to the hand as the *manipulator* and to the object interacting with the hand as the *primary object*, e.g. a knife or a cleaver. Other objects which are combined with the primary object are called *secondary objects* like the cucumber to be cut. Note that the second hand is almost always used to help the *manipulator*, hence it is called the *supporter*. Fig. 9 (a) illustrates all matched *primary* and *secondary objects* used for training of the *Cutting* & *Chopping* model. Fig. 9 (b) shows

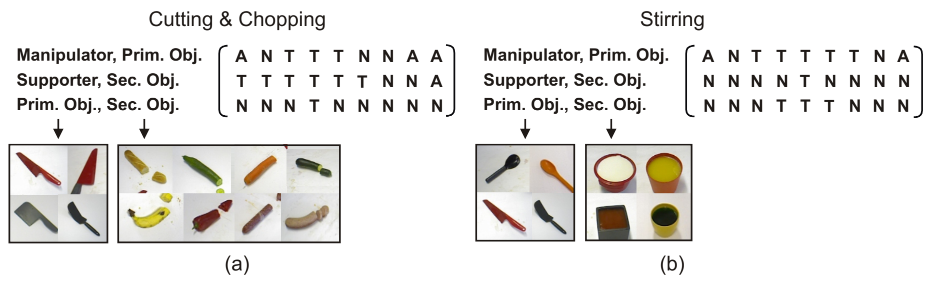


Figure 9: Learned *Cutting* & *Chopping* and *Stirring* models enriched with object information. Each column in the SEC model corresponds to one state indicated in red in Fig. 8. *Primary* and *secondary* objects are extracted from observed manipulations during the learning process.

the *primary* and *secondary objects* for the *Stirring* model. In this case, not only a spoon but a knife and a spatula are also selected by subjects as the *primary object* used for stirring. The *secondary object* is the stirred liquid and the buckets are the *supporters*. As learned model SECs are refined with every new observation, all these variations of the different objects will be attached to the model, simultaneously. Note that segments representing the *manipulator* and *supporter* are also matched, however, are not shown due to lack of space.

It is important to underline that the proposed framework is not utilizing any object recognition method, hence, we are here strictly at the level of segments. For the sake of simplicity, object images are shown instead of segments in Fig. 9. It is evident that this unsupervised segment categorization process could be coupled to object models, thus, providing access to object categorization, too.

4.3. Validation and Testing

A validation process of the learned 7 SEC models is performed with the classification of all 120 training samples according to their semantic similarities with the learned models. This step is required to show the clustering accuracy of the training data but nothing unexpected will be observed here. We note that the main and critical evaluation is then shown by the next following testing experiment with a set of novel and complex manipulation *sequences*.

We label each SEC model as a different class and introduce a static threshold chosen as 72 which is the converging value (τ) obtained during the learning phase as depicted in Fig. 4 (c). Once the highest semantic similarity between a training sample and any of the known models is higher than this threshold, the sample is assigned to that class. The classification method has also a class type called *Unknown* to detect samples that have low similarities with all known models.

Fig. 10 (a) shows the confusion matrix depicting the classification accuracies of the complete training data set with respect to the learned models. The first impression that the figure conveys is that there is no misclassification of any training data; for instance, 87% of the *Hiding* training manipulations are correctly classified in the model *Hiding* and the rest is assigned as *Unknown*. As there is only one representative SEC model existing for both *Cutting* and *Chopping* manipulations, training samples from those are assigned within the same model *Cutting & Chopping*. The validation phase of the complete training set leads to 100% average precision and 87% average recall.

As addressed in section 4.2, we can also extract objects which are manipulated in a similar manner in different demonstrations of the same manipulation type. Fig. 10 (b) indicates the primary object types frequently manipulated in each classified training data. It is observed that objects like *Knife*, *Cleaver*, and *Spatula* are manipulated often in the *Cutting & Chopping* model class, whereas, due to its size, *Bowl* is the only preferred object in the *Hiding* manipulation to cover other objects. Fig. 10 consequently proves the high success rates of the discriminative and descriptive features of the learned 7 SEC models and their direct relations with manipulated objects.

To further evaluate the performance of the learned model SECs, we create a new testing set with 20 long chained actions which consist of in total 103 different versions of the learned single manipulations such as *Cutting*, *Stirring*, and *Pushing*. We also introduce a new manipulation type called *Pouring* to

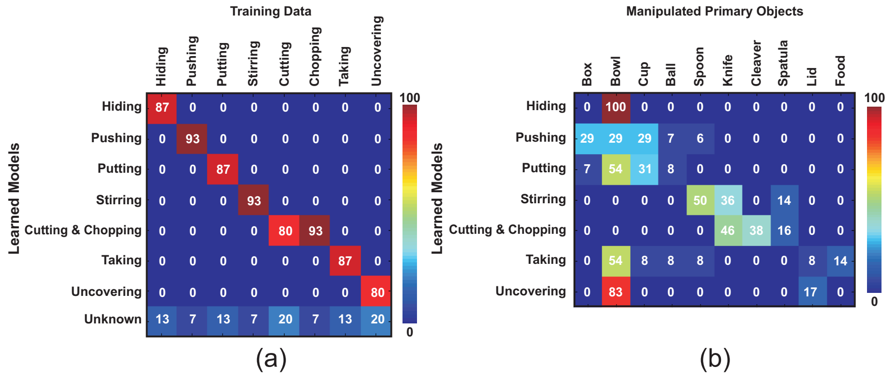


Figure 10: Confusion matrix showing (a) the classification accuracy for the complete training data set including in total 120 samples and (b) the usage rate of different objects primarily manipulated in the learned models.

measure the responses of the learned SEC models against a novel manipulation. In each chained action the subject has a certain task, e.g. "*making a sandwich*", which involves execution of multiple single manipulations in various orders, either sequentially or parallelly. Fig. 11 depicts sample frames from two different chained action sequences in which subjects are performing the same task "*making a sandwich*" by using novel objects in various ways to increase the complexity of the scenes. We here apply an unsupervised, probabilistic method that measures the frequency of the changes in the spatial relations embedded in event chains to extract the main manipulator, e.g. hand, and to decompose the long chained actions into their primitive action components according to the spatiotemporal relations of the manipulator. Hence, also the decomposition process is model free and automatic. Since the decomposition issue is not in the core of the proposed framework, we omit the details here and refer the interested

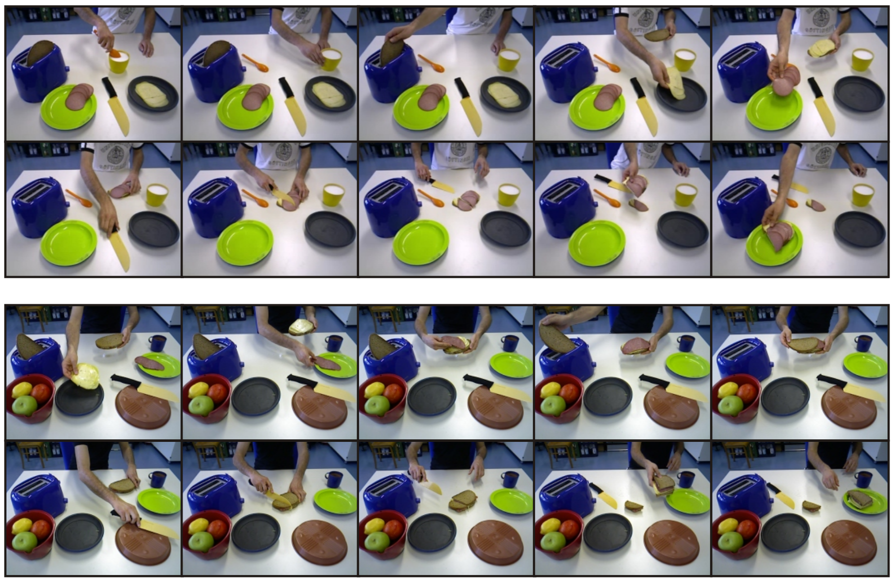


Figure 11: Sample frames from two different long chained manipulation sequences which are used to test the learned SEC models. In these demonstrations subjects are performing the same task "*making a sandwich*" by using novel objects in various ways to increase the complexity of the scenes.

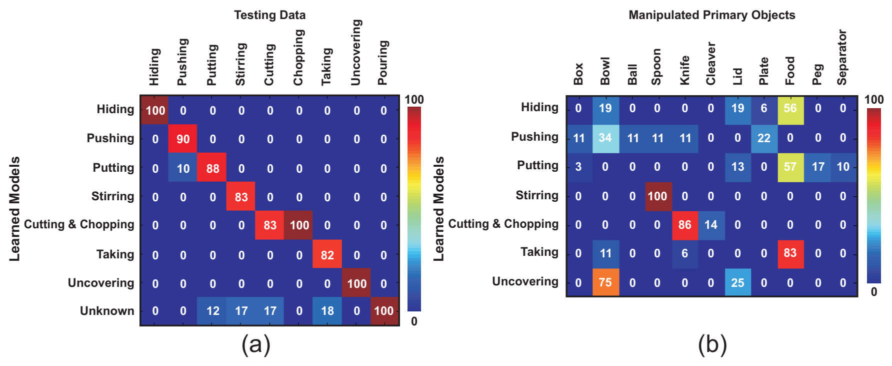


Figure 12: Confusion matrix showing (a) the classification accuracy for the complete testing data set including in total 103 samples and (b) the usage rate of different objects primarily manipulated in the learned models.

reader to [35].

Each single decomposed manipulation action is again analyzed as a classification task as described in the validation phase. Fig. 12 (a) indicates the highly successful classification results of decomposed manipulations with respect to the learned models. We receive minimum 82% accurate classification rate which is for the *Taking* manipulation and maximum 10% misclassification rate as observed only for the *Pushing* manipulation. It is also significant to note that the novel *Pouring* manipulation is never confused with any of the known SEC models. In this testing phase, average precision and recall values are measured as 99% and 92%, respectively.

Fig. 12 (b) shows the most often manipulated primary object types in each classified test data. Compared to the results obtained in the training phase, the major difference here is the high usage rates of the object type *Food* in the *Hiding*, *Taking*, and *Putting* models. This is because making a sandwich by taking and putting cheese or bread slices on top of each other naturally results in occlusions as expected by the *Hiding* model.

Note that all results shown in Figs. 10 and 12 are acquired in a fully automated, unsupervised manner and show that the learned SEC models are highly accurate and discriminative to recognize manipulation actions which can even be embedded in the long and complex chained demonstrations performed with novel objects under different circumstances.

5. Discussion

The main contribution of our paper is a novel method for incrementally learning the semantics of manipulation actions by observation. The proposed learning framework is bootstrapped with the semantic relations (SECs) between observed manipulations without using any prior knowledge about actions or objects while being fully grounded at the sensory level (image segments). To our best knowledge this is one of the first attempts in cognitive robotics to infer descriptive semantic models of observed manipulations in a fully automated and unsupervised manner.

In our previous work in [6] we only introduced the basic concept of event chains for the first time on $2D$ image sequences. In the subsequent work in [7] we presented the extended version of SECs together with a robust method to measure semantic similarities between event chains of different manipulations in a limited dataset with 3D image streams. Different from these works, the contribution of the current framework is manifold. We not only introduce an on-line learning method, but we also analyze robustness of the SEC concept with a large dataset including various complicated manipulations.

One of the most fundamental advantages of the proposed framework is that during the learning, when a new sample is observed, it is not compared with all previously acquired samples, which is an exhausting operation, but instead is compared only with the already learned models which are then updated accordingly. This is of importance to allow the cognitive agent to use its memory in a more efficient way for lifelong learning, which is known as "Assimilation Process" in human cognition as originally defined by Piaget [34].

The proposed framework can be easily enriched with object information since event chains naturally group objects considering only their performed roles in a manipulation. As a strong contribution, we showed that objects, i.e. segments, can be categorized based on how an object is being manipulated, rather than by knowing what type of object it is. As shown in our previous works [9, 28], not only object but pose and the followed trajectory information can also be embedded into the SEC representations as further enrichment.

In this paper, we also introduced a new manipulation action data set with 8 different manipulation tasks (e.g. *Cutting*, *Chopping*, *Stirring*, etc.), each of which consists of 15 different versions performed by 5 different human actors. This data set was used to learn an archetypal SEC model for each manipulation action. To further quantitatively evaluate the learned SEC models, we extended our data set with 20 long and complex chained manipulation sequences (e.g. "*making a sandwich*" or "*preparing a breakfast*") which consist of in total 103 different versions of these 8 manipulation tasks performed in different orders with novel objects under different circumstances. These data sets are publicly available and could be used for action/object benchmarking also of other methods.

In contrast to other well-known data sets, our new benchmark set captures manipulation activities from the subjects's own point of view with a static RGB-D camera since we are interested in understanding the spatiotemporal interactions between the manipulated objects and hands. The conventional data sets, however, employ the entire human body configurations and movements as main features and therefore either do not involve hand-tool features [36, 20, 37] or are not rich to provide enough recordings required for the learning [15, 16].

The observed high accuracy of our method when classifying the unknown (long-sequence) test-data set support that the learned models are indeed discriminative and descriptive of these actions (and objects). The here shown experimental results also exhibit a similar behavior to that of the ontologies presented in [38, 15]. In these both studies manipulation actions were classified into six distinct structural categories (e.g. *Rearrange*, *Destroy*, *Break*, etc.) in which *Cutting* and *Chopping* manipulations were subsumed in the same category as the learned single Cutting & Chopping model in our framework.

The main drawback that we are facing by the here presented framework is the *segment discontinuity* problem. Since we heavily rely on tracked scene segments, inconsistently tracked over-segmented scenes can lead to failures in the proposed method. We are currently investigating *feature binding* and *object permanence* concepts to reduce such failures. We are also aware of the fact that *touching* is a very naive and abstract event. However, to more efficiently use the memory we would like to keep our approach at this abstract level and apply fine-detailed scene analysis at a different level by further employing more action descriptors such as trajectory and pose information.

As mentioned in the introduction, additional information, if available about a given action, will further improve action understanding. This notwithstanding, we believe that the current study strongly supports the power of the Semantic Event Chain framework, because here we have "pushed it to an extreme" by fully relying on model-free, unsupervised algorithms for clustering and classification. Therefore, we would hope that this study might stimulate the research community to adopt this framework in the future.

Appendices

We here provide three appendices each describes details of individual algorithmic steps in details. The first appendix introduces the de-noising process to filter out noisy spatiotemporal relations in the event chains. In the next appendix, the detailed description of the similarity measure between event chains is given. The last appendix highlights the updating process of a learned SEC model with a novel SEC sample.

Appendix A. De-noising of SECs

Due to some early vision problems such as illumination variations or occlusions observed in the segmentation and tracking phases, extracted event chains can contain noisy spatial (rows) and/or temporal (columns) information. To prevent noisy event chain elements to propagate further to the next learning stage, we apply a de-noising process to the extracted raw SECs. The de-noising process is based on reasonable action descriptive assumptions (rules) introduced in [38], which are as follows:

1. only *single hand* manipulations are considered;
2. the hand can manipulate, i.e. *touch*, only one object at a time;
3. the manipulation can take place at the touched object itself (the one mentioned in rule 2) or only one other object can be a target, with which the first one interacts, i.e. *touches*;
4. before and after the manipulation the hand is *free* and *not-touching* anything;
5. before and after the manipulation the hand is *not in the scene*.

The first two rules guarantee that there is only one hand and at most one object interacting with the hand, which we call *manipulator* and *primary object*, respectively. Other objects, which are combined with the primary object, are called *secondary objects*. The third rule assures that *manipulator*, *primary* and *secondary objects* are the only ones having direct interaction with each other affecting the dynamics of the manipulation. The last two rules define the natural start and end points of the manipulation.

The de-noising process checks whether all those rules are satisfied in the SEC representation. For instance, the first two rules require that the event chain must have a row holding spatial relations between the *manipulator* and *primary object* and last three rules define these relations as:

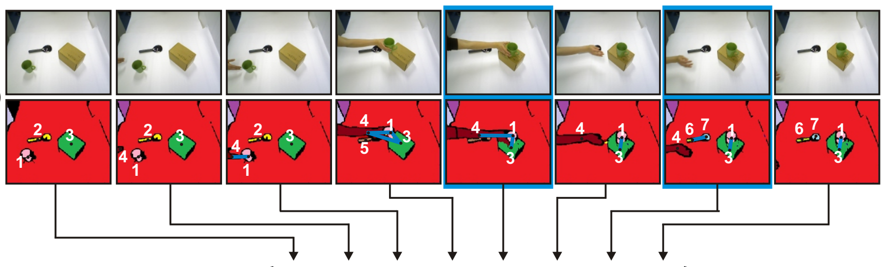
$$manipulator, primary\ object \quad \begin{bmatrix} A & N & T & \cdots & T & N & A \end{bmatrix}, \tag{A.1}$$

where the *manipulator* is first absent (A) in the scene *(rule 5)*, then appears but does not touch (N) the *primary object (rule 4)*. Next, the *manipulator* touches (T) *primary object* to apply a certain task on it *(rule 3)*. Depending on the manipulation, the temporal length of the touching (T) relation can vary. Finally, the *manipulator* releases (N) the *primary object (rule 4)* and leaves (A) the scene *(rule 5)*.

Since segments, i.e. graph nodes, are not identified as objects in event chains, we do not know which segment corresponds to the *manipulator* or *primary object*. Therefore, we apply a probabilistic reasoning to estimate segment roles in the manipulation. Probability values for each segment are assigned based on similarities of their relations with Eq. (A.1) and the frequency of their touching relations. See Appendix B for similarity calculation between SEC rows. In this regard, all rows in the event chain are compared with Eq. (A.1) and the most similar one is taken as the best candidate for the *manipulator* and the *primary object*.

Fig. A.13 (a-b) shows a noisy raw event chain with corresponding *key frames* extracted from a *Putting* manipulation sample where a hand is putting a cup on a box. For instance, the first and second rows of the SEC given in Fig. A.13 (b) are similar to Eq. (A.1), however, the second row has a higher probability to be

(a)



(b)
$$\begin{array}{c} 4,3 \\ 4,1 \\ 3,1 \\ 6,7 \end{array} \left[\begin{array}{cccccccc} A & N & N & T & N & N & N & A \\ A & N & T & T & T & N & N & A \\ N & N & N & T & T & T & T & T \\ A & A & A & A & A & A & T & N \end{array} \right]$$

(c)
$$\begin{array}{c} 4,1 \\ 3,1 \end{array} \left[\begin{array}{cccccc} A & N & T & T & N & A \\ N & N & N & T & T & T \end{array} \right]$$

Figure A.13: SEC representation for a sample *Putting* action where a hand is putting a cup on a box. (a) Extracted 8 *key frames* with original images, corresponding segments (colored regions), and main graphs. (b) Respective SEC where each *key frame* corresponds to one column. Rows are spatial relations between object pairs, e. g. between the hand (4) and box (3) in the first raw. Possible spatial relations are N, T, and A standing for *Not touching*, *Touching*, and *Absence*, respectively. (c) De-noised SEC after applying action descriptive rules. First and last rows as well as repetitive key frames, shown in blue frames, are removed from the raw SEC in (b).

a better candidate due to having more touching relations. Therefore, segments 4 and 1 in the second row have the highest likelihood to be the *manipulator* and the *primary object*. Since *rule 5* constrains the *manipulator* to appear in the scene later, we choose segment 4 as the *manipulator* and segment 1 as the *primary object*.

Once the *manipulator* and *primary object* are estimated, the de-noising process is concluded by examining the second and third rules once more. Since the second rule does not allow the *manipulator* to interact with any other object other than the *primary object*, such rows can be considered as noise to be omitted. In this manner, the first row in the SEC given in Fig. A.13 (b) can be ignored as the *manipulator* (segment number 4) is also touching the box (segment number 3) which is not the *primary object*. Note that the hand is here accidentally touching the box while putting the cup. Recalling the third rule, we can ignore any segment which does not have any interaction with the *manipulator* or *primary object*. In this sense, the forth row of the SEC in Fig. A.13 (b) is omitted because segment 6 and 7 represent the spoon which is occluded by the *manipulator* and *primary object* and not playing any role in the manipulation. Fig. A.13 (c) shows the final de-noised SEC representation for the *Putting* action in Fig. A.13 (a). Note that de-noised event chain includes less columns since redundant duplicate (repetitive) columns observed after deleting noisy rows (indicated in blue frames in Fig. A.13 (a)) are also removed.

It is important to underline that the de-noising process considers temporal interactions between entire segments in the manipulation to solve illumination or occlusion based early vision problems which can not be solved without reasoning at a higher level.

Appendix B. Measuring Semantic Similarity

Once event chains are extracted in the observation phase, their semantic similarities need to be compared to further explore whether they describe the same type of manipulation. In [7], we introduced a method to measure semantic similarities and here we describe an updated version which is more robust against noisy spatiotemporal information coming from the early vision stage. To better explain the semantic comparison we will use sample demonstrations from the *Cutting* and *Chopping* manipulations which are shown in Fig. B.14 (a-b) with extracted de-noised SECs including some sample key frames with respective segments and graphs. Note that even though those two samples have different perspectives and contain different number and types of objects, the dimensions of the event chains are accidentally the same. This is of no importance as our proposed method does not rely on dimensions, allowing to compare arbitrarily long manipulations.

To calculate the semantic similarity between two manipulations, spatial and temporal aspects are being analyzed in two separate steps. In the first step, we compare spatial information, i.e. relational changes in each row, and in the following second step the temporal information, i.e. the order of columns, is considered. In both steps we apply a standard sub-string search algorithm. To achieve this, we first perform a data-compression on the original chain (ξ_o) by simply scanning each row of ξ_o from left to right and substitute "changes" by combining their values into a two-digit format. For example a change from *Not touching* to *Touching*, hence from N to T, is now encoded by NT. When nothing has changed, a double digit like TT, is removed. This compressed event chain, represented by ξ_c, lost all temporal information and is used only for the spatial-relational analysis in the first step. The original chain (ξ_o) will then be used for the temporal analysis in the second step. ξ_o and ξ_c of the *Cutting* and *Chopping* actions are given in Fig. B.14 (a-d).

Let ξ_c^1 and ξ_c^2 be the sets of rows for the two manipulations, written as a matrix (e.g. Fig. B.14 (c) and B.14 (d)):

$$\xi_c^1 = \begin{bmatrix} r_{1,1}^1 & r_{1,2}^1 & \cdots & \cdots & r_{1,\gamma_1^1}^1 & \\ r_{2,1}^1 & r_{2,2}^1 & \cdots & r_{2,\gamma_2^1}^1 & & \\ \vdots & \vdots & \ddots & \vdots & & \\ r_{m,1}^1 & r_{m,2}^1 & \cdots & \cdots & \cdots & r_{m,\gamma_m^1}^1 \end{bmatrix},$$

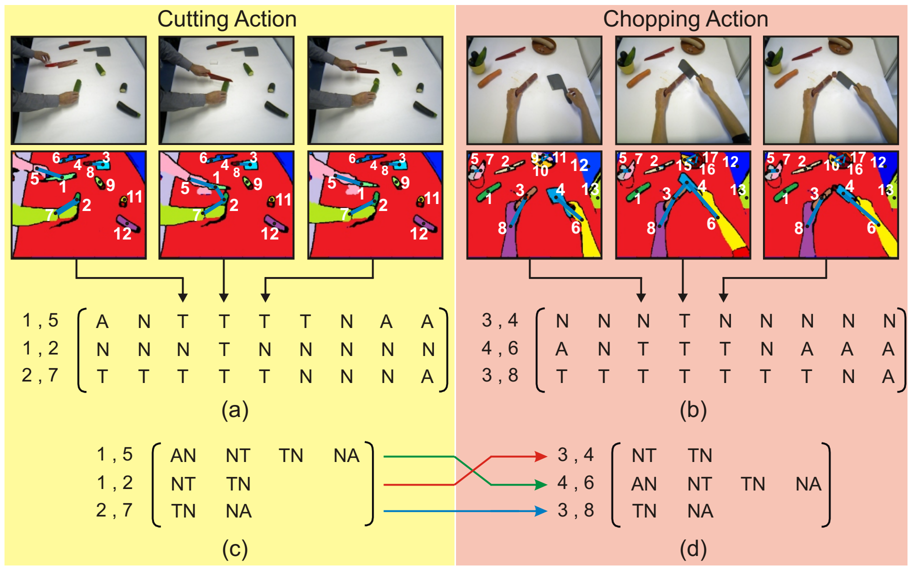
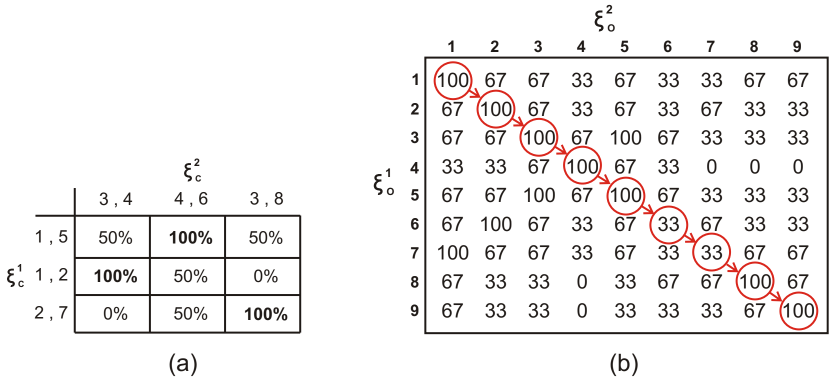


Figure B.14: Two sample manipulation action scenarios: *"Cutting a cucumber with a knife"* (on the left) and *"Chopping a sausage with a cleaver"* (on the right). (a-b) Extracted de-noised SECs (ξ_o) with some sample original key frames including respective segments and main graphs. (c-d) Corresponding compressed SECs (ξ_c). Colored arrows show row matchings.



(a)

ξ_c^1 \ ξ_c^2	3 , 4	4 , 6	3 , 8
1 , 5	50%	**100%**	50%
1 , 2	**100%**	50%	0%
2 , 7	0%	50%	**100%**

(b)

ξ_o^1 \ ξ_o^2	1	2	3	4	5	6	7	8	9
1	100	67	67	33	67	33	33	67	67
2	67	100	67	33	67	33	67	33	33
3	67	67	100	67	100	67	33	33	33
4	33	33	67	100	67	33	0	0	0
5	67	67	100	67	100	67	33	33	33
6	67	100	67	33	67	33	67	33	33
7	100	67	67	33	67	33	33	67	67
8	67	33	33	0	33	67	67	100	67
9	67	33	33	0	33	33	33	67	100

Figure B.15: Similarity matrices between the *Cutting* and *Chopping* samples given in Fig. B.14. (a) Spatial similarity matrix $\zeta_{spatial}$ indicates possible correspondences between rows (see colored arrows in Fig. B.14). (b) Temporal similarity matrix $\zeta_{temporal}$ with LCS matchings indicated in red circles shows correspondences between columns.

and

$$\xi_c^2 = \begin{bmatrix} r^2_{1,1} & r^2_{1,2} & \cdots & \cdots & \cdots & \cdots & r^2_{1,\gamma^2_1} \\ r^2_{2,1} & r^2_{2,2} & \cdots & \cdots & r^2_{2,\gamma^2_2} & & \\ \vdots & \vdots & \ddots & \vdots & & & \\ r^2_{k,1} & r^2_{k,2} & \cdots & \cdots & \cdots & r^2_{k,\gamma^2_k} & \end{bmatrix} ,$$

where $r_{i,j}$ represents a relational *change* between a segment pair

$$r_{i,j} \in \{AN, AT, NA, NT, TA, TN\} \ ,$$

where A, N, and T stand for *Not touching*, *Touching*, and *Absence*, respectively. The lengths of the rows are usually different and given by indices γ.

The first step is comparing the rows of the compressed event chains (ξ_c^1 and ξ_c^2) accounting for a possibly shuffling of rows in different versions of the same manipulations. Therefore, each row of ξ_c^1 is compared with each row of ξ_c^2 in order to find the highest similarity. The comparison process searches for equal entries of one row against the other using a standard sub-string search, briefly described next. Assume that we compare the a^{th} row of ξ_c^1 with the b^{th} row of ξ_c^2. If row a is shorter or of equal length than row b ($\gamma_a^1 \leq \gamma_b^2$), the a^{th} row of ξ_c^1 is shifted $\gamma_b^2 - \gamma_a^1 + 1$ times to the right. At each shift its entries are compared with the one of the b^{th} row of ξ_c^2 and we get as a result set $F_{a,b}$ defined as:

$$F_{a,b} = \{f_t : \ t \in [1, \gamma_b^2 - \gamma_a^1 + 1]\} \ ,$$

$$f_t = \frac{100}{\gamma_b^2} \sum_{i=1}^{\gamma_a^1} \delta_i \quad , \tag{B.1}$$

where γ_b^2 is the normalization factor and i is the row index and with

$$\delta_i = \begin{cases} 1 & \text{if } r^1_{a,i} = r^2_{b,i+t-1} \\ 0 & \text{else} \end{cases} \quad , \tag{B.2}$$

where the set $F_{a,b}$ represents all possible similarities for every shift t, given by f_t, which holds the normalized percentage of the similarity calculated between the shifted rows.

As usual for sub-string searches, we are only interested in the maximum similarity of every comparison hence we define:

$$M_{a,b} = \max(F_{a,b}),$$

For the case $\gamma_a^1 > \gamma_b^2$, a symmetrical procedure is performed by interchanging all indices of Eqs. (B.1), (B.2) above.

Spatial similarity values between all rows of ξ_c^1 and ξ_c^2 are stored in a matrix $\zeta_{spatial}$ with size $m \times k$ as

$$\zeta_{spatial} = \begin{bmatrix} M_{1,1} & M_{1,2} & \cdots & M_{1,k} \\ M_{2,1} & M_{2,2} & \cdots & M_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m,1} & M_{m,2} & \cdots & M_{m,k} \end{bmatrix}.$$

The final similarity value ($\psi_{spatial}$) between the rows of two compressed event chains is calculated by taking the mean value of the highest similarities across both rows and columns of $\zeta_{spatial}$ as

$$\psi_{spatial} = \frac{1}{m} \sum_{i=1}^{m} \max_j (M_{i,j}), \quad j \in [1, \cdots, k], \tag{B.3}$$

if

$$\max_j (M_{i,j}) = \max_t (M_{t,j}), \quad t \in [1, \cdots, m] \quad . \tag{B.4}$$

The spatial similarity matrix $\zeta_{spatial}$ indicates possible correspondences between rows of ξ_c^1 and ξ_c^2 used to compute temporal similarity in the second step. Note that there can be more than one correspondences between each row and all existing permutations need to be considered in the second step, separately. If there is a size differences between event chains, extra rows with no correspondences will be omitted here, but penalty values will then be applied at the end of the second step.

The complete similarity matrix ($\zeta_{spatial}$) between the *Cutting* and *Chopping* samples (ξ_c^1 and ξ_c^2) is given in Fig. B.15 (a) which shows that first row of ξ_c^1, i.e. $1, 5$, corresponds to the second row of ξ_c^2, i.e. $4, 6$. The same reverse relation exists between the second row of ξ_c^1 and the first row of ξ_c^2. Therefore, rows of the second event chain will be resorted by simply interchanging first and second rows to initiate the second step, i.e. temporal analysis of the method.

In the following second step, we use the time sequence, encoded in the order of columns in the original event chains, to find the best matching permutation and thereby arrive at the final semantic similarity. To this end we will now compare columns of resorted ξ_o^2 with that of ξ_o^1. Note that by contrast to rows, columns of event chains are never shuffled unless they represent different types of actions. Therefore, the column orders of type-similar event chains have to be the same. The comparison procedure of columns is very similar to the one for the rows. Since the lengths of the columns are the same, no shift-operator is required and columns are directly compared index-wise. Similarity values between all columns of ξ_o^1 and ξ_o^2 are stored in a matrix $\zeta_{temporal}$ with the size of $u \times v$ as

$$\zeta_{temporal} = \begin{bmatrix} \theta_{1,1} & \theta_{1,2} & \cdots & \theta_{1,v} \\ \theta_{2,1} & \theta_{2,2} & \cdots & \theta_{2,v} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{u,1} & \theta_{u,2} & \cdots & \theta_{u,v} \end{bmatrix} ,$$

where u and v are the lengths of columns in ξ_o^1 and ξ_o^2.

Once similarities between columns are calculated, we use "Longest Common Subsequence, (LCS)" in order to guarantee that the order of columns is the same. LCS is generally used to explore the longest sequence existing in both input samples sequences. Columns of event chains are used as sequences for this task and LCS matching is computed based on similarities in $\zeta_{temporal}$. Since the number of sequences is constant, the problem is solvable in polynomial time by dynamic programming. Fig. B.15 (b) shows $\zeta_{temporal}$ with LCS matchings indicated in red circles for the *Cutting* and *Chopping* samples ξ_o^1 and ξ_o^2.

The temporal similarity value $\psi_{temporal}$ between the columns of two event chains is then calculated by taking the mean value of the similarities given by LCS matching L_i as

$$\psi_{temporal} = \frac{1}{u}\sum_{i=1}^{u} L_i \quad , \tag{B.5}$$

$$L_i = \begin{cases} 100 & \text{if } \theta_{i,j} = 100 \\ 0 & \text{else} \end{cases} \quad , \tag{B.6}$$

where i and j are the matching column indices between ξ_o^1 and ξ_o^2.

Note that due to noisy segmentation and tracking, size of ξ_o^1 and ξ_o^2 can be different. Therefore, size differences between event chains are used as a penalty to prevent false similarities. The final semantic similarity is then computed as

$$\psi_{final} = \frac{r_1 c_1 \psi_{temporal}}{r_1 c_1 + \dfrac{r_2 c_2 - r_1 c_1}{\rho}} \quad , \quad r_1 < r_2 \quad and \quad c_1 < c_2 \quad , \tag{B.7}$$

where ρ is the penalty value and r_1, c_1, r_2, and c_2 are the number of rows and columns of ξ_o^1 and ξ_o^2, respectively. The final ψ_{final} value between the *Cutting* and *Chopping* samples in Fig. B.15 is calculated as 78% by using Eqs. (B.7), (B.6), and (B.7) with $\rho = 1$. The best matching permutation is further used for categorizing objects as described in [7].

Appendix C. Model Updating

Let ξ_m and ξ_n be two matrices representing a SEC model and a new SEC sample with sizes of $p \times q$ and $k \times l$, respectively. The two matrices can be written as

$$\xi_m = \begin{bmatrix} r^m_{1,1} & r^m_{1,2} & \cdots & r^m_{1,q} \\ r^m_{2,1} & r^m_{2,2} & \cdots & r^m_{2,q} \\ \vdots & \vdots & \ddots & \vdots \\ r^m_{p,1} & r^m_{p,2} & \cdots & r^m_{p,q} \end{bmatrix} \quad and \quad \xi_n = \begin{bmatrix} r^n_{1,1} & r^n_{1,2} & \cdots & r^n_{1,l} \\ r^n_{2,1} & r^n_{2,2} & \cdots & r^n_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ r^n_{k,1} & r^n_{k,2} & \cdots & r^n_{k,l} \end{bmatrix} \quad ,$$

where $r_{i,j} \in \{A, N, T\}$ is representing the spatial relations between each segment pair as described in section 3.3.

Each model ξ_m is initially assigned with a set of weights $\mathcal{W}$ as

$$\mathcal{W} = [w_1, w_2, \cdots, w_p]^T \quad , \tag{C.1}$$

for representing the appearance frequency of each row, which leads to extraction of all common rows observed in most of SEC samples. Each weight value w_i is initialized to 1. We first compare each row of ξ_n with each row of ξ_m to find identical matches and to further increment the corresponding weight values of the matched rows again by 1. This step is required since rows can be shuffled in the new observation sample ξ_n. While comparing rows, we search for only equal relational changes rather than temporal lengths of relations as explained in Appendix B. In the case of $k > p$, all novel rows observed in ξ_n will then be appended to ξ_m with weight values $\{w_{p+1}, \cdots, w_k\}$ initialized to 1. Common rows are then those with weights higher than $\frac{|\mathcal{W}|}{2}$. Next, the order of rows in ξ_n is resorted considering the order of their best matches with common rows in ξ_m. The sorting process yields the same row numbers in ξ_n and ξ_m, which is required for analyzing columns as described next.

The following step covers the temporal information embedded in the columns of ξ_n and ξ_m, and is similar to the previous approach explained for rows. We here assume that each column in an event chain is a *state* defining one action primitive. Hence, we seek for all primitives derived from new observations and compute the transition between them. Let $\mathcal{S}_m$ be a set of existing states in the current model ξ_m :

$$\mathcal{S}_m = \{s_1, s_2, \cdots, s_q\} \quad , \tag{C.2}$$

where each $s_i = [r^m_{j,i} : \; j \in [1, \cdots, p]]$. We now compare each state, i.e. column, in the sorted version of ξ_n with those in ξ_m by employing the same approach as defined for the temporal analysis in Appendix B. In the case of having more states in ξ_n, i.e. $l > q$, all novel states are appended to $\mathcal{S}_m$, and then transitions between each state are calculated. We assign a probability value $P_{i,j}$ defining the transition from s_i to s_j, which is incremented when two states are consecutive, i.e. $s_j = s_{i+1}$ in ξ_n.

Following state transition calculation, the learned model ξ_m is refined with the new states $\hat{\mathcal{S}}_m$ having the maximum transitions between each; that is,

$$\hat{\mathcal{S}}_m = \{s_{\alpha_1}, s_{\alpha_2}, \cdots, s_{\alpha_l}\} \quad , \tag{C.3}$$

$$\alpha_{t+1} = \arg\max_j \, (P_{\alpha_t, j}) \quad , \tag{C.4}$$

where $\alpha_0 = 1$ is for the initial state and $P_{i,j} = 0$ is the termination condition of the state sequence.

Note that in the process of creating a new model, $\hat{\mathcal{S}}_m$ will directly be equal to the states of ξ_n. In the case of merging similar models, i.e. those with high semantic similarity, one of the models will be assumed as ξ_n to employ the same refinement procedure explained above. It is also important to note that all SEC samples used for updating the same model ξ_m will be assigned with the same cluster label which yields self-clustering of observed SEC samples.

Acknowledgements

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

References

[1] S. Schaal, Is imitation learning the route to humanoid robots?, Trends in Cognitive Sciences 3 (1999) 233–242.

[2] A. Billard, S. Calinon, F. Guenter, Discriminative and adaptive imitation in uni-manual and bi-manual tasks, Robot. Auton. Syst. 54 (2006) 370–384.

[3] M. Pardowitz, S. Knoop, R. Dillmann, R. D. Zöllner, Incremental Learning of Tasks From User Demonstrations, Past Experiences, and Vocal Comments, IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics 37 (2) (2007) 322–332.

[4] S. Ekvall, D. Kragic, Robot learning from demonstration: a task-level planning approach, International Journal of Advanced Robotic Systems 5 (3) (2008) 223–234.

[5] R. Cubek, W. Ertel, Learning and Execution of High-Level Concepts with Conceptual Spaces and PDDL, in: 3rd Workshop on Learning and Planning, ICAPS (21st International Conference on Automated Planning and Scheduling), 2011.

[6] E. E. Aksoy, A. Abramov, F. Wörgötter, B. Dellen, Categorizing object-action relations from semantic scene graphs, in: IEEE International Conference on Robotics and Automation (ICRA), 2010, pp. 398–405.

[7] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, F. Wörgötter, Learning the semantics of object-action relations by observation, The International Journal of Robotics Research 30 (10) (2011) 1229–1249.

[8] E. E. Aksoy, B. Dellen, M. Tamosiunaite, F. Wörgötter, Execution of a dual-object (pushing) action with semantic event chains, in: Proceedings of 11th IEEE-RAS International Conference on Humanoid Robots, 2011, pp. 576–583.

[9] E. E. Aksoy, M. Tamosiunaite, R. Vuga, A. Ude, C. Geib, M. Steedman, F. Wörgötter, Structural bootstrapping at the sensorimotor level for the fast acquisition of action knowledge for cognitive robots, in: IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EPIROB), 2013.

[10] S. Niekum, S. Chitta, A. Barto, B. Marthi, S. Osentoski, Incremental semantically grounded learning from demonstration, Robotics Science and Systems (RSS) 2013.

[11] N. Badler, Temporal scene analysis: Conceptual descriptions of object movements, Ph.D. thesis, University of Toronto, Canada (1975).

[12] K. Ikeuchi, T. Suehiro, Toward an assembly plan from observation, part I: Task recognition with polyhedral objects, IEEE Trans. Robotics and Automation 10 (3) (1994) 368–385.

[13] M. Sridhar, G. A. Cohn, D. Hogg, Learning functional object-categories from a relational spatio-temporal representation, in: Proc. 18th European Conference on Artificial Intelligence, 2008, pp. 606–610.

[14] H. Kjellström, J. Romero, D. Kragić, Visual object-action recognition: Inferring object affordances from human demonstration, Comput. Vis. Image Underst. 115 (1) (2011) 81–90.

[15] Y. Yang, C. Fermüller, Y. Aloimonos, Detection of manipulation action consequences (mac), in: International Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 2563–2570.

[16] D. Summers-Stay, C. Teo, Y. Yang, C. Fermuller, Y. Aloimonos, Using a minimal action grammar for activity understanding in the real world, in: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, 2012, pp. 4104–4111.

[17] K. Nagahama, K. Yamazaki, K. Okada, M. Inaba, Manipulation of multiple objects in close proximity based on visual hierarchical relationships, in:

IEEE International Conference on Robotics and Automation, 2013, pp. 1303–1310.

[18] K. Ramirez-Amaro, E.-S. Kim, J. Kim, B.-T. Zhang, M. Beetz, G. Cheng, Enhancing Human Action Recognition through Spatio-temporal Feature Learning and Semantic Rules, in: IEEE-RAS International Conference Humanoid Robots, 2013.

[19] Y. Yang, C. Fermüller, Y. Aloimonos, A cognitive system for human manipulation action understanding, in: Proceedings of the Second Annual Conference on Advances in Cognitive System, 2013.

[20] H. S. Koppula, R. Gupta, A. Saxena, Learning human activities and object affordances from rgb-d videos, The International Journal of Robotics Research 32 (8) (2013) 951–970.

[21] A. Gupta, A. Kembhavi, L. Davis, Observing human-object interactions: Using spatial and functional compatibility for recognition, IEEE Trans. on Pattern Analysis and Machine Intelligence 31 (10) (2009) 1775–1789.

[22] A. Fathi, A. Farhadi, J. M. Rehg, Understanding egocentric activities, in: International Conference on Computer Vision, 2011, pp. 407–414.

[23] D. Martinez, G. Alenya, P. Jimenez, C. Torras, J. Rossmann, N. Wantia, E. E. Aksoy, S. Haller, J. Piater, Active learning of manipulation sequences, (in press), in: IEEE International Conference on Robotics and Automation (ICRA), 2014.

[24] G. Luo, N. Bergstrom, C. Ek, D. Kragic, Representing actions with kernels, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2011, pp. 2028–2035.

[25] R. Vuga, E. E. Aksoy, F. Wörgötter, A. Ude, Augmenting semantic event chains with trajectory information for learning and recognition of manipulation tasks, in: 22nd International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD), 2013.

[26] M. Wächter, S. Schulz, T. Asfour, E. E. Aksoy, F. Wörgötter, R. Dillmann, Action sequence reproduction based on automatic segmentation and object-action complexes, in: IEEE/RAS International Conference on Humanoid Robots (Humanoids), 2013.

[27] J. Papon, T. Kulvicius, E. E. Aksoy, F. Wörgötter, Point cloud video object segmentation using a persistent supervoxel world-model, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013.

[28] M. J. Aein, E. E. Aksoy, M. Tamosiunaite, J. Papon, A. Ude, 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.

[29] K. Pauwels, N. Krüger, M. Lappe, F. Wörgötter, M. M. Van Hulle, A cortical architecture on parallel hardware for motion processing in real time, Journal of Vision 10.

[30] J. Papon, A. Abramov, E. E. Aksoy, F. Wörgötter, A modular system architecture for online parallel vision pipelines, in: IEEE Workshop on Applications of Computer Vision (WACV), 2012, pp. 361–368.

[31] A. Abramov, K. Pauwels, J. Papon, F. Wörgötter, B. Dellen, Depth-supported real-time video segmentation with the kinect, in: Applications of Computer Vision (WACV), 2012 IEEE Workshop on, 2012, pp. 457–464.

[32] A. Abramov, E. E. Aksoy, J. Dörr, K. Pauwels, F. Wörgötter, B. Dellen, 3d semantic representation of actions from efficient stereo-image-sequence segmentation on GPUs, in: 5th International Symposium 3D Data Processing, Visualization and Transmission, 2010, pp. 1–8.

[33] N. Otsu, A Threshold Selection Method from Gray-level Histograms, IEEE Transactions on Systems, Man and Cybernetics 9 (1) (1979) 62–66.

[34] J. Piaget, The Origins of Intelligence in the Child, Routledge, London, New York, 1953.

[35] E. E. Aksoy, M. Tamosiunaite, F. Wörgötter, Decomposition of long manipulation actions (under review), Computer Vision and Image Understanding.

[36] C. Schuldt, I. Laptev, B. Caputo, Recognizing human actions: a local svm approach, in: Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, Vol. 3, 2004, pp. 32–36 Vol.3.

[37] A. Gupta, L. Davis, Objects in action: An approach for combining action understanding and object perception, in: Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on, 2007, pp. 1–8.

[38] F. Wörgötter, E. E. Aksoy, N. Krüger, J. Piater, A. Ude, M. Tamosiunaite, A simple ontology of manipulation actions based on hand-object relations, IEEE Transactions on Autonomous Mental Development 5 (2) (2013) 117–134.

Geometric Edge Description and Classification in Point Cloud Data with Application to 3D Object Recognition

Troels Bo Jørgensen, Anders Glent Buch and Dirk Kraft
The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Campusvej 55, 5230, Odense M, Denmark
{*trjoe,anbu,kraft*}@*mmmi.sdu.dk*

Keywords: Edge Detection, Object Recognition, Pose Estimation.

Abstract: This paper addresses the detection of geometric edges on 3D shapes. We investigate the use of local point cloud features and cast the edge detection problem as a learning problem. We show how supervised learning techniques can be applied to an existing shape description in terms of local feature descriptors. We apply our approach to several well-known shape descriptors. As an additional contribution, we develop a novel shape descriptor, termed *Equivalent Circumference Surface Angle Descriptor* or ECSAD, which is particularly suitable for capturing local surface properties near edges. Our proposed descriptor allows for both fast computation and fast processing by having a low dimension, while still producing highly reliable edge detections. Lastly, we use our features in a 3D object recognition application using a well-established benchmark. We show that our edge features allow for significant speedups while achieving state of the art results.

1 INTRODUCTION

Edge detection in general is a highly investigated topic in computer vision, mainly due to the possibility of condensing the input observations with a limited loss of information. This is beneficial also for 3D applications, e.g., point cloud enrichment (Gumhold et al., 2001) and pose estimation (Buch et al., 2013a), since it can decrease computation times. For these reasons, 3D edge detection should be fast, and it should be easy to use for general point clouds, containing noise and varying sampling densities. A 3D edge detection example is shown in Figure 1.

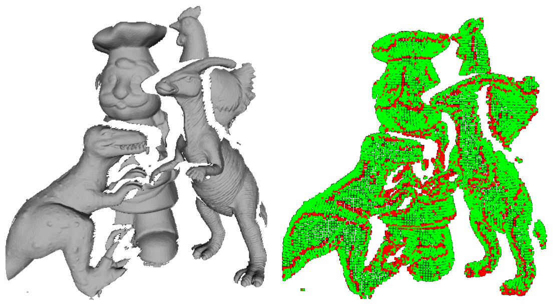


Figure 1: Left: a scene captured with a laser scanner (Mian et al., 2006). Right: edge detector response using our method (red means high confidence).

Several other methods have been proposed to tackled the issue, including (Gumhold et al., 2001; Guy and Medioni, 1997; Pauly et al., 2002; Pauly et al., 2003). These methods tend to rely on complex handcrafted analyses of large local neighborhoods in order to determine stable edge confidences. For this reasons they become computationally expensive.

We propose to use a staged approach to produce a simpler and faster algorithm, as done in 2D by e.g., Canny (Canny, 1986), but with very different processes since we are dealing with 3D data. We first estimate the edge direction using a local neighborhood. Then we compute our local ECSAD descriptor for describing the neighborhood, and then use the descriptor to refine the edge direction estimate. Based on this descriptor we provide two alternative methods for finding an edge confidence: 1) directly using a curvature estimate produced by our descriptor or 2) using machine learning techniques with labeled training data. Finally, we adopt a non-maximum suppression technique similar to that of Canny for our 3D edges to arrive at a more condensed representation of point clouds, which is desirable for matching tasks in e.g., object recognition applications.

We evaluate both our curvature based and learning based edge detectors against several other methods on point cloud data from multiple sensor modalities. For these experiments, we have manually annotated both training and test data, which provides a benchmark for comparing 3D edge detectors, and allows for future extensions. In a final application, we apply our edge representation to a 3D object recognition and 6D pose estimation system and show how to achieve both high recognition rates and significant speedups during this process.

This paper is structured as follows. We start by relating our work to other methods for edge detection in Section 2. Then we present our descriptor which is used for reliable edge detection in Section 3. We then show in Section 4 how any descriptor can be used for learning an edge detector. In Section 5 we present a simple edge thinning scheme for point clouds. In Section 6 we provide extensive experiments of various edge detectors, and we additionally show how to use our features for 3D object recognition. Finally, we make concluding remarks in Section 7.

2 RELATED WORK

The majority of edge detectors have been developed for 2D images, and one of the most common edge detection algorithms is the Canny edge detector (Canny, 1986), which revolves around semi-global methods in order to capture more salient features. In (Choi et al., 2013), Canny based methods were applied to RGB-D images from a Kinect camera in order to determine geometric and color based edges in organized point clouds.

Geometric edge detection in general 3D data structures has also gained some attention. For instance, (Bähnisch et al., 2009; Monga et al., 1991) have implemented edge detector in voxel based 3D images. These methods are largely extensions of the Canny detector to 3D, with a few modification to reduce computation times.

For unorganized point clouds, local point or direction information has been exploited to detect edges. In (Guy and Medioni, 1997) a PCA analysis of the normals is made in order to determine how much the surface varies. The work in (Pauly et al., 2003) proposes to use the curvature estimates at several different scales in order to determine a edge confidence. Gumhold et al. (Gumhold et al., 2001) propose to use a more complex combination of eigenvalues, eigenvectors and other curvature estimates in order to determine a handcrafted edge confidence. This paper also proposes to use a minimum spanning tree where short branches are removed in order to do edge thinning. A final spline fitting provides a smoother visual representation.

In this work, we address the detection of 3D edges in unorganized (or unstructured) point clouds. Such edges often occur at orientation discontinuities where two planar surfaces coincide. For this task, we have derived an appropriate local shape descriptor, termed ECSAD, which can be used for detecting edges, either directly by a curvature estimate produced by the descriptor or by learning an edge classifier in descriptor space. To our knowledge, current shape descriptors, such as e.g., (Johnson and Hebert, 1999; Mian et al., 2006; Rusu et al., 2009; Tombari et al., 2010), are focused strictly on the task of describing local shape patches of arbitrary geometry for use at a later matching stage.

Finally, we motivate the use of our edges and associated descriptors in a 3D object recognition application, where we also apply our descriptor for matching, leading to state of the art recognition performance. We note that, similar to our work, the edges detected in (Buch et al., 2013a; Choi et al., 2013) were also applied for object registration, in the latter case based on point pair features, originally proposed by (Drost et al., 2010). However, the edge detection method of (Choi et al., 2013) is restricted to organized RGB-D images, and not general 3D shapes, which renders evaluations against our work impossible. We do, however, compare ourselves with the registration algorithm of (Drost et al., 2010).

3 LOCAL SURFACE DESCRIPTOR FOR EDGE DETECTION

We have developed a local descriptor focusing on edge detection and classification, partly to determine the direction of the edges, and partly to be used in supervised learning for edge detection. The descriptor is a vector of relative angles between opposing sides of the edge, which we have found to provide a good description for geometric edges caused by orientation discontinuities. Before descriptor estimation, the input point cloud is down-sampled to a uniform resolution. The radius of the spherical support (the area that influences the descriptor) is a free parameter, but we have consistently used a value of five times the down-sampling resolution for simplicity.

As will be explained in the following, our descriptor uses a spatial decomposition which gives each spatial bin approximately the same circumference. Contrary to other descriptors that use histograms, our uses simple but stable angle measurements. For these reasons, we term our descriptor *Equivalent Circumference Surface Angle Descriptors* (ECSAD).

Spatial Decomposition Similar to other local surface descriptors, we use a spatial decomposition,

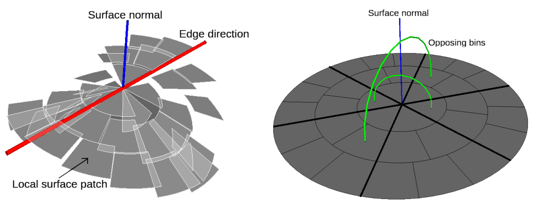


Figure 2: Left: visualization of the computed descriptor at an edge point with a tangent direction (red) and a surface normal (blue), please see text for a description how these are defined. The spatial bins are intersected by a number of surface points, and the contents of each bin is visualized by the plane patch spanned by these intersecting surface points. Right: a cross section showing the tangent plane of the local support, showing the decomposition used by our descriptor.

which is illustrated in Figure 2 by a cross section through the spherical region. Our descriptor splits the local space along the radial and azimuth dimensions, but not along the elevation as in e.g., (Frome et al., 2004; Tombari et al., 2010). This choice is justified by the fact that it is extremely rare that more than one surface passes through the same azimuth bin at different elevations, thus resulting in a high fraction of empty bins along the elevation. This again leads to instabilities towards position noise, and we have found this to produce worse results for edge detection and description. In Figure 2, left this can be seen as noise in the elevation of the surface patches.

Instead, we have devised a more sophisticated and uneven binning of the azimuth dimension (see Figure 2, right). We start by splitting the local space into six equiangular azimuth bins of 60° through all radial levels (bold lines). Now, for each of these six azimuth bins, we increase the number of azimuth splittings by one for each radial increment, giving a total increase of six azimuth bins per radial level (thin lines). This leads to an almost uniform angular coverage of all the bins in the azimuth dimension, and we have found this to produce much better performance than simply using an equal number of azimuth bins at all radial levels. We have tested different numbers of radial levels and found a good compromise between specificity and robustness for four radial levels (note that only three radial levels are shown in Figure 2, right).

Reference Frame Estimation and Bin Angles The first step of the algorithm is to estimate the surface normal and a tangential edge direction of the center point which is to be described. This is done by the eigendecomposition of the scatter matrix of all the points in the support, giving the direction and normal along the eigenvectors corresponding to the largest and smallest eigenvalues, respectively. The local reference frame (LRF) x- and z-axis is given by these two vectors, and the y-axis by their cross product. Then we map each of the supporting points into the correct spatial bin based on its radial and azimuth coordinates relative to the center point. This is done using the direction vector from the center point to the supporting point. The radial component is immediately given by the norm of this vector, while the azimuth component is given by the relative angle between this vector and the x-axis, measured in the tangent plane of the normal vector. For each bin, we now compute the relative angle between the surface normal and the direction vector to each point in the spatial bin. This angle is then averaged over all points that fall in the same spatial bin, giving a single angle measurement per spatial bin. After the angles to the individual bins have been determined, an interpolation strategy is used to assign values to bins with missing information, i.e., bins which have no points.

The interpolation value of a bin is performed by averaging the angles of up to five neighbor bins: one at a lower radial level, two next to the bin at the same radial level, and the two closest at a higher radial level. The neighbors at the same and at the higher radial level are only used if they contain points and thereby an angle measurement. The interpolation then starts from the center and moves outwards. At the first radial level, the bin angle at a lower radial level defined as zero. This ensures a value will be assigned to every bin.

Description Using Sum of Angles At an edge point, the x-axis separates two surfaces meeting at the center point. Our descriptor tries to approximate the angle between these two surfaces using the individual angle measurements of the spatial bins. To achieve this we identify opposing spatial bins, i.e., bins that have the same radial component but separated by an azimuth angle of π. We now take the sum of angles of each opposing bin pair, reducing the number of angle observations by a factor of two (green lines in Figure 2, right). Each angle sum approximates the angle between the coinciding surfaces, but this summation also makes our descriptor invariant to the sign of the x-axis, which is desirable, since this direction is ambiguous.

Reference Frame Refinement and Curvature Estimate A special case occurs in concave regions, i.e., at points where the normal vector (z-axis) has an angle of less than $\pi/2$ to the two opposing surfaces. This can easily be measured by checking if the average of the sum of angles defined above is larger than π. In such cases, we negate the y- and z-axis of the LRF. Finally, we perform a refinement of the x-axis by treating the sum of angle entries as a local 2D map, where each entry equals the angle measurement weighted by the radial component. We compute the eigendecomposition of the covariance matrix of this local 2D map of weighted angle entries, and the edge direction will now be better approximated by the in-plane eigenvector of the smallest eigenvalue. We rotate the LRF around the z-axis to coincide with the updated x-axis. Using this refined RF, we now recompute all the sum of angle measurements to get a more robust descriptor.

As a side effect, the biggest eigenvalue of the local 2D map computed above provides a good estimate of the the local curvature around the edge. In Section 6 we show results of using this measure for edge detection. In all the experiments, we have used four radial levels, leading to a descriptor dimension of $(6+12+18+24)/2=30$

In order to use the descriptor in pose estimation, it is beneficial to orient the normals to point outwards from the underlying objects. This is done to improve correct match rates, since it enables distinction between convex and concave regions. For scenes, this is done by rotating the scene normals towards a viewpoint. For models it is done based on a technique proposed by (Hoppe et al., 1992).

If the normal signs are changed, the descriptors are updated, similarly to how concave regions are oriented to produce similar descriptors for concave and convex regions to simplified edge detection.

4 SUPERVISED LEARNING FOR EDGE DETECTION

Using our ECSAD descriptors, a random forest (RF) classifier (Breiman, 2001) was trained in order to determine edge confidences in point clouds containing structured noise, such as point clouds captured by range sensors.

The training dataset consists of manually labeled point clouds, captured by Kinect cameras, stereo cameras and sampled from CAD models. Examples of labeled point clouds from these different sources are seen in Figure 3. Here the red lines are positive edge examples, the blue lines are ignored due to uncertainty of the human annotator, and the rest of the points are negative examples. We trained using four CAD models, two stereo scenes, two Kinect scenes and three Kinect views of different objects. All in all this provided more than 12500 positive and 285000 negative training examples.

These data, along with the local feature descriptors computed over the full point clouds, were then

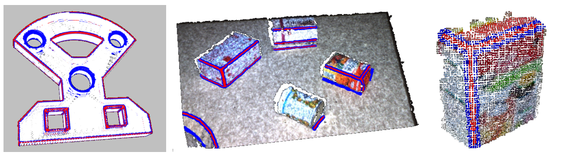


Figure 3: Examples of labeled point clouds from various sources used for training (relative sizes are not preserved in this figure). Ground truth edges (red) are used as positive examples, and transition regions between edges and non-edges (blue) are discarded during training. The rest of the points are non-edges, which are used as negative examples. Left: an ideal CAD model, resampled to a point cloud. Middle: a real scene with projected texture pattern, reconstructed by a block matching algorithm. Right: a partial view of a textured object, taken from the RGB-D Dataset (Lai et al., 2011).

used in order to train the random forests. Based on multiple runs over different parameters, we found that a point cloud resolution of 4 mm, a support radius of 20 mm, and an RF with 30 trees and a maximum tree depth of 15 provided good results. Similar figures hold for the other methods which we will compare against in Section 6.

In the test phase, a new point cloud with computed feature descriptors is fed to the RF classifier. The output edge confidence at a feature point is then simply given by the number of trees in the RF that classify the feature as an edge.

A smoothness technique is applied to the edge confidences, which is beneficial as an extra step before applying non-maximum edge suppression for thinning the edge map. This is simply implemented by determining the ten closest points to the edge point in question and averaging the edge confidences. After this step, the cloud is ready for non-maximum edge suppression.

5 EDGE THINNING

For some applications, e.g., object recognition, a sparse representation can be desirable. One of the simplest solutions in our case is to use a non-maximum edge suppression technique. This is implemented by determining the 20 closest edge points to a potential edge. Denote the current center point as p_C and a neighbor as p_N, both with associated edge directions d_C and d_N and edge confidences $c(p_C)$ and $c(p_N)$. To determine whether p_C suppresses p_N, three criteria are used. First, p_C must have the highest edge confidence:

$$c(p_C) > c(p_N) \tag{1}$$

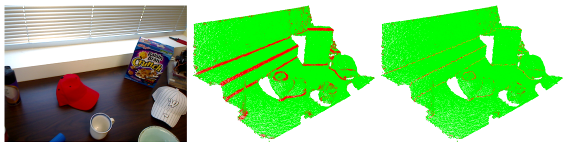


Figure 4: Left: a table scene from the RGB-D Dataset. Middle: edge detector responses. Right: remaining edges after non-maximum suppression.

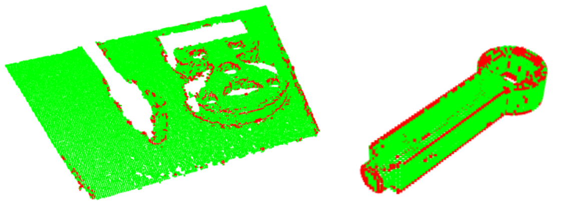


Figure 5: Edge responses after non-maximum suppression. Left: A real scene with projected texture pattern, reconstructed by a block matching algorithm. Right: An ideal CAD model resampled to a point cloud.

Secondly, we impose the following collinearity constraint:

$$\angle(d_C, p_N - p_C) > \frac{3}{8} \cdot \pi \quad (2)$$

This ensures that points on the same line do not suppress each other. Thirdly, we require the following:

$$\angle(d_C, d_N) < \frac{\pi}{4} \quad (3)$$

This ensures that orthogonal edges do not suppress each other. As an optimization, we check (1)–(3) bidirectionally, i.e., if a neighbor point suppresses the center point, the center point is discarded. This step is done to reduce the number of neighborhood searches, which is computationally expensive in point clouds.

A visualization of this suppression process is shown for a Kinect scene in Figure 4. We note that the thinning is an optional step, and in this paper we use it only in the final object recognition application. For a fair comparison of edge detectors, it is more appropriate to directly use the output edge confidences, as we will show in Section 6.1.

Figure 1 and Figure 5 show the edge response after line thinning for three other point clouds . Here it is seen that the detector has a decent response for all data sources, but it should be noted that the response near borders is poor, partly due to higher noise levels in these areas. In the Kinect point cloud it is also seen that the responses become poor at the most distant parts of the scene, where the noise and quantization levels are particularly high.

6 EXPERIMENTS

In this section we provide experimental results both for our edge detection algorithms, and for an object recognition application. All algorithms were implemented in single-threaded C++ applications, primarily using functionality from the Point Cloud Library[1] (Rusu and Cousins, 2011). OpenCV[2] was used for its interface to machine learning algorithms.

The algorithms was evaluated using an Intel core i3 3217U, 1.8GHz with 4GB RAM. This computer is roughly equivalent to the one used by Drost et al. (Drost et al., 2010).

We have tested a range of parameters for our method, and the performance varies between different data sources (Kinect, CAD and stereo). A full evaluation of these parameters and their influence on the performance on various data sources is beyond the scope of this paper. In this section we present results using the previously mentioned parameter values, providing good results in general for all data sources.

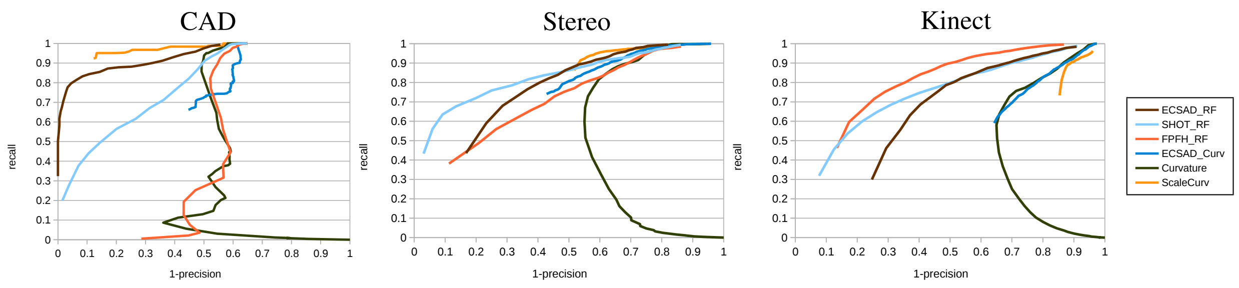
6.1 Quantitative Evaluation of Edge Detectors

For the purpose of evaluating the strength of our edge detector, we have created test data in a similar manner to the training data (see Figure 3). The test set was generated using two CAD models, two stereo scenes and two Kinect scenes, providing more than 6000 positive and 170000 negative test examples, respectively. We split the test set into three different categories (CAD, stereo and Kinect), as we have observed quite a varying performance across the different data sources. Note that the training set has not been split; only one training pass over the full training set is performed. All training and test data are publicly available on our web site.[3]

We train an RF classifier using our ECSAD descriptor. Additionally, we perform the same procedure using two recent shape descriptors, the Signature Histogram of Orientations (SHOT) (Tombari et al., 2010) and the Fast Point Feature Histogram (FPFH) (Rusu et al., 2009). Both features have been widely used for surface description.

In addition to the RF test, we evaluate the use of the internal curvature estimate produced by our descriptor for directly providing an edge confidence. For comparison, we also include in our test other curvature estimates, namely the total surface variation

[1]`http://pointclouds.org`
[2]`http://opencv.org`
[3]`https://sites.google.com/site/andersgb1/projects/3d-edge-detection`



ECSAD_RF	SHOT_RF	FPFH_RF	ECSAD_Curv	Curvature	ScaleCurv
9.23 s	18.17 s	16.0 s	7.66 s	9.02 s	18.8 s

Figure 6: Performance curves in terms of (1 - precision) vs. recall for the CAD (left), stereo (middle) and Kinect (right) test scenes. The bottom table shows the accumulated detection time for all six point clouds (in total ca. 176000 points) in the test dataset. For the learned detectors, this includes both descriptor computation and RF classification.

(Pauly et al., 2002) (termed *Curvature*) and a multi-scale extension of this algorithm (Pauly et al., 2003) (termed *ScaleCurv*). In these two algorithms, the curvature is estimated using the three eigenvalues of the scatter matrix of the supporting points around a point, simply by dividing the smallest eigenvalue by the sum of all three eigenvalues.

In Figure 6 we show results for all three data sources as (1 - precision) vs. recall curves, which is a standardized way to evaluate interest point detectors (Mikolajczyk and Schmid, 2005). For the ideal CAD models, which have noise-free edges, we observe a very high performance of the multi-scale curvature as edge confidence. Our learned detector comes close in performance, and shows a very high initial precision at low recall. For the real stereo and Kinect data, the performance of the curvature based detectors immediately drops, and the learned detectors become superior. For the Kinect data with the highest noise, the FPFH detector shows the best performance. Our learned detector shows comparable performance for all three data sources. In addition to this, our descriptor is computationally efficient–almost twice as fast as SHOT and FPFH. In addition, the SHOT descriptor has a dimension more than ten times higher than both FPFH and ECSAD.

6.2 Application: 3D Object Recognition

In order to assess the benefits of edge detection for another application, we applied a previously proposed point cloud registration algorithm to our features. The method is presented in (Buch et al., 2013b) and is based on RANSAC (Fischler and Bolles, 1981), with a crucial optimization step used for early rejection of point samples that are unlikely to produce valid pose hypotheses. We further improve the method by allowing for multiple feature matches within a predefined radius in descriptor space. The algorithm is presented below.

Initialization:

1. The object and scene surfaces are down-sampled to a voxel size of 3 mm to ensure a uniform point cloud resolution.
2. Edges are detected within both the object and scene point clouds using the learned RF detector, using ECSAD descriptors computed with a support radius of 15 mm. Non-maximum suppression is applied to reduce the number of features. The descriptors are stored for use below.
3. For each object edge feature, we use *k*-d trees to search for all matching feature descriptors in the scene within a radius of one unit in descriptor space.

Iterate:

1. Three random feature points are sampled on the object. For each of these points a random scene correspondence is retrieved from the list of correspondences generated in step 3 of the initialization.
2. Apply the pre-rejection of (Buch et al., 2013b): if any of the distances between the three object points differs more than 10 % from the equivalent distance between the corresponding scene points, continue to the next iteration.
3. A pose hypothesis is generated based on the three matches.
4. The pose is applied to the object point cloud, and we count the number of inliers supporting the pose by an Euclidean proximity threshold of 3 mm. Additionally, we require that the aligned

normal vectors have a relative angle less than $\pi/3$. If the number of inliers satisfying both these conditions is higher than 15 % of the number of object points, we break out and consider the object as recognized.

The pre-rejection step makes the search for valid poses very fast, so we run the algorithm for a maximum of 100000 iterations. In case all 100000 iterations are completed without finding a pose with more than 15 % inliers, the best pose is chosen to ensure recognition of highly occluded objects. Finally the determined pose is refined by ten iterations of the iterative closest point algorithm (Besl and McKay, 1992).

As an additional test, we also implemented our method with the full set of ECSAD features at all down-sampled surface points, not only at the edge features. We have tested our algorithms on the well-known laser scanner dataset by Mian et al. (Mian et al., 2006), consisting of four complete objects to be recognized in view-based 50 test scenes.[4] For comparison, we present previous results for three state of the art methods: Spin images by Johnson and Hebert (Johnson and Hebert, 1999), Tensor matching by Mian et al. (Mian et al., 2006), and finally the PPF registration by Drost et al. (Drost et al., 2010). The results are presented as occlusion vs. recognition rate, similarly to how (Mian et al., 2006) evaluated the original algorithms on the dataset. Occlusion is the percentage of the object which is visible, and recognition rate is the relative number of times an object is recognized in the 50 scenes. An object pose is accepted if it diverges with less than 12° and 5 mm from the ground truth pose, which is similar to the criterion used in (Drost et al., 2010).

For our surface-based method, we see a high performance, which indicates a high performance of the registration algorithm. The edge features, being more discriminative, show an even higher performance, giving the best recognition results at the highest occlusion rates. Additionally, we report the average recognition time per object, which for our algorithm includes both ECSAD computation, edge detection by the classifier and non-maximum suppression. These numbers clearly show the gain of using our sparse edge representation, giving a significant speedup relative to both the surface-based registration algorithm and the fast PPF registration.

[4] `http://www.csse.uwa.edu.au/~ajmal/recognition.html`

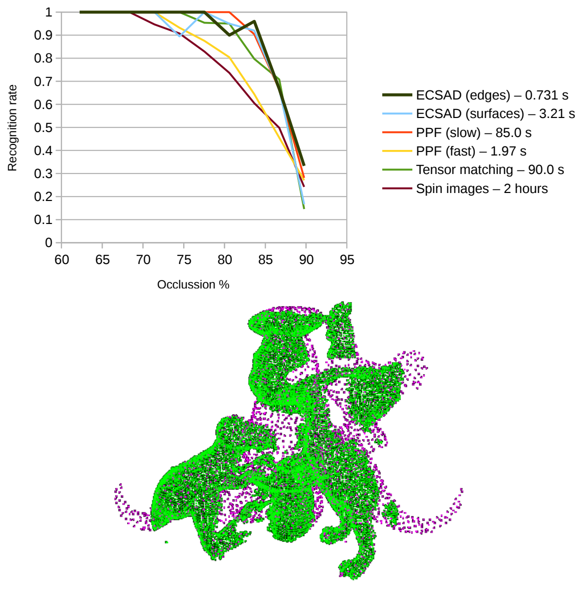


Figure 7: Comparison of our surface- and edge-based recognition systems with other works. The upper figure shows different pose estimation algorithm performances in terms of recognition-occlusion curves along with average running times per object. The bottom figure shows the first scene of the dataset, also shown in Figure 1. The magenta objects are the determined poses, so in this scene all objects have been correctly recognized.

7 CONCLUSION AND FUTURE WORK

A new edge detection approach for 3D point clouds from various sources has been developed, focusing on speed and overall performance. In these aspects our detector shows superior performance compared to other methods, even with limited parameter tuning.

A RANSAC based pose estimation algorithm was developed, which shows that using edges can significantly improve the runtime of 3D recognition algorithms. Furthermore the simple pose estimation application matches the performance of state of the art recognition systems on an established laser scanner benchmark, while being significantly faster.

In future work, a robustness study of the local reference frame compared with other reference frame estimation algorithms would be highly interesting. Since the descriptor has the best performance for a relatively small support radius, it would be interesting to apply the edges in higher level descriptors to determine if such an approach can result in a higher match rate for large noisy scenes. It would also be interesting to investigate the performance of the descriptor if it was used in a Hough-like voting algorithm instead of a RANSAC based approach. It is doubtful that this will increase the speed for the tested recognition dataset, but it may improve the recognition rate in more complex scenarios, where segmentation is often performed. In this context it would also be interesting to investigate if the edges can be used in a point cloud segmentation algorithm.

ACKNOWLEDGMENTS

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

REFERENCES

Bähnisch, C., Stelldinger, P., and Köthe, U. (2009). Fast and accurate 3d edge detection for surface reconstruction. In *Pattern Recognition*, pages 111–120. Springer.

Besl, P. and McKay, N. D. (1992). A method for registration of 3-d shapes. *TPAMI*, 14(2):239–256.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Buch, A. G., Jessen, J. B., Kraft, D., Savarimuthu, T. R., and Krüger, N. (2013a). Extended 3d line segments from rgb-d data for pose estimation. In *Image Analysis*, pages 54–65. Springer.

Buch, A. G., Kraft, D., Kamarainen, J.-K., Petersen, H. G., and Kruger, N. (2013b). Pose estimation using local structure-specific shape and appearance context. In *ICRA*, pages 2080–2087.

Canny, J. (1986). A computational approach to edge detection. *TPAMI*, PAMI-8(6):679–698.

Choi, C., Trevor, A. J., and Christensen, H. I. (2013). Rgb-d edge detection and edge-based registration. In *IROS*, pages 1568–1575.

Drost, B., Ulrich, M., Navab, N., and Ilic, S. (2010). Model globally, match locally: Efficient and robust 3d object recognition. In *CVPR*, pages 998–1005.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Frome, A., Huber, D., Kolluri, R., Bülow, T., and Malik, J. (2004). Recognizing objects in range data using regional point descriptors. In *ECCV*, pages 224–237.

Gumhold, S., Wang, X., and MacLeod, R. (2001). Feature extraction from point clouds. In *International Meshing Roundtable*.

Guy, G. and Medioni, G. (1997). Inference of surfaces, 3d curves, and junctions from sparse, noisy, 3d data. *TPAMI*, 19(11):1265–1277.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1992). Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph.*, 26(2):71–78.

Johnson, A. E. and Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3d scenes. *TPAMI*, 21(5):433–449.

Lai, K., Bo, L., Ren, X., and Fox, D. (2011). A large-scale hierarchical multi-view rgb-d object dataset. In *ICRA*, pages 1817–1824.

Mian, A. S., Bennamoun, M., and Owens, R. (2006). Three-dimensional model-based object recognition and segmentation in cluttered scenes. *TPAMI*, 28(10):1584–1601.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *TPAMI*, 27(10):1615–1630.

Monga, O., Deriche, R., and Rocchisani, J.-M. (1991). 3d edge detection using recursive filtering: application to scanner images. *CVGIP: Image Understanding*, 53(1):76–87.

Pauly, M., Gross, M., and Kobbelt, L. P. (2002). Efficient simplification of point-sampled surfaces. In *IEEE Conference on Visualization*, pages 163–170.

Pauly, M., Keiser, R., and Gross, M. (2003). Multi-scale feature extraction on point-sampled surfaces. In *Computer Graphics Forum*, volume 22, pages 281–289.

Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. In *ICRA*, pages 3212–3217.

Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *ICRA*, pages 1–4.

Tombari, F., Salti, S., and Di Stefano, L. (2010). Unique signatures of histograms for local surface description. In *ECCV*, pages 356–369.

Multi-View Object Instance Recognition in an Industrial Context

Wail Mustafa*, Nicolas Pugeault†, Anders G. Buch* and Norbert Krüger*
*Mærsk Mc-Kinney Møller Institute University of Southern Denmark,
Campusvej 55, DK-5230 Odense M, Denmark
Email: wail@mmmi.sdu.dk
†Centre for Vision, Speech and Signal Processing, Faculty of Engineering & Physical Sciences, University of Surrey,
Guildford GU2 7XH, United Kingdom

***Abstract*—We present a fast object recognition system coding shape by viewpoint invariant geometric relations and appearance information. In our advanced industrial work-cell, the system can observe the work space of the robot by three pairs of Kinect and stereo cameras allowing for reliable and complete object information. From these sensors, we derive global viewpoint invariant shape features and robust color features making use of color normalization techniques.**

We show that in such a set-up, our system can achieve high performance already with a very low number of training samples, which is crucial for user acceptance and that the use of multiple views is crucial for performance. This indicates that our approach can be used in controlled but realistic industrial contexts that require—besides high reliability—fast processing and an intuitive and easy use at the end-user side.

I. Introduction

The task of object recognition in an industrial assembly set-up (as shown e.g., in Fig. 1) is fundamentally different from the 'general object recognition problem' from 2D images as addressed, for instance, in the Pascal Challenge [1]. It also differs largely from object recognition problems posed by 3D datasets such as [2], which have been in particular recently discussed with the availability of cheap RGBD sensors, such as the Kinect camera[1]. The main difference for an industrial set-up is that the sensors and the number thereof can be chosen freely as well as the fact that illumination can be controlled to a large degree by the light sources on the platform or alternatively, that color can be calibrated by color normalization techniques. Such a difference has not been exploited by object recognition systems used in industrial set-ups such as [3]. A particular challenge is that the goal of performing large sequences of actions in assembly processes requires very reliable and also fast object recognition and localization as well as intuitive use at the end-user side.

In this paper, we address the task of object recognition in a well-controlled scenario assuming objects occurring only in the rather restricted work space of the robot as shown in Fig. 1b. The set-up resembles an 'intelligent work-cell' in an advanced production scenario. The task at hand is to determine the presence of objects in the working space covered by three pairs of Kinect and stereo cameras. In contrast to the object recognition problem addressed by standard databases operating on 2D images or 3D depth information extracted from individual views, in our set-up we can operate on rather complete 3D data computed by three different views arranged in a triangle (see Fig. 1a). Our method is then supposed to be used to trigger other mechanisms such as pose estimation or manipulation actions (e.g., grasping, peg-in-hole, or screwing actions) as well as monitoring such processes in the context of complex assembly operations (see, e.g., [4]).

In this paper, 3D *texlets* (see Subsect. III-D) serve as basic visual representations of objects. These texlets are acquired by two different sensors—stereo and Kinect cameras—simultaneously. From these, viewpoint invariant representations based on appearance and geometric relations are computed. The use of 3D information is attractive since it allows us to extract viewpoint invariant features in terms of geometric relations (such as distance or angle) between 3D entities. The fact that we operate in a limited and controlled workspace leads to reliable 3D shape and appearance information. In our representations, both aspects—shape and color—are represented separately, allowing us to investigate their relative importance. This space of feature relations (in the following also called 'relational space') can be expressed in (potentially high-dimensional) histograms providing unique and interpretable descriptors for specific objects (e.g., the distance between two parallel surfaces, see Subsect. III-E) which, besides being viewpoint invariant, is also rather specific for a certain object. As we will show in this paper, this is useful for efficient learning because a relatively few object recordings are required to learn representations for reliable object recognition.

As a classification algorithm, multi-class Random Forest [5], [6] is applied in this paper. Random Forests (RFs) have been found to be efficient because they combine the simplicity of decision trees with the stability of voting methods. The algorithm is trained with a set of real objects represented by a combination of their relations and appearance histograms (see Subsect. III-H).

The main achievements of our work can be summarized as follows:

- we demonstrate the potential of applying 3D viewpoint invariant relations by achieving high-performance clas-

[1]http://www.xbox.com/kinect

sification with very few training samples. The remaining misclassifications are caused mainly by object pairs with very fine shape and color differences. For these objects, the sensor resolution simply does not allow for the required precision for coding the shape and color differences.

- we can achieve a significant improvement in performance by using multiple cameras comparing to single—or even two—cameras. This is due to the fact that significant aspects of objects are expressed in our representation by relations, which only manifest themselves with a rather complete 3D representation only achievable by means of three views from different perspectives.
- we show that our approach, when applied to Kinect sensor data, has a much better performance in comparison with the sensor data extracted by standard stereo cameras.
- we show that, even under varying illumination conditions, it is possible to derive strong appearance features from color information when a color normalization step prior to the classification is performed.
- we show that the combination of color and shape information leads to higher recognition results, hence both features are complementary.
- we show how our approach can be used as a trigger for pose estimation and by that complex scene description in terms of object identity and object pose can be computed.

This work is based on a representation introduced in a conference paper [7]. In this journal paper, we however go significantly beyond the work in [7] in multiple respects: *(i)* we apply our approach to a larger and significantly more difficult object set[2]; *(ii)* we investigate the representation in terms of two crucial parameters connected to binning and smoothing; *(iii)* we investigate the effect of color normalization; *(iv)* instead of using only 1D and 2D histograms, we also make use of higher dimensional representations; and *(v)* we combine our representation with a pose estimation step allowing for a complete description of complex scenes in terms of object identity and pose.

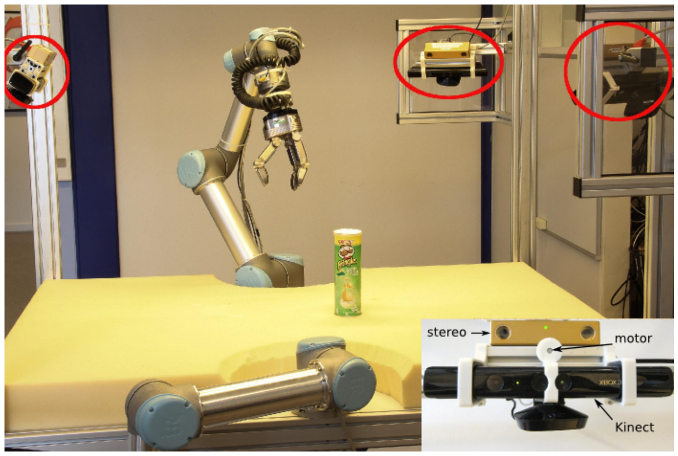
This paper is organized as follows: Sect. II discusses the state of the art. Sect. III describes in details the object recognition system introduced in this paper. In Sect. IV, we present a benchmark dataset, describe the experiments performed on the system and show the results. Sect. V presents an application scenario in which the object recognition system is used to trigger a pose estimation task and by that allow for the interpretation of complex scenes. Finally, a conclusion is given in Sect. VI

II. State of the Art

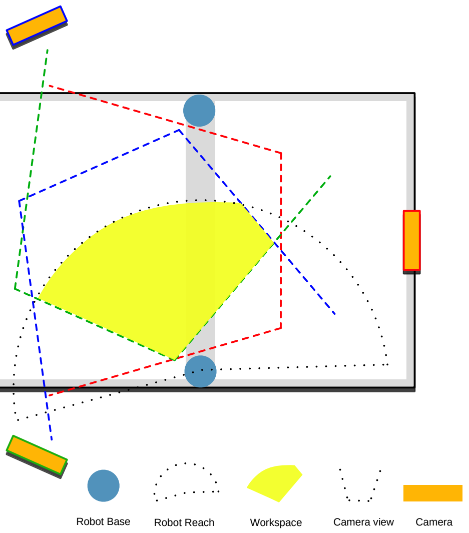
We first discuss the state of the art of the general problem of object recognition and then we focus on this problem in an industrial context.

Object recognition and classification learning: The problem of object recognition and classification has been intensively studied over the last decades. The annual Pascal challenges (see, e.g., [1], [8]) promote rigorous evaluation and comparison of object recognition algorithms. Although significant successes have been reported, criticism has been raised that the typical visual class recognition may learn pose and context-specific features rather than the object itself. In that context, notably Nicolas Pinto and colleagues showed that a simple model of the V1 cortical area of the human brain could perform well on a typical natural image benchmark [9]. Also, the generalization of classifiers or detectors learned on a specific dataset to another, called domain adaptation, remains a challenge [10]. In contrast to those works, this article is not concerned with the detection and recognition of objects "in the wild", but rather with the reliable and fast recognition of objects in a specific industrial set-up, for which we however cannot assume consistency in context. Also the number of training examples is supposed to be kept very low since the

[2]This data set is available at http://caro.sdu.dk/sdu-dataset (we will make it available by the final submission of this manuscript)



(a)



(b)

Fig. 1. The set-up. (a) overview of the set-up showing the robot arms and the camera pairs. The close-up view shows one pair of stereo and Kinect (with a vibration-inducing motor attached to it). (b) a top-view sketch for the set-up depicting the workspace.

requirement of recording a large training set would increase the complexity of the application of such a system at the end-user side.

Classification is typically done in two steps, feature extraction and classification, where the first step extracts or learns a set of features or parts to describe the objects' training samples and a second step which associate an object class to a new unseen object sample. The features used typically describe local image patches, often chosen for robustness to affine transformation, e.g., SIFT [11]. Alternatively, feature descriptors based on relative shape information, called 'shape context' were proposed by Belongie et al. [12]. The shape context of a point encodes the relative distribution of other points on the shape. It has been used as such to perform point-to-point matching in 2D. In [13], the shape context was extended to 3D and defined for a local neighborhood.

After feature extraction, classification can be done in two ways: the first class of methods effectively performs image *retrieval* and is based on nearest-neighbor matching (e.g., [11]); the second makes use of discriminative classification algorithms (such as Support Vector Machines [14] or Boosting [15]). Generally, discriminative approaches lead to higher classification performance, but can suffer from poor generalization when using weak visual features or when the variety of the training data is too limited.

Recently, hierarchical approaches such as convolutional networks have shown high performance (at the price of a significant computational cost) on such large dataset as ImageNet [16]. Interestingly, it was shown that the hierarchies learned on this dataset could then perform well when applied on a different dataset [17], offering some hope for solving the domain adaptation problem. It is worth noting that these results are based on very large training data and obtained at a significant computational cost. Both the computation time and the necessity to create large training data cause significant hurdles for the application of such systems in an industrial context.

The approach used in this work differs in particular in two aspects from the approaches to object recognition discussed above: First, the system is based on a multi-view set-up that is specific to an industrial scenario, aiming at high level recognition performance; Second, this set-up allows us to develop a feature describing the objects' 3D-shape in a pose-invariant fashion allowing the robust use of discriminative classification methods. As a consequence, a small amount of training data is required to achieve good performance.

Object recognition in industrial setup: Object recognition has been used in industrial production set-ups mainly for the identification of a small set of objects (mostly less than five objects) and is in general used as a prior step for pose estimation. Such approaches are nowadays part of standard vision softwares such as Cognex[3], Scorpion Vision[4] and Matrox[5]. These systems mostly provide 2D approaches. This necessarily leads to a larger complexity in using these systems, since the projective map need to be accounted for in the set-up of cameras. This requires covering all possible viewpoint and appearance changes by the training set as well as handling quite a number of parameters in the software that need to be adapted. Recently, also approaches using 3D data have evolved [3], but such approaches have not, to our knowledge, been used on industrial vision systems for object recognition tasks.

Although vision gradually enters production units, statements from end-users and even robot integrators such as ”vision does not work” are not uncommon. Such statements are usually caused by the fact that the use of the applied vision software requires at least some expert knowledge about the involved visual processes and the camera geometry. As argued above, the use of 3D vision approaches—as done in our work—can facilitate the application of vision algorithms in industrial scenarios by reducing the complexity introduced by viewpoint changes caused by the projective map, or in other words, by allowing the end-user to operate in the more intuitive Euclidean space.

Another advantage in an industrial context compared to the general object recognition problem discussed above is that the actual camera set-up can be freely chosen. This opens the possibility to increase robustness by using multiple cameras. In addition, due to relatively short distances between camera and object, 3D sensors such as Kinect like cameras can be used. The novelty of our approach lies in the explicit use of multiple simultaneously recorded views, utilizing viewpoint invariant relations that can only be generated based on the combination of all three views.

Another aspect of our approach is that due to the pose invariant representation only few training examples are required to achieve a high recognition performance. This facilitates the often quite sophisticated training that is in general required for view based systems (see, e.g., [2]).

In Subsect. IV-D, we will show that we can achieve with very few training samples high object recognition performance in a controlled—but, from a point of view of industrial production, realistic—environment for a recognition task which is much harder than it usually occurs in an industrial setting. This allows systems to perform object recognition for assembly processes with some complexity in an industrial context based on visual information.

III. Object Instance Recognition System

In this section, we describe in details the components of the object recognition system introduced in this article.

A. System overview

Fig. 2 shows the system components. The system operates on the robot platform described in Subsect. III-B in which three views are captured by three sensors (Kinect or stereo). For a single view, the process starts with applying colorimetric camera calibration on images as explained in Subsect. III-C. This is followed by scene preprocessing for table removal and object segmentation as a prior step for object recognition.

[3]http://www.cognex.com

[4]http://scorpionvision.co.uk

[5]http://www.matrox.com

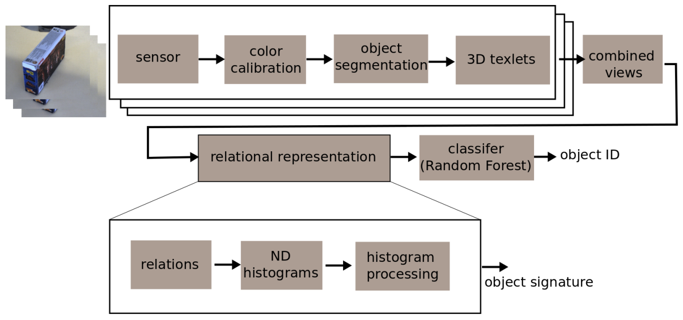


Fig. 2. Object instance recognition: block diagram of the different components. The three layers on top shows the components that process the sensory data from each single camera. The lower components are the ones process the 3D combined data where the relational representation of objects is obtained to form the object signature, which is then passed to the supervised classification algorithm.

The table removal is applied using a RANSAC-based plane detection whereas the segmentation is performed using 3D Euclidean clustering. All this is performed in the 3D point cloud data using PCL library [18].

For a segmented object, the 3D texlets features described in Subsect. III-D are then extracted forming a single 3D view of the object. Using the relative camera transformations, which are estimated through external camera calibration, the three views are combined in the 3D space. These combined features form the 3D representation of objects from which the relational representation is computed.

The relational representation is a pose-invariant object description obtained by computing shape and color relations from pairs of 3D features, see Subsect. III-E for details. The different relations are then binned in multi-dimension (ND) histograms to form the object signature (Subsect. III-F). Optionally, histograms are processed by means of spatial filtering for noise reduction (Subsect. III-G). The resulting object signature is finally passed to a classifier; Random Forest (Subsect. III-H) is used here. During the training phase, which is performed in a supervised manner, a classification model of decision trees is created from the training data. The model is used to predict the object ID (with an associated conference value) during execution (i.e., prediction phase).

B. Multi-view sensors (set-up)

The environment in which we want to solve the object recognition task is a robot work-cell (which can be used e.g., in industrial assembly processes as in [19]). Fig. 1a shows an overview of the set-up and camera pairs in use. The work-cell consists of two robot arms performing manipulation tasks with a variety of objects. Three pairs of Kinect and stereo cameras are mounted in a close to equilateral triangular configuration. By combining the three views of this set-up, we obtain a complete (except for the surface in contact with table) representation of the objects' 3D shape. Note that a vibration-inducing motor is attached to each Kinect to reduce the interference effects occurring when multiple Kinects with overlapping views are simultaneously used [20].

Fig. 1b is a sketch (plan view) of the set-up, showing the field of view of each camera and the area of reach of the main robot arm. The yellow-shaded area depicts the workspace in which our system operates. The workspace is defined by the intersection of the three fields of view and the area of reach. The requirement that all cameras cover the area is strictly limiting the usable workspace. On the other hand, for complicated manipulation tasks, such as the ones this set-up is intended for, high performance of object recognition and pose estimation is needed. In this paper, we show that having multiple views enhances performance significantly by providing a complete 3D representation, which allows for encoding a rich set of relations unavailable from single views (e.g., opposite surfaces). Furthermore, such a multi-view approach also increases the system's robustness against occlusion.

C. Colorimetric camera calibration

One way to increase color robustness is to apply colorimetric camera calibration (see Fig. 3). By doing so, we minimize two effects causing instability of color features. First, the variation in illumination due to having different lighting conditions. The second effect is the variation in the color representation that may occur due to different sensors. On the system level, this process leads to a more robust object instance recognition based on color (see Subsect. IV-C).

Essentially, the process involves reading reference color values obtained from the image and do the correction based on their true values. These reference colors are presented in a color checker[6] lying within the sensor's field of view. The color checker contains 24 color patches representing natural and gray-scale colors, which was first introduced by [21].

The method implemented here consists of two steps [22]: color normalization and color transformation. The normalization step is applied to make sure that intensity values of the image falls within $[m, 255-m]$. Note that m, which is set to 10, is a margin added to the standard image range of $[0, 255]$ to lower the risk of exceeding that range after performing the color transformation.

From the original image I, the normalized image I_n is obtained by:

$$I_n = sI + t \tag{1}$$

where $s = (256-2m)/(w_o - b_o)$ and $t = m - b_o$, which are scaling and translation factors. w_o and b_o are the gray-scale values (averaged RGB values) of the reference white and black colors, which are also given by the color checker. Note that images are stored in matrices where the columns and the rows represent the pixels and their corresponding RGB values, respectively.

The next step is the color transformation by which the color-calibrated image I_c is obtained:

$$I_c = MI_n \tag{2}$$

[6]We use the standard x-rite color checker, see http://xritephoto.com/

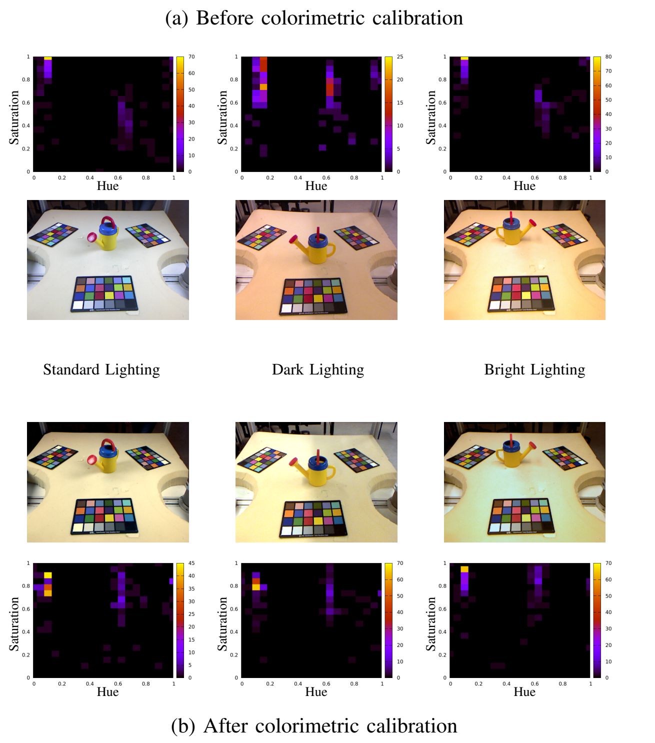


Fig. 3. Colorimetric camera calibration under varying lighting conditions. Object samples (images from one of the Kinect sensors) and their corresponding 2D histogram of hue and saturation are shown for: (a) before applying the calibration and (b) after. The x-rite color checker used to get reference color values is shown at the bottom of each image. This figure should be viewed in color.

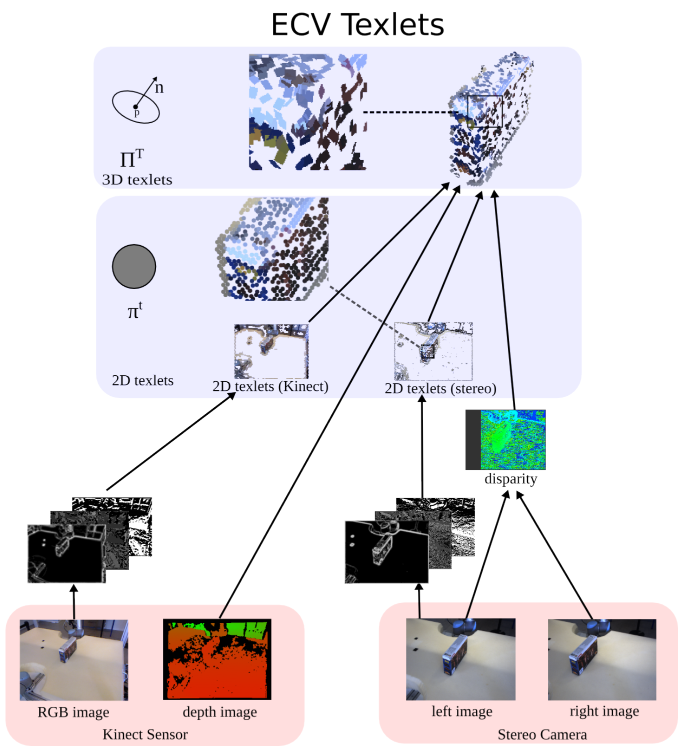


Fig. 4. The hierarchical representation of the texlets. Example images from Kinect and stereo cameras are shown at the bottom. In the middle, 2D texlets are extracted after filtering operations. On top are the extracted 3D texlets from different cameras. This figure is best viewed in color.

M is a 3x3 matrix calculated as the least-square solution of the transformation of the 24 reference color values (in RGB space) relative to their ground truth values.

D. 3D Texlets

As visual descriptors of objects, 3D texlets are extracted from both stereo and Kinect sensors. Fig. 4 shows the extraction process. 3D texlets (the top level in the figure) represent small, flat local surface patches in the Euclidean space. A 3D texlet is constructed by fitting a plane to a cloud of 3D points surrounding the 3D position that corresponds to the 2D position of a 2D texlet, which is a primitive feature extracted through local filtering of images [23]. The 3D reconstruction is performed using the depth image for Kinect and the dense disparity map (OpenCV implementation of the semi-global block matching algorithm [24]) for the stereo cameras. In the following, we provide a brief description of 3D texlets attributes used in this paper—for full description, the reader is referred to [23].

We define as $\mathcal{T}$ the space of all texlets and the 3D texlet, $\Pi_i^T \in \mathcal{T}$, is formalized as:

$$\Pi_i^T = (\mathbf{p}_i, \mathbf{n}_i, \mathbf{c}_i) \tag{3}$$

where the index i is used to identify the texlet Π_i^T, $\mathbf{p}_i$ is the texlet's 3D position and $\mathbf{n}_i$ its orientation (given by the normal vector). In addition to the above geometric attributes, the 3D texlets also encode color information in RGB format: $\mathbf{c}_i = (r_i, g_i, b_i)$. The number of the extracted 3D texlets depends on the kind of sensor used, the properties of the objects in the scene (mainly size, texture and reflectiveness). The extraction rate depends on the number of texlets and the use of parallel processing. When operated in GPU, 3D texlets extraction using Kinect can be achieved with approximately 5 Hz (frame per second) [23].

E. Relations

The 3D texlets introduced in the previous section provide absolute features (relative to an external reference frame) of objects in the 3D space. One limitation when representing shapes, with e.g., bags of features [14], is that this representation may vary drastically depending on viewpoint. For this reason, we propose to represent objects' shapes as distributions of *relations* between features, that are intrinsically pose-invariant. Pose-invariance is necessary for efficient learning of object classes.

Shape relations are similar to the 3D shape context introduced as local descriptors by [13], however, they are used here as global descriptors of objects. Having combined multiple 3D views of objects allows the global descriptors to be robust and rich representations for fast learning. We also use the term *color relations* to refer to color descriptors, which provide a more robust appearance descriptors compared to the absolute color. This section gives a detailed description of how the

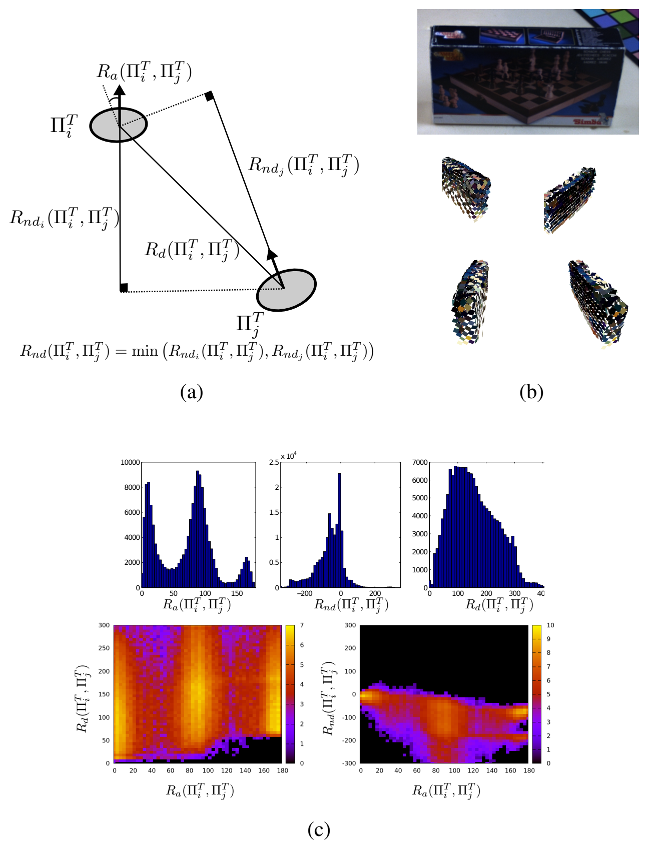


Fig. 5. Texlet's shape relations. (a) definition of shape relations between texlet Π_i^T and texlet Π_j^T; Euclidean distance $R_d(\Pi_i^T, \Pi_j^T)$, angle $R_a(\Pi_i^T, \Pi_j^T)$ and normal distance $R_{nd}(\Pi_i^T, \Pi_j^T)$. (b) example of an object and its extracted texlets as seen from different views (c) shape relation histograms of all pairs of texlets extracted from object shown in (b), 1D histograms on top and 2D histograms at the bottom.

different relations are computed.

To describe an object, we compute a set of relations from all pairs of texlets in the object. Formally, a pairwise relation $\mathcal{R}_k$ between texlets is defined as:

$$\mathcal{R}_k : \mathcal{T} \times \mathcal{T} \longrightarrow \mathbb{R} \tag{4}$$

Hence, a shape described by a set of N texlets $S = \{\Pi_1^T, \Pi_2^T, \ldots, \Pi_N^T\}$ will then be described by $N \times (N-1)$ values for a given relation. For convenience, we will note the set of those values as $\mathcal{R}_k(S) \in \mathbb{R}^{N\times(N-1)}$, where

$$\mathcal{R}_k(S) = \left\{R_x(\Pi_i^T, \Pi_j^T) : i, j \in [1, N], i \neq j\right\} \tag{5}$$

and $R_x(\Pi_i^T, \Pi_j^T)$ is the inter-texlet relation between Π_i^T and Π_j^T.

One important aspect is that the relation transforms an absolute pose-dependent representation in S into a relative pose-independent one in $\mathcal{R}_k$. For instance, the distance relation $\mathcal{R}_d$ transforms texlets' positions into inter-texlet distances. Because this kind of relations involves pairs of texlets, we refer to it as 'second-order' relations.

In the following, we describe all the texlets relations used in this paper.

Shape relations

The first class of relations that we will consider are *Shape relations*, which are defined to encode the objects' geometric information. This section introduces three shape relations used in this paper. Later, we will investigate which and how to combine those relations for best performance (see Subsect. IV-C).

It is important to note that for instance recognition, our shape representation should be *scale-variant*, i.e., object size matters and shall be encoded. Additionally, to characterize the different shape variations, we need to encode the deviation in orientation, i.e., curvature in a global context. Therefore, our set of relations shall address those two aspects. In this paper, we introduce the following relations(illustrated in Fig. 5a):

Angle relation: It is defined as the angle between the two texlets' normals.

$$R_a(\Pi_i^T, \Pi_j^T) = \measuredangle(\mathbf{n}_i, \mathbf{n}_j) \in [0°, 180°]$$

The angle relation is important to describe the shape variations. For instance, a flat surface will be dominated by 0° angle relations, whereas a sphere will have a set of relations that are uniformly distributed within the range $(0°, 180°]$.

Distance relation: It is defined as the Euclidean distance between two texlets in the 3D space.

$$R_d(\Pi_i^T, \Pi_j^T) = ||\mathbf{p}_i - \mathbf{p}_j||$$

The distance relation describes how texlets are distributed relative to each other. Note that, we don't apply scale normalization to keep the size encoded.

Normal distance relation: The normal distance relation is defined by:

$$R_{nd}(\Pi_i^T, \Pi_j^T) = \min\left(R_{nd_i}(\Pi_i^T, \Pi_j^T), R_{nd_j}(\Pi_i^T, \Pi_j^T)\right)$$

where

$$R_{nd_i}(\Pi_i^T, \Pi_j^T) = (\mathbf{p}_j - \mathbf{p}_i) \cdot \mathbf{n}_i$$

and

$$R_{nd_j}(\Pi_i^T, \Pi_j^T) = (\mathbf{p}_i - \mathbf{p}_j) \cdot \mathbf{n}_j$$

For an object with two parallel surfaces, the normal distance describes the distance between those surfaces, and therefore, explicitly encodes the object's dimensions. Additionally, this relation encodes whether two surfaces are pointing inward (toward each other) or outward; specifically, *positive* value for inward distance and *negative* value for outward. This allows for explicit characterization of certain object properties such as openness and closeness (see Subsect. III-F).

Note that, the two requirements of describing the geometric variation and being scale-variance can be well-fulfilled by combining the angle relation with either the Euclidean distance relation or the normal distance relations.

Color relations

The second class of relation describe the object's appearance using color. The color relations are computed from color channels of HSV and CIELAB (or Lab) spaces. Those two spaces are commonly used for color indexing [25] because they provide a color coding that is more stable under changing lighting conditions than RGB . They both separate the lighting information, luma, from the color information, chroma. More specifically, in HSV, the chroma is represented by the Hue (H) and the Saturation (S) whereas the luma is represented by the value (V). In CIELAB, the luma is the lighting (L) component and the chroma is the a and b components. This allows for the presentation of color with two values, when luma is undesired.

The inter-texlet relation of a certain color channel, c, is computed as:

$$R_c(\Pi_i^T, \Pi_j^T) = \langle c(\Pi_i^T), c(\Pi_j^T) \rangle$$

where the symbol $\langle\rangle$ denotes averaging operation. Using the average as color relation maintains the distinctiveness of color as a feature for objects with uniform colors whereas the difference of colors as used in, e.g., [26], would be close to zero. That would mean that homogeneously colored objects of different colors would not be distinguishable. Furthermore, averaging smooths out the noise and hence enhances the color robustness. In practice, experiments on our dataset showed that recognition performance was reduced by nearly 50% when using color difference rather than color average.

For the three color channels of HSV space, the average inter-texlet relations are referred to as $R_h(\Pi_i^T, \Pi_j^T)$, $R_s(\Pi_i^T, \Pi_j^T)$ and $R_v(\Pi_i^T, \Pi_j^T)$. For CIELAB, they are $R_l(\Pi_i^T, \Pi_j^T)$, $R_a(\Pi_i^T, \Pi_j^T)$ and $R_b(\Pi_i^T, \Pi_j^T)$.

The transformation from the RGB space, which is the default space in 3D texlets, is implemented using the standard formulae[7]. Note that $sRGB$ is the RGB standard used by the sensors, hence, the $sRGB$ corresponding white point reference is used to convert to CIELAB space.

F. Multi-dimensional histograms

In the previous section, we introduced a set of relations between texlets. Although individual texlets carry implicit information about objects' shape and appearance, overall statistics over different relations between texlets forming an object do provide pose-invariant and rich description of the objects. This statistical representation is implemented by binning relations in multi-dimensional histograms, which model their distributions as fixed-sized vectors.

For instance, angle and distance relations are mapped into a 2D histogram and color relations formed from the three color channels are mapped into a 3D histogram. In the following, we will show that such a representation of objects is naturally pose-invariant.

For a set of D relations, denoted as $V = \{\mathcal{R}_{x1}, .., \mathcal{R}_{xD}\}$, the D-dimensional histogram is defined as:

$$H(V, b) = \{h_1(V), ..., h_{b^D}(V)\} \qquad (6)$$

[7]See e.g.,http://brucelindbloom.com

where b is the number of bins that is, for simplicity, kept constant along all dimensions (relations) and $h_i(V)$ is the number of relations that fall jointly within the boundary of the ith bin. This means that the total number of the bins in this multi-dimensional histogram (i.e., the size of the corresponding feature vector) is equal to b^D. The optimal value of b is experimentally determined for all kind of relations in different ways of grouping (see Subsect. IV-C). All bins of the multi-dimensional histograms can then be used as a fixed size feature vector, f, describing the object's shape and appearance.

Fig. 6 shows 2D histograms for different objects. In this figure (a) and (b) represent the same box with two different poses and appearances; (c) is a similar box, but with an empty cavity in the front side; and (d) is a cylindrical box. First, note that the shape histograms in (a) and (b) are very similar, despite the object being in a different pose. This demonstrates the invariance of the relation statistics as a feature descriptor. For those two objects, the 2D histograms of shape (on the left) illustrate characteristics of the object's shape: the peak visible for normal distance of zero and angle of zero encodes all coplanar texlets. Then, two peaks are visible for angle of 180 degrees and normal distance of -150 and -270 that correspond to the parallel sides of the box. Finally, the area around 90 degrees correspond to orthogonal surfaces.

Second, in (c) we can see the representation for an open box. In this case, the color histogram (right) is similar to (b), but it also shows additional peaks for the inside color. In the shape histogram, we also see additional peaks illustrating the parallel surfaces from the inside and outside of the box. For the cylindric object in (d), the shape histogram as well the appearance histogram are significantly different from the rest.

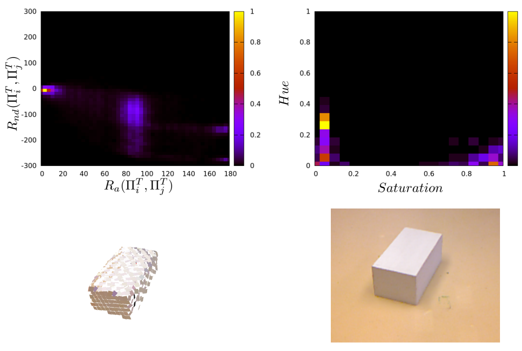
In summary, the above examples demonstrates three characteristics of the shape relations: pose-invariance, distinctiveness and interpretability.

G. Histogram processing

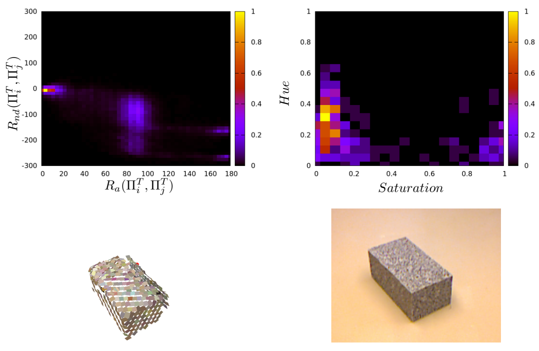
In previous sections, we showed how histograms of relations provide a rich description of objects. In such a high level representation, reducing noise will enhance recognition. The noise is a result of error propagated from lower processes such as 3D reconstruction, relative camera calibration, texlet sampling, uncompensated variation in color, and histogram binning.

We use Gaussian smoothing filter to perform noise reduction by convolving the histogram with a Gaussian (normal distribution) function. Gaussian filter is a low-pass filter that reduces the noise and only attenuates the high-frequency components because it does not have a sharp cut-off frequency. The filter is widely used in image processing applications (e.g., Canny edge detector [27]) where the information is contained in the high-frequency components. This also applies to our histograms – shape and color information are high-frequency. Let $\acute{H}(V, b)$ be the histogram after smoothing, which is computed by:

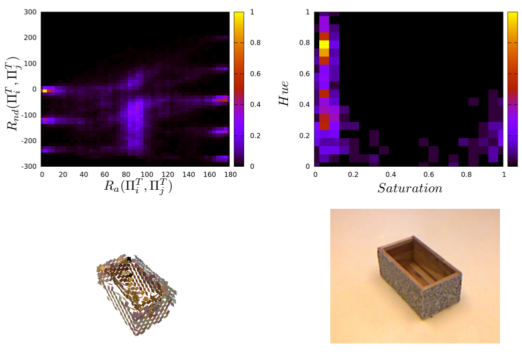
$$\acute{H}(V, b) = H(V, b) * K(\sigma) \qquad (7)$$



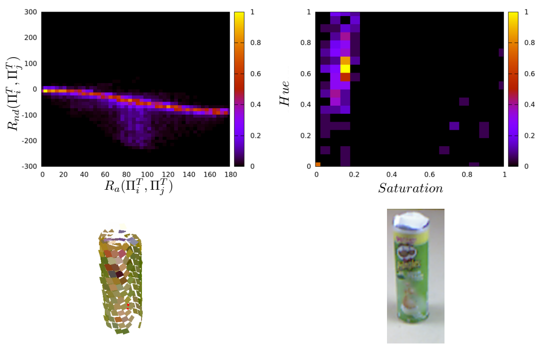
(a)



(b)

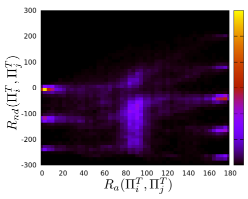


(c)

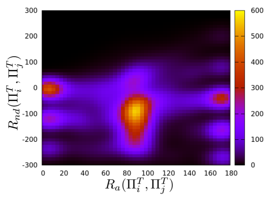


(d)

Fig. 6. Four different scene configurations and corresponding histograms. The histogram blocks for each scene consists of the following components: (top row, left) 2D histogram of angle $R_a(\Pi_i^T, \Pi_j^T)$ and normal distance $R_{nd}(\Pi_i^T, \Pi_j^T)$ for all possible pairs of texlets. (top row, right) 2D appearance histogram representing the hue (H) and the saturation color information. (bottom row, right) overview of the object in the scene. (bottom row, left) the extracted 3D texlets of the object.



(a)



(b)

Fig. 7. Histogram filtering. (a) the original histogram (b) the histogram after filtering with $\sigma = 2$.

where $K(\sigma)$ is a D-dimensional Gaussian kernel and σ is its standard deviation, which will be chosen empirically in Subsect. IV-C. In the implementation, we make use of the separability property of the Gaussian kernel. Therefore, the D-dimensional convolution is performed by a series of D consecutive convolutions using 1D kernel. The kernel size is determined using the 3-sigma rule ($k = 10/3(2\sigma - 1)$), which implies that the kernel covers 99.7% of the Gaussian function.

H. Classification (Random forests)

The quality and invariance properties of the histogram representation presented in the previous section makes it attractive for the purpose of object recognition. Supervised classification is a field that is well explored in machine learning (e.g., [28], [29]). In this work, we make use of Random Forest classification [5], [6]. The reasons for this choice are multiple: first, RF can be trained efficiently and are very fast at classification time, even for large input dimensions; second, RF are intrinsically multi-class allowing for an efficient learning in contrast to 1 vs. all approaches; third, RF have shown to reach very high level of performance on a variety of tasks (notably [30], [31]); finally, RFs effectively perform a form of dimension selection and which makes the resulting models interpretable.

Random forests learn a collection of randomized decision trees from different random subsets of the available training data, in a manner similar to *Bagging* [32].

Formally, if we consider a dataset $D = (f_j, y_j)_{j \in [1..|D|]}$, where f_j is an observation represented by the feature vector f presented in Subsect. III-F and $y_j \in [1..C]$ is a class label and $|D|$ denotes the number of samples in D, then we draw M random subsets $D_i \subset D$, $\forall i \in [1..M]$ from the data ($M = 100$ here) and train a population of M decision trees $P = T_{i, i \in [1..M]}$ such that T_i is trained from the subset D_i. Typically, the subsets D_i are drawn randomly such that $|D_i| = \gamma |D|$ (we used a common value of $\gamma = 0.5$).

From each subset D_i, we train a Randomized Classification Tree (RCT). RCT are binary trees, where each node n splits the input space (and thereby the dataset such that $D_l \cup D_r = D_n$) recursively in order to maximize class purity in all partitions and sending the samples that fall on each side of the partition to each child node. The recursion stops when a node receives too few samples to split ($|D_n| < 5$ here) or reaches a maximum depth ($\mathrm{depth}(n) > 10$ here)—such nodes are called leaf nodes and label the corresponding region according to the majority label in the available samples.

The split operation is traditionally done along a hyperplane, by applying a threshold operation to one input dimension. The randomization of the decision trees is done by selecting randomly a subset of input dimensions (computed to be the first integer less than $log_2|f| + 1$, [6]) for each non-leaf node and optimizing amongst those the dimension and threshold defining the split which minimizes all partitions' class impurity, using the so–called Gini coefficient $G(D_n)$:

$$G(D_n) = 1 - \sum_{k=1}^{C} \left(\sum_{j=1}^{|D_n|} \frac{\mathbf{I}_k(y_j)}{|D_n|} \right)^2 , \qquad (8)$$

where $\mathbf{I}_k(y_j)$ is an indicator function that returns 1 if $y_j = k$ and 0 otherwise.

Finally, the RF prediction $P(f)$ for an input vector is obtained by calculating the class with the largest amount of votes amongst all RCTs T_i, i.e.,:

$$P(f) = \arg \max_{k \in [1..C]} \sum_{i \in [1..M]} \mathbf{I}_k(T_i(f)) \qquad (9)$$

hlwhereas the associated *confidence* is computed as the ratio of the number of votes (of the predicted class) to the number of RCTs.

The hierarchical greedy search for splits allows for a high performance classification, while the randomization and redundancy provided by the bagging reduces the model's overfitting, increasing generalization and robustness.

IV. Dataset and Experiments

In this section, we present the benchmark dataset and the different experiments performed to evaluate the system. First, the multi-view object dataset is introduced in Subsect. IV-A. This is followed by describing how the experiments are set up in Subsect. IV-B. In Subsect. IV-C, we investigate how to form the optimal description of an object by separately considering the color and shape representations. Using multiple camera views versus single view is compared in Subsect. IV-D. In Subsect. IV-E, we show the performance obtained through Kinect data in comparison with stereo data. In Subsect. IV-F, we do error analysis by discussing the cases in which the system performs relatively low.

A. Multi-view object dataset

To benchmark our system, a dataset of 100 objects was created[8](see Fig. 8). The selection of objects cover a wide range including industrial and household objects, some of them taken from the KIT dataset [33]. The dataset contains RGBD and stereo images from the three Kinects and stereo pairs presented in Subsect. III-B, along with the relative transformations of the sensors (calibrated). For each sample, we extract 3D texlets (as discussed in Subsect. III-D) from all views. Texlets from different views are then combined (in 3D space) using the camera transformations. This allows for having a rather complete 3D visual representation of objects (see Fig. 5b).

There is a total of 30 different samples (random poses) for each object captured under three defined lighting conditions (see Fig. 2a): 'standard', 'dark' and 'bright' with 10 samples each. The variation in lightning is created to test the robustness of the system in light-changing conditions and to study the impact of the colorimetric calibration. Fig. 3 shows samples of the different lighting conditions. Objects were selected such that the set has objects with the same shape and different appearances and vice versa (see Fig. 8). The reason for this is to test the use of shape and color both individually and in combination.

B. Experiment setup

In the following experiments, unless otherwise specified, the 3D texlets from three Kinects (colorimetrically calibrated) are used. In each experiment, the dataset is divided into training and test subsets. The test subset is taken from one lighting condition (10 samples per object) whereas the training subset is taken from the other two (20 samples per object). The test set is used to evaluate the system performance in terms of recognition accuracy, which is defined as the trace mean of the confusion matrix.

To quantify the robustness of color information associated to the texlets, the experiment is executed in three different modes. For each mode, the test subset is taken from a different lighting condition and the experiment is iterated 5 times where the RF is differently seeded each time. This results in 15 iterations from which the average accuracy and the standard deviation are computed.

Here, we want to point out that a big advantage of our ~~our~~ approach is that good recognition performance is already possible with very few object instances stored (see Subsect. IV-D) due to the high degree of pose invariance of the representation as well as the color normalization procedure. This allows for a fast teaching of objects by putting them into the field of view of the camera system and record data for very few standard poses (e.g., two for a cylindric object corresponding to standing and lying). This fast teaching is particularly important in an industrial context.

[8]http://caro.sdu.dk/sdu-dataset (we will make it available by the final submission of this manuscript)

Fig. 8. Multi-view dataset of 100 objects shown in thumbnails. The set contains RGBD and stereo images: 30 samples each object from three camera views. The data were captured under three lighting conditions. The dataset is available at http://caro.sdu.dk/sdu-dataset (we will make it available by the final submission of this manuscript).

C. Optimal representation for color and shape

In this section, various experiments have been conducted to find the best combination of color and shape relations and to determine histogram and filtering settings. The process has been performed for color and shape relations separately, such that they can later be combined in one representation. Whenever color is involved, the use of the colorimetric calibration is also evaluated. The results presented in this section address the following aspects:

Set of relations: Here, we aim at selecting the best set of relations encoding shape and color, from the ones defined in Subsect. III-E.

Histogram binning: To determine the optimal bin size of histograms. For simplicity, the bin size is fixed across dimensions.

Filtering: To determine the value of σ (Gaussian filtering of histograms) that yields the best recognition.

Relational dimensionality: To determine the construction of relations into ND histograms, i.e., the best composition of the feature vector f defined in Subsect. III-E. For instance, three relations can be arranged in three 1D histograms, two 2D histograms or one 3D histogram. The optimal color and shape representations are separately determined by investigating the above aspects. The overall object representation is then defined by the combination of the two representations.

Color

Fig. 9 shows the classification accuracy over varying number of bins using different color relations derived from CIELAB space. Generally, the figure shows that the performance increases to a maximum value before it starts to decrease again. The decrease is steeper for histograms of higher dimensions – this is particularly clear for the 3D histogram. This can be interpreted as a result of data sparsity in feature space, which is exponentially proportional to the number of dimensions. Moreover, the higher the number of bins, the higher the number of features involved in learning (see Subsect. III-F). This makes learning slower and more prone to over-fitting.

From the figure, we find that the optimal number of bins is 10, 6 and 4 for 1D, 2D and 3D histograms, respectively.

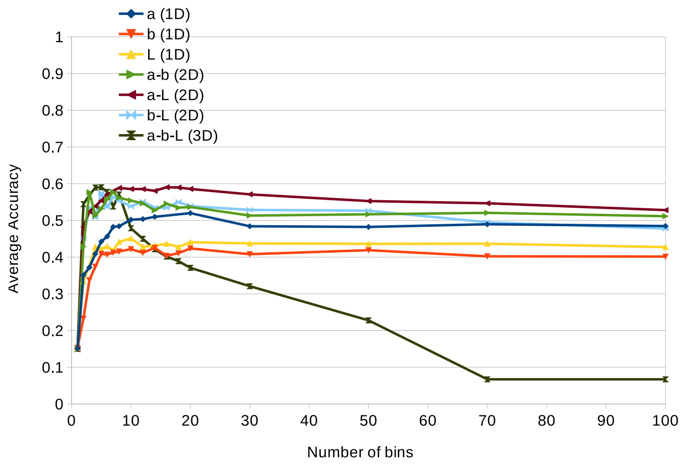


Fig. 9. Color histograms binning. The histogram binning of CIELAB color relations: a, b and L are 1D histograms of the color space components; a-b, a-L and b-L are 2D histograms; and a-b-L is a 3D histogram.

The number of bins corresponds to a resolution of 10% of the color space in 1D histograms, 16.7% in 2D histograms and 25% in 3D histograms.

Based on the optimal bin numbers, we investigated the effect of filtering under varying values of σ. We found that $\sigma = 1$ yields the highest performance. This value of σ corresponds to 5% of the color space in 1D histograms, 8% in 2D histograms and 12.5% in 3D histograms.

In Fig. 10, the HSV and CIELAB color spaces are compared in terms of the system classification accuracy when color relations are used. In this figure, color relations from the different components binned in ND histograms are shown. The figure also demonstrates the effect of the colorimetric calibration in each case. We can observe a significant improvement with calibration in all cases except for the L component of the CIELAB and the *value* component of the HSV. Although the Kinect sensor automatically performs exposure adjustment resulting in stabilizing luma components, which L and *value* represent, the result shows that luma is not a strong feature for recognition under changing lighting conditions. The figure

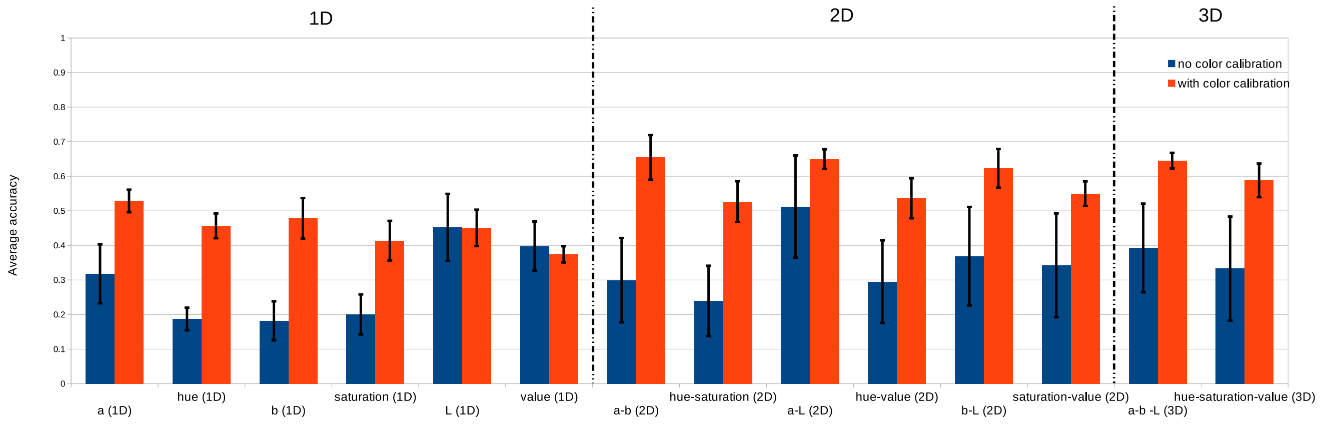


Fig. 10. CIELAB Vs HSV. a, b and L are 1D histograms of CIELAB components; a-b, a-L and b-L are 2D histograms; and a-b-L is a 3D histogram. Hue, Saturation and Value are 1D histograms of HSV components; Hue-Saturation, Hue-Value and Saturation-Value are 2D histograms; and Hue-Saturation-Value is a 3D histogram.

shows that, in all cases, the colorimetric calibration accounts for smaller standard deviation, i.e., higher stability. It also shows that CIELAB outperforms HSV as a color space when color is used for recognition.

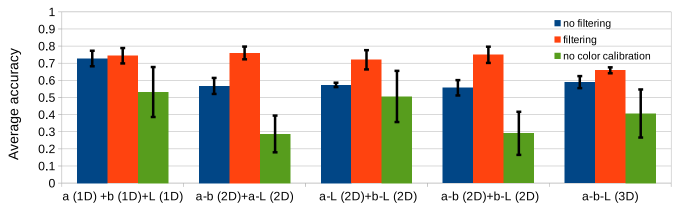


Fig. 11. Color relation dimensionality. a+b+L is the combined 1D histograms of CIELAB components; a-b+a-L, a-L+b-L and a-b+b-L are two combined 2D histograms each; and a-b-L is one 3D histogram

In Fig. 11, we show the classification accuracy when the three components of CIELAB arranged in different dimensionalities: three 1D histograms, two 2D histograms and one 3D histogram, hence it shows all the possible arrangements in which the three color components can be combined. In determining the overall color representation, we find that 1D histograms (1D histograms of *L* relations, *a* relations and *b* relations) slightly outperform the 2D histograms. Additionally, the figure also emphasizes the advantage of filtering and the colorimetric calibration.

Shape

Similar to color, we first aim at determining the optimal number of bins for the different shape relations discussed in Subsect. III-E as shown in Fig. 12. We can see the same pattern occurring: the performance reaches a maximum value before it starts to decrease and that it has a steeper slope for higher dimensions. From the figure, we find that the optimal number of bins is 50 for distance and 19 for angle in 1D histograms, 12 in 2D histograms and 8 in 3D histograms. The number of bins corresponds to a resolution of 2% of the shape relations

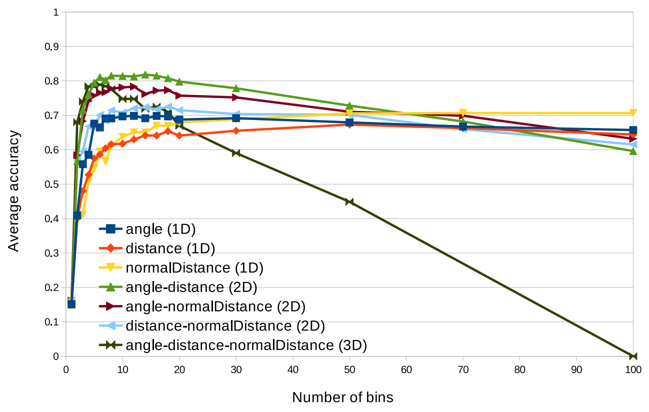


Fig. 12. Shape histograms binning. The histogram binning of the shape relations: Angle, Distance and NormalDistance are 1D histograms; Angle-Distance, Angle-NormalDistance and Distance-NormalDistance are 2D histograms; and Angle-Distance-NormalDistance is a 3D histogram.

spaces in 1D histograms, 8.3% in 2D histograms and 12.5% in 3D histograms. Note that the distance ranges from 0 to $300mm$ and the angle ranges from 0 to 180°.

Based on the optimal bin numbers, we investigated the effect of filtering under varying values of σ. We found that $\sigma = 0.5$ yields the highest performance. This value of σ corresponds to 1.2% of the shape spaces in 1D histograms, 2.1% in 2D histograms and 3% in 3D histograms.

When the shape relations are arranged in different dimensionalities, we found that the 2D histogram of angle and distance yields the best performance. Moreoever, as opposed to color, we found that filtering in shape relations does not achieve significant improvement.

Combined shape and color

Based on the above findings, we show how the system performs when the optimal color and shape representations are combined. This is demonstrated in Fig. 13 and also compared

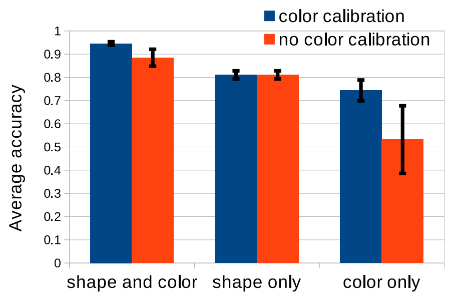


Fig. 13. The optimal representations of color and shape separately and combined.

with the separate representations of color and shape. The figure shows that a classification accuracy of 94% is achieved, which is significantly higher than 81% for shape alone and 74% for color alone. We can also see that the colorimetric calibration contributes with improving the accuracy as well as the stability (i.e., smaller standard deviation).

D. Performance depending on number of views and samples

In this experiment, we show the system performance in two aspects: the number of training samples per class and the number of camera views used to capture objects. For each experiment, the number of samples per class is fixed to 10 samples in the test subset and changed from 1 to 20 in the training subset. Fig. 14 shows three learning curves for three cases: three views, two views and one view. Note that, in contrary to previous experiments, for both subsets, the samples are randomly selected across all lighting conditions. This explains the slightly higher performance for three views with 20 samples per class in training (96% compared with 94% as in Fig. 13).

The figure highlights important features of the our system. First, the learning efficiency by which high performance is achieved with a few training samples. We can observe that with already 1 sample per class we get 60 % and above 90 % with 5 samples. Secondly, the figure shows the advantage of having multiple views. This is evident in terms of performance (about 17% improvement compared with one view, 5% compared with two views). It is also evident by obtaining faster learning with more views, i.e., less number of samples is needed to reach the 'steady-state' of accuracy (it is about 18, 15 and 10 for one, two and three views respectively).

E. Kinect vs stereo cameras

Fig. 15 shows the classification accuracy of the system when the stereo cameras are used compared with the Kinect sensors. The figure shows that the system performed significantly better with Kinect data (26% higher) on our dataset. Contrary to Kinect data extraction, dense stereo algorithms generally fails on non-textured objects. Having many objects in the dataset that fall in this category explains the lower performance of the stereo data. Non-textured objects are widely available

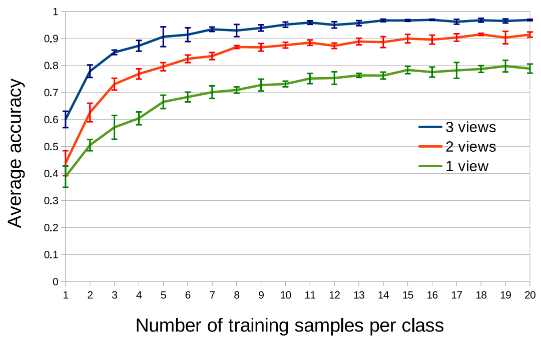


Fig. 14. Learning curve for object instance recognition. Three cases are compared depending on the number of camera views: singe view, two views and three views.

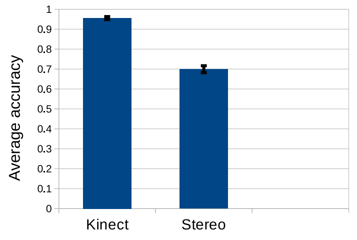


Fig. 15. System performance on stereo versus Kinect data.

especially in industrial platforms and that limits the reliability of the stereo sensory data.

F. Misclassification analysis

The maximum classification accuracy the system reaches is 96% (Fig. 14). In Fig. 16, the 3 objects with the lowest classification accuracy are presented together with their top confusing objects. The figure is derived from the average confusion matrices computed within the same experiment discussed in Subsect. IV-D.

The confused objects, as shown in the figure, are very similar in shape or color. The two objects on the left and the center are confused with objects that have the same shape, which suggests that the system fails to detect differences in their color representations. The object on the right is confused with an object with the same color and a only slightly different shape. Given that the two objects are relatively small, such geometric differences are beyond the limit of the sensor (Kinect in this case) to extract any distinctive 3D information.

The system accuracy discussed above considers only the RF top prediction, i.e., the prediction with the highest confidence (or the majority of tree votes). If we allow predictions with confidence values that are above a certain threshold, we will obtain, instead of single prediction, a list of recognition candidates per test instance. In order to find the accuracy limit the system can reach by possibly including a process capable of finding the correct prediction from this list. To do this, a test instance is considered correctly recognized if it is in the list. The confidence threshold is set to 25% allowing for a maximum of 4 candidates. By applying the same settings as in Subsect. IV-D, we reach an average accuracy of 99.76% with a standard deviation of 6×10^{-3}.

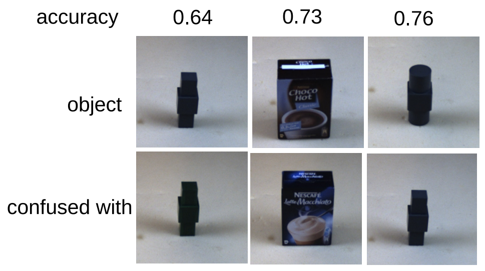


Fig. 16. The three least classified objects and the objects they are confused with. Note that the objects on the left have different color (top: dark gray, below: dark green). The thumbnails are resized for better visualization and they don't necessarily reflect their actual relative sizes.

V. Application for Pose Estimation

We have tested our system in the more application-oriented scenario of free-form recognition and full 6D pose estimation. In this application, we assume a typical tabletop setting where multiple objects are observed in a scene (see Fig. 18). The task is to perform instance recognition and pose estimation of all objects present for further processing, e.g. robotic manipulation. To facilitate the use of our representation in the recognition process, we assume spatial separability of the objects, which allows us to preprocess and segment the scene (see Subsect. III-A) and then recognize all the clusters. Note that such a separation is straightforward to achieve in an industrial setting, by, e.g., any feeders.

Our algorithm for recognition and pose estimation works as follows:

1) *Cluster recognition:* the RF classifier is now run on each of the clusters separately. If the classifier returns a prediction confidence below 0.25 for a cluster, the cluster is rejected as an unknown object.
2) *Nearest training instance search:* the RF classifier is designed in such a way that it generalizes over the training instances for identification of an object in novel views. The RF output of a cluster is thus the ID of the object producing the highest prediction confidence. For pose estimation, however, we wish to perform a 3D alignment between the identified object and the scene cluster. To this end, we do a search for the concrete three view training instance of the object showing the highest degree of similarity with the cluster and use this model to compute the relative pose. This information is not available in the RF output, so we perform a linear search within the training set for the nearest matching view using the global histogram descriptors.
3) *Pose estimation:* the recognized object is now aligned with the scene cluster using the identified training instance. Here we use an optimized RANSAC-based algorithm presented in our prior work [34]. This algorithm injects a prerejection step to quickly discard samples that are unlikely to produce a correct alignment, making the search for the pose much faster. The best pose is determined by the number supporting inliers, given by the number of aligned object points that lie within 5 mm of the nearest scene point. The output pose of the RANSAC algorithm is finally refined using the ICP algorithm [35] to get a more accurate pose.

The above procedure is repeated for all clusters in the scene for which the RF classifier returns a high enough confidence. A block diagram is shown in Fig. 17 and the procedure is used as a direct addition to the recognition procedure in Fig. 2. The whole pose estimation process for each object, including pose refinement, takes on average less than 500 ms, due to the prerejective nature of the modified RANSAC algorithm.

In Fig. 18, we show pose estimation results for several different scenes of varying difficulty. During these tests, we experienced a very high amount of accuracy, as long as the objects were clearly visible in the scene

VI. Conclusion

We presented an object instance recognition system for an industrial work-cell with multiple vision sensors. Our system represents objects with viewpoint invariant 3D shape features as well as robust color features. The system was evaluated on a dataset of 100 objects recorded under three lighting conditions.

The results show that our system is able to achieve high performance (in terms of classification accuracy) with a few training samples. The results also shows that the system performance using multi-view representation of objects, i.e., combined representations of multiple cameras, is significantly higher compared to single view. Regarding color encoding, the result shows that color normalization, which aims at compensating for variation in lighting, enhances the performance. Therefore, the use of multi-view object representation for shape combined with applying color normalization is crucial for a reliable recognition system operating in this set-up. This high reliability allows for using the system to trigger other processes such as pose estimation, which we have also demonstrated in several complex scenes.

Acknowledgment

This work has been funded by the EU project Xperience (FP7-ICT-270273) and the project CARMEN funded by the The Danish Council for Strategic Research.

References

[1] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2009 (VOC2009)," Summary presentation at the 2009 PASCAL VOC workshop, 10 2009.
[2] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 1817–1824.
[3] A. M. Pinto, L. F. Rocha, and A. P. Moreira, "Object recognition using laser range finder and machine learning techniques," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 1, pp. 12 – 22, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584512000798

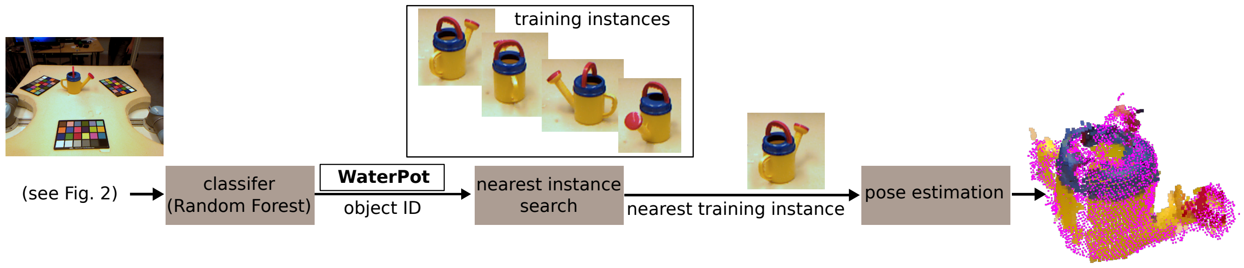


Fig. 17. Object instance recognition and pose estimation for a single object cluster: extended block diagram from Fig. 2, with the aligned training instance shown in purple. See text for details.

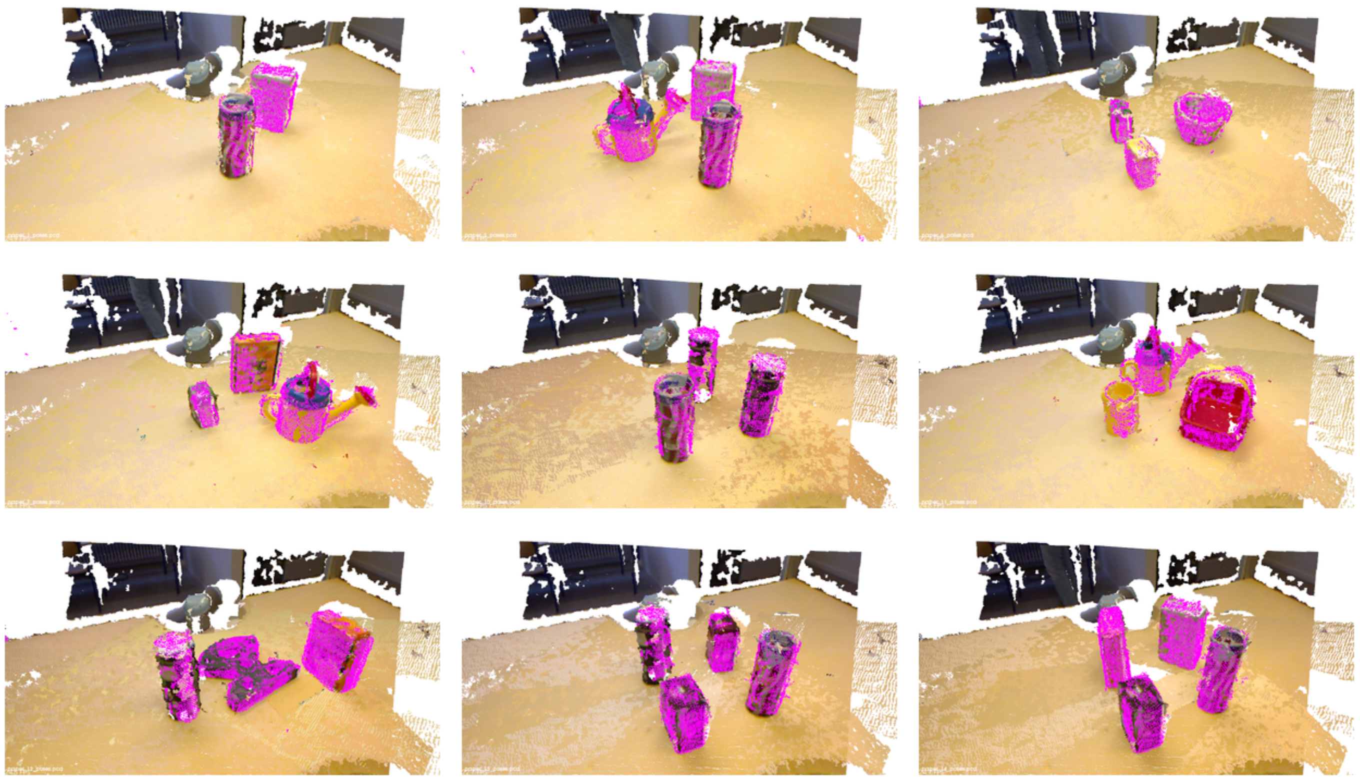


Fig. 18. Object instance recognition and pose estimation results, aligned models overlaid in purple. In all cases, all objects are correctly recognized.

[4] T. R. Savarimuthu, A. G. Buch, Y. Yang, W. Mustafa, S. Haller, J. Papon, D. M. nez, and E. E. Aksoy, "Manipulation monitoring and robot intervention in complex manipulation sequences," in *Workshop on Robotic Monitoring at the Robotics: Science and Systems Conference (RSS)*, 2014.

[5] Y. Amit and D. G. Y, "Shape quantization and recognition with randomized trees," *Neural Computation*, vol. 9, pp. 1545–1588, 1997.

[6] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[7] W. Mustafa, N. Pugeault, and N. Krüger, "Multi-view object recognition using view-point invariant shape relations and appearance information," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[8] M. Everingham, A. Zisserman, C. K. I. Williams, L. van Gool, M. Allan, C. M. Bishop, O. Chapelle, N. Dalal, T. Deselaers, G. Dorko, S. Duffner, J. Eichhorn, J. D. R. Farquhar, M. Fritz, C. Garcia, T. Griffiths, F. Jurie, D. Keysers, M. Koskela, J. Laaksonen, D. Larlus, B. Leibe, H. Meng, H. Ney, B. Schiele, C. Schmid, E. Seemann, J. S. Taylor, A. Storkey, S. Szedmak, B. Triggs, I. Ulusoy, V. Viitaniemi, and J. Zhang, "The 2005 PASCAL Visual Object Classes Challenge," in *Pascal Challenges Workshop*, ser. LNAI. Springer, 2006, vol. 3944, pp. 117–176.

[9] N. Pinto, J. J. DiCarlo, and D. D. Cox, "How far can you get with a modern face recognition test set using only simple features?" 2009.

[10] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 999–1006.

[11] D. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, 1999, pp. 1150 –1157 vol.2.

[12] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 4, pp. 509–522, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1109/34.993558

[13] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *Proceedings of the European Conference on Computer Vision (ECCV)*, May 2004.

[14] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *In Workshop on Statistical Learning in Computer Vision, ECCV*, 2004, pp. 1–22.

[15] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*

(CVPR'05) - Volume 2 - Volume 02, ser. CVPR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 994–1000. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2005.254

[16] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," in *International Conference on Learning Representations (ICLR 2014)*. CBLS, April 2014. [Online]. Available: http://openreview.net/document/d332e77d-459a-4af8-b3ed-55ba

[17] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," *arXiv preprint arXiv:1403.6382*, 2014.

[18] 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.

[19] B. Nemec, F. Abu-Dakka, J. Rytz, T. Savarimuthu, B. Ridge, N. Krger, H. Petersen, J. Jouffroy, and A. Ude, "Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile," in *Advanced Robotics (ICAR), 2013 16th International Conference*, 2013.

[20] A. Maimone and H. Fuchs, "Reducing interference between multiple structured light depth sensors using motion," in *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, March 2012, pp. 51–54.

[21] C. S. McCamy, H. Marcus, and J. Davidson, "A color-rendition chart," *J. App. Photog. Eng*, vol. 2, no. 3, pp. 95–99, 1976.

[22] Y. P. Touati, "Image color calibration using a color calibration rig," University of Southern Denmark, Tech. Rep., 2010.

[23] S. M. Olesen, S. Lyder, D. Kraft, N. Krüger, and J. B. Jessen, "Real-time extraction of surface patches with associated uncertainties by means of kinect cameras," *Journal of Real-Time Image Processing*, pp. 1–14, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11554-012-0261-x

[24] H. Hirschmller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *In Proc. CVRP*. IEEE Computer Society, 2005, pp. 807–814.

[25] M. Swain and D. Ballard, "Color indexing," *International Journal of Computer Vision (IJCV)*, vol. 7, no. 1, pp. 11–32, 1991.

[26] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek, "Evaluating color descriptors for object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1582–1596, 2010. [Online]. Available: http://www.science.uva.nl/research/publications/2010/vandeSandeTPAMI2010

[27] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 8, no. 6, pp. 679 – 698, 1986.

[28] C. Cortes and V. Vapnik, "Support-vector networks," in *Machine Learning*, 1995, pp. 273–297.

[29] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[30] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, june 2008, pp. 1 –8.

[31] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, june 2011, pp. 1297 –1304.

[32] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996. [Online]. Available: http://dx.doi.org/10.1023/A:1018054314350

[33] A. Kasper, Z. Xue, and R. Dillmann, "The kit object models database: An object model database for object recognition, localization and manipulation in service robotics," *International Journal of Robotics Research (IJRR)*, vol. 31, no. 8, pp. 927–934, 2012.

[34] A. G. Buch, D. Kraft, J.-K. Kamarainen, H. G. Petersen, and N. Kruger, "Pose estimation using local structure-specific shape and appearance context," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2080–2087.

[35] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, no. 2, pp. 239–256, Feb 1992.

Physical Interaction for Segmentation of Unknown Textured and Non-textured Rigid Objects

David Schiebener, Aleš Ude and Tamim Asfour

***Abstract*— We present an approach for autonomous interactive object segmentation by a humanoid robot. The visual segmentation of unknown objects in a complex scene is an important prerequisite for e.g. object learning or grasping, but extremely difficult to achieve through passive observation only. Our approach uses the manipulative capabilities of humanoid robots to induce motion on the object and thus integrates the robots manipulation and sensing capabilities to segment previously unknown objects. We show that this is possible without any human guidance or pre-programmed knowledge, and that the resulting motion allows for reliable and complete segmentation of new objects in an unknown and cluttered environment.**

We extend our previous work, which was restricted to textured objects, by devising new methods for the generation of object hypotheses and the estimation of their motion after being pushed by the robot. These methods are mainly based on the analysis of motion of color annotated 3D points obtained from stereo vision, and allow the segmentation of textured as well as non-textured rigid objects. In order to evaluate the quality of the obtained segmentations, they are used to train a simple object recognizer. The approach has been implemented and tested on the humanoid robot ARMAR-III, and the experimental results confirm its applicability on a wide variety of objects even in highly cluttered scenes.

I. INTRODUCTION AND RELATED WORK

The ability of a humanoid robot to adapt to situations that it has not explicitly been programmed for is crucial for its usefulness in future assistive tasks in human-centered environments. Many of these not-yet-experienced situations for a robot will arise due to the appearance of objects that it has not encountered before but now needs to deal with. In such situations, the robot needs to autonomously make itself familiar with these new objects. The first two crucial steps in this process, whatever outcome may be expected, are to locate and segment the new objects. Once they are segmented, a visual descriptor can be learned that allows later recognition, and essential information for grasping and manipulation is provided.

The focus of this work is to present our approach for the autonomous, interactive discovery and segmentation of textured and non-textured unknown objects in a cluttered environment by a humanoid robot. To demonstrate its usefulness, we use the obtained segmentations to learn visual descriptors of the new objects and show that they allow reliable recognition.

D. Schiebener and T. Asfour are with the Institute for Anthropomatics and Robotics, High Performance Humanoid Technologies Lab (H^2T), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. A. Ude is with the Humanoid and Cognitive Robotics Lab, Jožef Stefan Institute, Ljubljana, Slovenia.
schiebener@kit.edu, asfour@kit.edu, ales.ude@ijs.si

The segmentation of unknown objects from a complex unknown background has turned out to be very difficult, if not impossible, if a robot is restrained to passive observation. On the other hand, individual motion of an object is a strong cue that usually dissolves any visual ambiguities, manifests clear object borders and thus vastly facilitates segmentation. Usually, such helpful motion does not happen on its own when needed, therefore the robot has to create it itself. This fundamental idea has been pioneered by the authors of [1] who detect the sudden spread of optical flow from the hand of a robot when it touches and starts to move another object. The pushing motion is pre-programmed, and the obtained segmentation is not used for anything.

In [3], an articulated object is pushed to explore its kinematic properties, i.e. joints and solid parts, exploiting the observed relative 2D motion of local visual features. Again, the robot motion is pre-programmed. In [4], an object is pushed and segmented, which allows for the learning of a visual descriptor. Yet this approach is restricted to symmetric objects in simple scenes. [5] focuses on the singulation of individual objects from a pile by pushing them systematically, and [6] sorts colored bricks from clutter, strongly leveraging physical interaction for separating them. In [7] and [5] heuristics are proposed for systematically pushing clusters of objects in order to separate them.

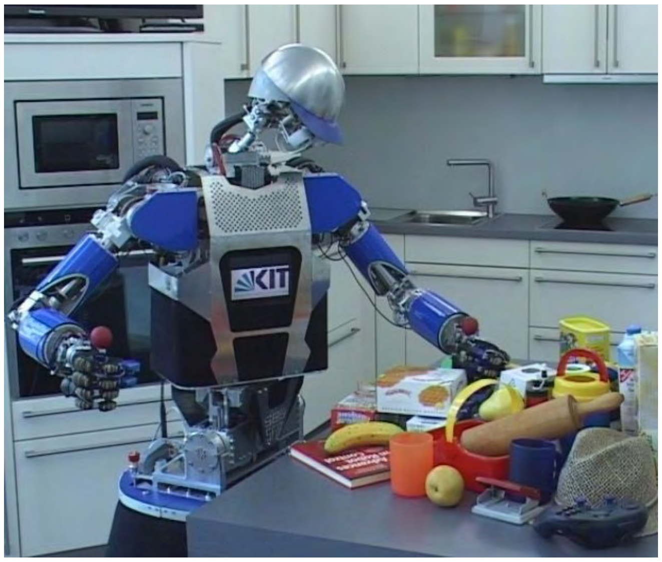


Fig. 1: Interactive object segmentation performed by the humanoid robot ARMAR-III [2]. By pushing unknown objects, they can be segmented from the environment based on the induced motion.

In our previous work (see [8], [9], [10], [11]) we used local visual features (SIFT[12] and later also color MSER[13]) to create initial object hypotheses. Those features are grouped based on their lying on a common regular geometric 3D structure (planes, later cylinders and spheres) as well as spatial proximity. One of these hypothetical objects is then pushed, and by observing the 3D motion of the local visual features, an object hypothesis can be verified by checking if it moves as a rigid body. This also permits to analyze each single local feature for concurrent motion and thus verify the individual features of the hypothesis. Other features that move consistently with the hypothesis are added and thus after two or three pushes a complete object segmentation in terms of the contained local features is achieved. We also demonstrated that this allows for the creation of object descriptors for recognition. In [14], we extended this concept by using the obtained object detection and segmentation to initialize a reactive grasping approach that enables the robot to grasp the formerly unknown object using tactile and haptic feedback without the need for a good 3D model for grasp planning.

While these results are very encouraging, our approach was always restricted to objects which have a sufficiently textured surface that offers enough distinctive local visual features to relocalize the object after it has been pushed. Most of the related approaches are also based on local visual features, with the exception of [15], where unicolored cylinder- and box-shaped objects are segmented interactively, tracking their edges in the image and depth data obtained from a Kinect sensor.

Based on the idea of interactive segmentation that we followed in our earlier work, we have now developed a different approach that enables the segmentation of textured as well as non-textured rigid objects, which we present in this paper. The only remaining restrictions are that the object can be moved by the robot, that it is not completely transparent or looks exactly like the background, and of course that it has an appropriate size with relation to the field of view and resolution of the cameras of the robot.

The following section will give an overview of our approach, which will be explained in detail in sections III and IV. In section V we present the results of our experiments on the humanoid robot ARMAR-III, and section VI concludes the paper.

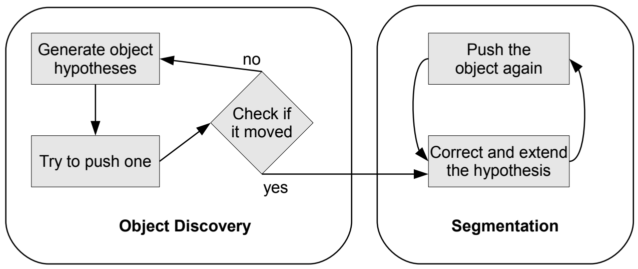


Fig. 2: System overview: Our approach can be divided into two main phases. First, the robot generates object hypotheses and tries to verify one of them by moving it. If an object has been discovered, the segmentation is improved and different views can be learned in the course of several further pushes.

II. PHYSICAL INTERACTION FOR SEGMENTATION

Physical interaction enables a humanoid robot to overcome the problems that usually arise if an unknown object is to be segmented in a complex scene that causes visual ambiguities. If the robot is e.g. confronted with a heap of unknown objects, there is probably no certain and infallible criterion to tell two objects apart that can be analyzed by observation only (at least none has been discovered yet). However, if an object moves, it can in principle be distinguished clearly from its environment.

To cause such helpful motion, the robot needs to induce it on the object somehow. The most simple and foolproof way to do so is to carefully push the object. This requires an idea about the existence and location of the object, which we can not take for granted when dealing with unknown objects in an unknown local environment. Consequently, the robot needs the ability to discover possible objects and estimate their location before being able to examine them. Our approach for generating object hypotheses is described in section III-A.

When such an object candidate has been pushed by the robot, there are two possible outcomes: If it moved, the robot can segment it, learn its appearance and examine it further. If it did not move, we have to assume that the robot did not actually push an object but something else that does not move. Thus, we implicitly define an ”object” as a physical entity that can be moved (and seen) by the robot. The problem of determining the motion of the object after it has been pushed is not trivial and has only been solved for special cases until now; we present our new and more general solution in section IV-A.

When the motion of the object has been determined, it can be exploited to acquire a complete and certain segmentation of the object in the camera image. We showed in our previous work [9] that if the object motion is known, it is simple to check for each local visual feature if it moved concurrently. But we do not want to rely on the existence of local features (i.e. texture), and we want an actual segmentation that tells for each pixel of the camera images whether it belongs to the object or not. Section IV-B describes how this is achieved.

III. INITIAL OBJECT DISCOVERY

A. Generation of object hypotheses

The first step in our approach for interactive segmentation is to create object hypotheses, i.e. to analyze the camera images of the robot for possible unknown objects. One of these hypotheses is then chosen for pushing and subsequent verification. A criterion for finding object candidates that has proven to be useful in our previous work is grouping of local features that lie on a common regular geometric structure like a plane, cylinder or sphere. Such a structure frequently indicates an underlying object. Another indication for promising candidates are unicolored regions of a

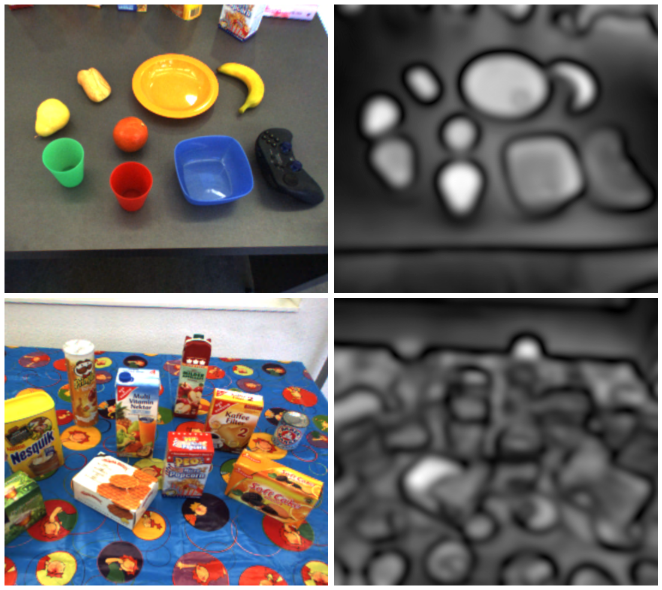


Fig. 3: A relatively simple and a confusing scene with their respective saliency images. As can be seen, the algorithm for saliency computation is not of much use in scenes where objects and background are equally rich in colors and contrasts.

size within the dimensions we would expect an object to have (about 5-50 cm in diameter). While these two criteria are certainly useful, we want to be able to detect objects independently of their appearance, therefore we complement these criteria with the generic concept of visual saliency.

Saliency is a bottom-up trigger for attention, a psychological concept that has been applied in computer vision to support other tasks by restraining the analysis of images to regions that ”stand out” in a certain respect (cf. [16] [17]). We use the saliency detector proposed in [18] to calculate a saliency map for the whole camera image. In that work, saliency is defined as the difference of an image region to its neighborhood, which is calculated at different scales using band-pass filters. The filters are realized using a Difference of Gaussian (DoG) filter $G(x, y, \sigma_j) - G(x, y, \sigma_k)$ with $\sigma_j > \sigma_k$. Summing up all edge images at different scales is equivalent to using a filter that is the sum of all filters, which can be simplified as follows:

$$\sum_{n=1}^{N} G(x, y, \sigma_n) - G(x, y, \sigma_{n+1}) = G(x, y, \sigma_1) - G(x, y, \sigma_N)$$

Thus the resulting saliency image is calculated as $S = |G(\sigma_1) * Img - G(\sigma_N) * Img|$, i.e. the difference of the image after being filtered with a Gaussian kernel with the lowest and highest desired standard deviation. This is done for all three color channels of the RGB image, and the results are added. We choose $\sigma_1 = 80px$ which limits the size of detected regions to a size that corresponds to the maximal extent we expect objects to have in the image, and $\sigma_N = 10px$ which smooths out the fine textures that are already accounted for by the hypotheses generation for textured objects.

The resulting salient image regions that are not yet occupied by object hypotheses from the first two criteria (unicolored regions, and local features lying on a regular geometric structure) are used to generate additional object hypotheses. In practice, the first two criteria covered most of the objects we tried, but for those which do not clearly fall into one of the two categories the saliency detection turned out to be a useful complement.

Figure 3 shows the saliency map calculated for different images. As can be seen, in simple scenes it does indeed yield the regions occupied by actual objects. In contrast, if the scene has a rather confusing background, the saliency detection is clearly overburdened and not helpful anymore. The two criteria based on local features and unicolored regions also return very many hypotheses in such a scene. In general, in a nontrivial image the separate use of all three criteria will usually yield a large number of initial object hypotheses.

This is not a fatal problem, as the robot could just systematically try all hypotheses, including those that result e.g. from the tablecloth. But it would save a lot of time to filter the hypotheses beforehand. As we are only interested in things that can be pushed, an additional criterion can be applied in order to keep only those hypotheses that seem to allow pushing. A simple heuristic for estimating if this is the case is to check whether a candidate object is higher than its direct neighborhood. We calculate a dense depth map from the stereo camera images of the robot using semi-global block matching (SGBM) [19]. The resulting 3D points are transformed into world coordinates. The camera image is subdivided into regular bins for which we calculate the average height of the contained points and compare them to their eight direct neighbor cells. Doing this at different scales and adding up the results, we obtain a map that gives a value for the relative local height of the image regions. This map is used to filter the object hypotheses and keep only those that lie in a region which is higher than its direct surroundings. Figure 4 illustrates this relative local height map and its effect on the hypothesis generation.

As we do not want to rely on the existence of local visual features, we use color and shape to describe the object hypotheses. To this end, we calculate a dense depth map from the stereo camera images and annotate the resulting 3D points with their color in the image. This kind of point cloud is usually referred to as RGBD (RGB+depth) data. After the initial object hypotheses have been generated, each one is represented by the RGBD points in the image region that it occupies. These point clouds will be used throughout the rest of the paper.

B. *Pushing for Verification*

One of the initial hypotheses is chosen to be pushed in order to verify that it is indeed an object and, in case of success, to segment it. We choose the hypothesis that is closest to an optimal location in front of the robot that allows flexible manipulation by both arms, has at least a minimal size, and is higher than its direct surroundings. This is a

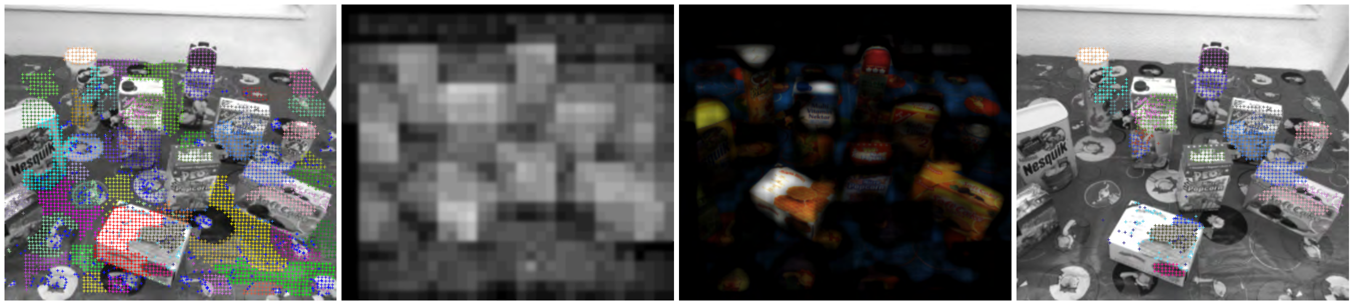


Fig. 4: Suppression of object hypotheses that do not lie in a region that is higher than its direct surroundings. The first image shows a complex scene that leads to the creation of very many initial object hypotheses. The second image displays the map quantifying the relative local height of the image regions. The third image demonstrates the selective effect of applying this criterion: the original image has been multiplied with the height map, thus the high regions are highlighted while lower regions appear dark. The right image shows the remaining initial hypotheses that lie in high regions.

pragmatical choice if the robot does not have any other intention than exploring the objects in front of it. If the object is to be grasped later, it is particularly reasonable to choose one that is higher than its local neighborhood. If the robot is interested in a specific kind of object, other criteria may be appropriate.

The push is planned in such a way that the object is kept in front of the robot and within the camera images. To this end, a central point in front of the robot is defined towards which the object is pushed over a fixed distance to ensure sufficient motion. The motion has to be significant enough to be distinguishable from noise, and as the object extent is unknown, the actual outcome is hard to predict. Therefore the intended motion length should not be too small: values in the range of 10-20 cm turned out to work reliably.

The arm that is better suited to execute this push is chosen based on a reachability analysis [20]. The hand approaches the object on a trajectory significantly above it to avoid collisions with other objects. It is then lowered besides the object, and the force-torque sensor in the wrist is used to react to unplanned collisions during that phase (for details see [14] or [11]). The object is pushed, the hand is lifted again and moved out of sight. Afterwards, we analyze if the object has moved and determine its translation and rotation.

C. *Detection and Analysis of Change in the Scene*

Now we have to find the object that moved by comparing the point clouds before and after the push, which is the most important and most difficult subtask within our segmentation approach. This is due to the fact that (besides the general difficulty of the matching of point cloud subsets) we do not know which part of the point cloud is the object, neither for the cloud before nor the one after the push. Thus, we have to use the difference between them to determine both the subset constituting the object and the transformation that it underwent.

As a first step, we determine which part of the point cloud changed due to the push. This can easily be achieved by comparing the old and new camera images and calculating the difference image. Yet that is only possible if the camera pose before and after pushing is virtually the same. On our robot ARMAR-III, the precision and repeat accuracy of the joints is high enough to allow that; we only need to shift the new image by up to four pixels in all directions when comparing it with the old one, and choose the modified position that causes a minimal difference. On other robots such a precise motion might not be possible, in which case an alternative is to align the two point clouds and find the points that are far away from their nearest neighbor or have a different color. Both methods yield comparable results and enable us to divide the old and new point cloud into a part that is unchanged and a part where a change occurred.

A first result we get immediately from this difference is an answer to the question if anything happened at all. If nothing changed in the scene, the robot was evidently unable to move the potential object or did not hit anything at all. In this case, the robot tries pushing another object candidate. If a change in the scene is detected, all initial object hypotheses are analyzed on whether they lie in image regions that changed. Each object hypothesis is represented by a set of RGBD points, and if more than half of them lie in a region that changed due to the push, the hypothesis will be analyzed for having moved; otherwise it is discarded. In addition to the initial hypotheses, we create new ones from the points that changed. This is done by determining 2-5 clusters[1] amongst these points using x-means, a variant of k-means that automatically chooses the number of clusters [21]. These new hypotheses frequently match the actual object better than the initial ones, although usually not perfectly either.

IV. OBJECT SEGMENTATION

A. *Estimation of the Object Motion*

All the hypotheses that lie in parts of the scene which changed may correspond to the object (or one of several objects) that moved, and therefore they are examined further.

[1]There have to be at least two clusters, as a moving object causes change in the image regions of its old and new position (which may overlap though). More clusters may be appropriate if several objects move, or if there are false foreground regions due to errors in the background subtraction.

Each hypothesis consists of a set of 3D points with associated color information from the point cloud recorded before the push and has to be relocalized within the new point cloud. The probably most popular approach for matching (also referred to as *registration*) of 3D point clouds is the Iterative Closest Point (ICP) algorithm [22]. To register a point cloud with another, two steps are repeated iteratively:

- The nearest neighbor of every point of the first point cloud is determined in the second point cloud
- Based on these correspondences, the 3D transformation that minimizes the mean squared distance between all the pairs is calculated and applied to the first point cloud

These two steps are repeated iteratively until the improvement, i.e. the relative reduction of the mean square distance, lies below a threshold, or a maximal number of iterations has been executed. The algorithm reduces the mean square distance between the point sets in each step and converges to a local minimum.

In our implementation, we define the distance between two points as the weighted sum of their cartesian distance and their distance in normalized RGB space. The weighting is such that the maximal possible color distance is equivalent to a cartesian distance of 10 cm.[2] As we use both shape and color information, we avoid the problem of mismatching in case of similar shapes which would otherwise occur frequently, as the shapes of artificial household objects are mostly dominated by planar surfaces.

When trying to determine the transformation that a hypothetical object underwent during the push, we first try to register the hypothesis with the new point cloud by initializing ICP with its original pose (= position and orientation). If a good match is found, i.e. the resulting (cartesian + RGB) distance is small and the determined transformation indicates that the hypothesis did not move significantly, we consider it to be unchanged. If the determined transformation indicates that the object has moved, or only a bad match was found, it has to be relocalized. The one serious disadvantage of ICP is that it converges to a local optimum, therefore its initialization is decisive for finding the correct match of the object hypothesis after a push. Starting the registration at the original position frequently fails in complex scenes if the object moved over a large distance.

Thus, we execute ICP several times with different initial estimates of the new object pose, and keep the resulting transformation that yields the best match. As the object may have been moved over a large distance, finding it again requires an appropriate choice of the initial poses for ICP. To this end, we detect image regions that resemble the hypothesis in terms of color histogram similarity and initialize the alignment there. If the object surface contains stable local visual features, those can be used to get an initial estimation of the motion, too. The necessary number of different initial positions can be reduced by taking into account the direction of the push, which must not be done in a too restrictive manner as the caused object motion is rather unpredictable.

The best transformation returned by the differently initialized registration attempts is refined by another execution of ICP on a reduced point set where all those points are left out that still have a large distance to their nearest neighbor. The resulting final transformation is used to decide whether the estimated object motion is accepted, and if this is the case, to determine the object segmentation.

B. Verification, Correction and Extension of the Segmentation

After the motion of an object hypothesis has been estimated, the robot needs to decide whether the determined match and transformation are plausible. A hypothesis is only accepted, i.e. considered to correspond to an actual object, if it meets the following three criteria: First, the estimated motion has to be large enough to be sure that it is not due to noise or a slight mismatching[3]. Secondly, the match must be good, i.e. the average distance of the hypothesis points to their respective nearest neighbors in cartesian and normalized RGB space must be below a threshold. Thirdly, the relocalized hypothesis must lie mostly in image regions that have changed. This removes mismatches where by pure chance a good alignment to some part of the scene could be found, e.g. a part of the table surface that was matched to another part of the table after the object has been moved onto it.

The remaining hypotheses do most likely belong to an actual object that has been moved by the robot. But of course we must assume that they do not cover the object completely, and that they also contain points that do not belong to the object. We remove the latter ones by checking each point of the hypothesis: After applying the estimated object motion, a point must match its nearest neighbor in the scene point cloud well with respect to cartesian and color distance. It also has to lie in a region that changed due to the push. If both of these criteria are met, the point is considered to be verified, otherwise it is removed from the hypothesis.

After removing the false points, we try to extend the hypothesis to cover the whole object. To this end, we add all those points to it as candidates that lie close to the verified points and within the image region that changed. By pushing the object again and repeating the steps described before, these new candidate points can be verified or discarded, and new candidates can be added. Depending on the object size and the quality of the initial hypothesis, it usually takes two or three pushes until the whole object is contained in the hypothesis and thus segmented completely.

Usually, more than one object hypothesis is verified by the first push and the subsequent analysis. This happens

[2]This parameter allows to balance the relative importance of color and shape matching. The weight of the color component should not be too small to avoid mismatching due to similar shapes. If it is set too high, the risk of mismatches due to similar color rises. Empirically, values between 5 and 30 cm produced reasonable behavior. The choice may also depend on the precision of the 3D sensor and the sampling density.

[3]Given the precision of our stereo calibration and a distance of 50-80 cm between camera and object, a threshold of 3 cm turned out to be definitely safe.

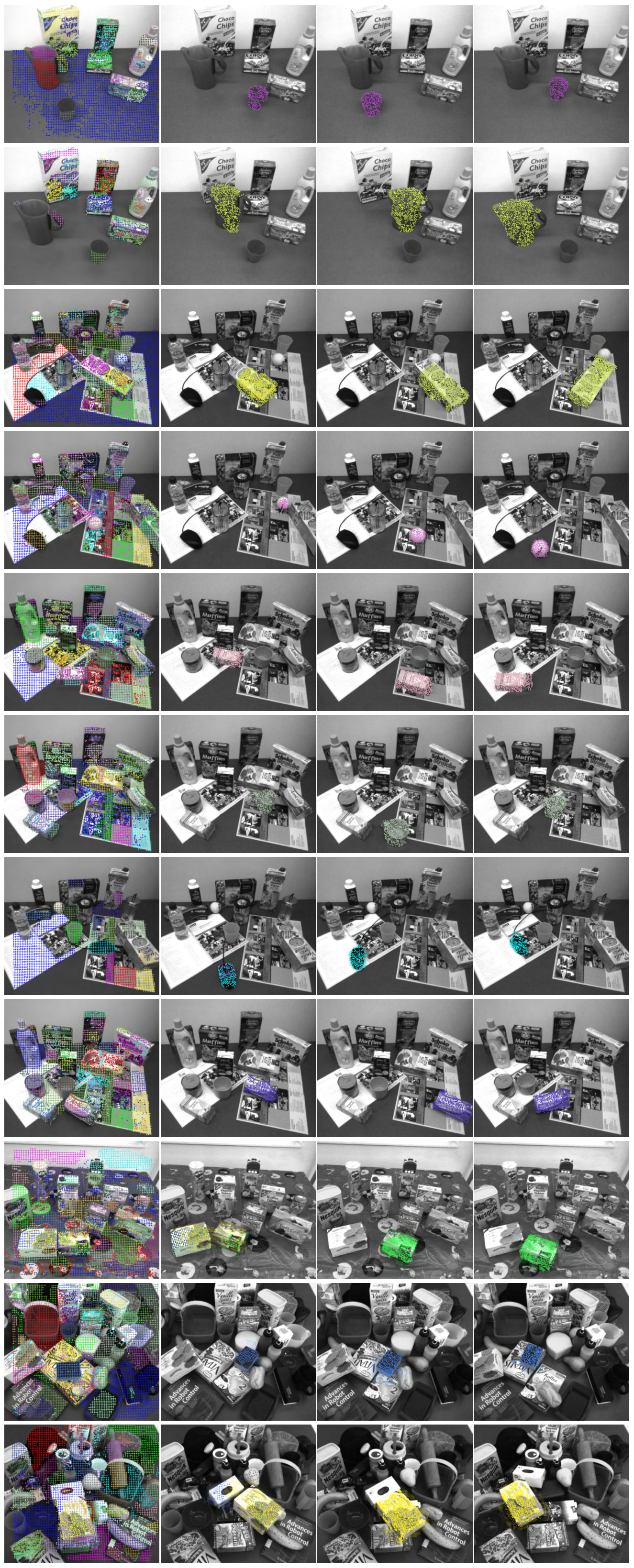


Fig. 5: Examples of object segmentations in different scenes. The first image in each row shows the initial object hypotheses, the second to fourth images show the verified hypothesis after one, two and three pushes.

in particular when several actual objects are moved. We choose the hypothesis containing the maximal number of confirmed points for the second push. After that, we discard the hypotheses that did not move again, and from the remaining ones we keep only the one with the maximal number of confirmed points and continue examining it as long as desired. If the robot did indeed move several objects, all of them can be segmented, but for the sake of simplicity we only observed one in our experiments. As long as the objects undergo different 3D transformation, they can easily be separated based on their different motion. It may happen though that two objects move exactly alike, in which case they are subsumed in one hypothesis. Most likely they are separated when pushed several times from different directions. Heuristics for systematical pushing to this end have been proposed in [5] and [7]. When two objects contained in one hypothesis are separated, the hypothesis will follow the object that is matched better after the motion, which is usually the bigger one.

Pushing an object several times will reveal different sides of it, thus the creation of a multi-view object descriptor is possible, although some sides will probably never be observed. In section V-D we demonstrate that the obtained segmentations are well suited to train an object descriptor that allows for reliable recognition.

V. EXPERIMENTAL EVALUATION

A. System Setup

We have implemented and tested our approach on the humanoid robot ARMAR-III [2]. The video accompanying this paper shows an interactive object segmentation executed by it. The robot has an active stereo camera system in its head, and its arms have seven degrees of freedom each and are equipped with force-torque sensors in the wrists. The cameras provide color images with a resolution of 640×480 pixels. About 85% of the stereo images overlap, and after calculating the dense depth map we use only every second pixel in x and y direction for the point cloud, thus we obtain around 65000 RGBD points that we work with.

The computational effort is dominated by the relocalization of the object hypotheses using the ICP algorithm, in which the computational complexity is proportional to $n\ log(m)$, with n being the number of points of the object hypothesis and m the overall number of points in the scene. On a 3 year old standard PC with a quadcore processor, the computations after each push took between 2 and 5 seconds, depending on the size and number of moved objects.

An important aspect in comparison to some related work is that in our case the robot itself executes the object pushing, and we do not use an artificial setup where the camera always has an undisturbed view of the object. This is the reason why we do not try to track the object during the push, as the robot's hand frequently occludes large parts of it.

B. General Observations

Our approach aims at making it possible to segment rigid objects independently of their appearance or shape, thus we tested it with a large variety of items. They can roughly be classified by their visual appearance as being strongly textured, sparsely or partially textured, multicolored but (almost) non-textured, unicolored, reflective (e.g. polished metallic objects or mirrors), or transparent. As far as we know, the related work in this field (including ours) has so far either depended on local features, i.e. texturedness, on unicoloredness, or on a certain shape.

It turned out that our segmentation approach works very well for all kinds of objects except the very reflective and the transparent ones. This is due to the fact that they appear to change their color when moved, and also tend to cause problems when trying to obtain depth information. All other objects were segmented successfully by our approach; for the transparent and reflective ones a special treatment might be necessary. Although we were able to tune the parameters of the background subtraction and the matching so that the segmentation worked for most of them, it does not function reliably and the chosen parameters depend strongly on the lighting conditions, thus we do not claim that our approach can handle this kind of objects.

In contrast, the shape of the examined objects did not seem to make an observable difference. While distinctive shape features are necessary for algorithms that match point clouds solely based on 3D data, the fact that we use color helps to overcome ambiguities that might arise otherwise. The combined use of shape and color information usually allows a good alignment of the object hypothesis with the object after it has been pushed. An exception here are symmetric unicolored objects, but in that case it actually does not matter if the orientation around the axis of symmetry is met correctly as long as the match is good. The only case in which problems occurred was when a flat, unicolored object was placed on a table of the same color.

C. Assessing the Segmentation Quality

We examined the performance of our interactive segmentation approach by testing it with 30 objects of different shape, size and visual appearance type (as defined above), which have been segmented twice each. To measure the quality of the obtained segmentations, two metrics are determined: First, the object should be segmented as completely as possible, i.e. in an optimal case the point cloud forming the object hypothesis should fully cover the object. The second metric is the size of the falsely segmented area, i.e. the part of the scene that is segmented but does not belong to the object. This happens when the object hypothesis includes points that belong to the background or other objects.

Figure 6 shows these two values depending on the number of pushes executed. As can be seen, after the first push the object is usually not covered completely, but already to a large part. After two to three pushes, the hypothesis contains almost the complete object, with the exception of small patches that newly appeared due to object rotation or that were discarded from the hypothesis due to a change in their appearance (e.g. reflections or bad depth estimation). After four or more pushes, the coverage does not improve further, but different parts of the object may become visible, thus more information can still be gained.

The ratio of falsely segmented image regions compared to the whole object is always quite small. It seems to grow a bit from the first to the second push, but not any further afterwards. Such false positives occur when the shadow cast by the object leads to neighboring image regions being considered to have changed, and some of them look alike before and after the push, which frequently happens on unicolored table surfaces. In this case, the part of the table on which the object casts a shadow appears to belong to the object itself. We are not sure whether there is a theoretically sound solution for this specific ambiguity; it is probably necessary to grasp and lift the object to dissolve it.

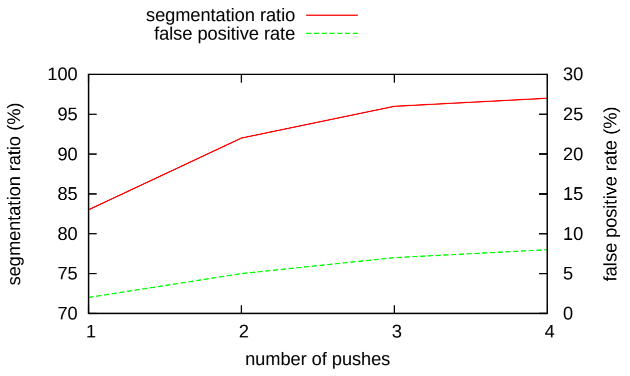


Fig. 6: The average segmentation quality depending on the number of pushes that were executed. The red line shows the segmentation ratio, i.e. the percentage of the object that is included in the segmentation. The dashed green line depicts the false positive rate, i.e. the fraction of the segmentation that does not belong to the actual object.

D. Learning of an Object Descriptor

To demonstrate that the obtained segmentations are sufficiently complete and correct, we use them to train a simple object recognition system. The available information we can use are the image region that contains the object hypothesis, i.e. the segmentation, as well as the 3D and color information contained in the hypothesis point cloud itself. After each push, the object hypothesis and thus the segmentation are different, therefore we could generate several descriptors for each object from different perspectives. For the sake of simplicity, we just use the segmentation obtained after the second push for each object, which usually yields a good coverage, and generate only one descriptor.

To detect the learned objects in new images, we train a color histogram based descriptor using the image region that is occupied by the object hypothesis. The descriptor uses Receptive Field Cooccurrence Histogram (RFCH) features [23], [24] which are based on histograms of the colors and their first and second derivatives in the segmented image area.

These features allow to find image regions that have the same color distribution as the learned object. We then try to

TABLE I: Object recognition rates.

similar point of view	different point of view	partly occluded	false positive rate
98.5 %	70.6 %	67.2 %	3.8 %

match the learned RGBD point cloud in those areas using Iterative Closest Point (ICP) as in the motion estimation step of our segmentation approach. The localization is accepted if the resulting average point distance in Cartesian and color space is below an equivalent of 1 cm (with the maximal possible color distance being equivalent to 10 cm in Cartesian space).

Table I displays the recognition results for our set of autonomously learned objects. They are placed in potentially confusing scenes comparable to those shown in figure 5. When the object is seen from approximately the same point of view as during learning, the recognition rate is almost 100%. If the object has a significantly different orientation with relation to the camera, or if it is partly occluded by other objects, the recognition rate drops to around 70%. This can be improved by using object descriptors generated from different views, as we did in [11]. The false positive rate is about 4%, which is entirely due to two small unicolored objects in our test set that are sometimes fitted into blobs of similar color. These solid recognition results demonstrate the usefulness and quality of the segmentations obtained by the robot following our approach.

VI. CONCLUSIONS

We have presented a new approach for interactive object segmentation exploiting the manipulation capabilities of a humanoid robot. The proposed method enables it to discover and segment unknown rigid objects in an unknown, complex scene by pushing them and analyzing the motion of color-annotated 3D points obtained from the robot's stereo vision system. We have demonstrated that the provided segmentation results are of excellent quality and allow to train a well performing object recognition system. As already shown in [14], it is also possible to subsequently grasp the discovered objects for further examination or manipulation.

In contrast to our previous work in this direction, the approach proposed here works with almost any kind of rigid object except those which are transparent, highly reflective or impossible for the robot to move. We therefore believe that it is a small but important step for increasing the adaptability and autonomy of humanoid robots that will frequently have to deal with new, unknown objects in realistic scenarios.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement 270273 (Xperience).

REFERENCES

[1] G. Metta and P. Fitzpatrick, "Grounding vision through experimental manipulation," *Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences*, vol. 361, no. 1811, 2003.

[2] T. Asfour, K. Regenstein, P. Azad, J. Schröder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann, "ARMAR-III: An integrated humanoid platform for sensory-motor control," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2006.

[3] D. Katz and O. Brock, "Manipulating articulated objects with interactive perception," in *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, 2008.

[4] W. H. Li and L. Kleeman, "Segmentation and modeling of visually symmetric objects by robot actions," *International Journal of Robotics Research*, vol. 30, no. 9, 2011.

[5] L. Chang, J. Smith, and D. Fox, "Interactive singulation of objects from a pile," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3875–3882.

[6] M. Gupta and G. Sukhatme, "Using manipulation primitives for brick sorting in clutter," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3883–3889.

[7] T. Hermans, J. Rehg, and A. Bobick, "Guided pushing for object singulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2012.

[8] E. S. Kuzmič and A. Ude, "Object segmentation and learning through feature grouping and manipulation," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2010.

[9] D. Schiebener, A. Ude, J. Morimoto, T. Asfour, and R. Dillmann, "Segmentation and learning of unknown objects through physical interaction," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Bled, Slovenia, 2011.

[10] A. Ude, D. Schiebener, N. Sugimoto, and J. Morimoto, "Integrating surface-based hypotheses and manipulation for autonomous segmentation and learning of object representations," in *IEEE International Conference on Robotics and Automation (ICRA)*, St. Pauls, Minnesota, 2012.

[11] D. Schiebener, J. Morimoto, T. Asfour, and A. Ude, "Integrating visual perception and manipulation for autonomous learning of object representations," *Adaptive Behavior*, vol. 21, no. 5, pp. 328–345, 2013.

[12] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. Int. Conf. Computer Vision*, Corfu, Greece, 1999.

[13] P. Forssen, "Maximally stable colour regions for recognition and matching," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[14] D. Schiebener, J. Schill, and T. Asfour, "Discovery, segmentation and reactive grasping of unknown objects," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012.

[15] K. Hausman, F. Balint-Benczedi, D. Pangercic, Z. Marton, R. Ueda, K. Okada, and M. Beetz, "Towards tracking-based interactive segmentation of textureless objects," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[16] S. Frintrop, E. Rome, and H. Christensen, "Computational visual attention systems and their cognitive foundations: A survey," in *ACM Transactions on Applied Perception 7*, 2010.

[17] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.

[18] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, "Frequency-tuned salient region detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 1597–1604.

[19] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, Feb. 2008.

[20] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Efficient inverse kinematics computation based on reachability analysis," *International Journal of Humanoid Robotics*, vol. 9, no. 4, 2012.

[21] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *Proc. 17th Int. Conf. Machine Learning*, San Francisco, CA, 2000.

[22] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[23] K. Welke, "Memory-based active visual search for humanoid robots," Ph.D. dissertation, Karlsruhe Institute of Technology (KIT), Computer Science Faculty, Institute for Anthropomatics (IFA), 2011.

[24] S. Ekvall and D. Kragic, "Receptive field cooccurrence histograms for object detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 84–89.

Visual Collision Detection for Corrective Movements during Grasping on a Humanoid Robot

David Schiebener, Nikolaus Vahrenkamp and Tamim Asfour

***Abstract*— We present an approach for visually detecting collisions between a robot's hand and an object during grasping. This allows to detect unintended premature collisions between parts of the hand and the object which might lead to failure of the grasp if they went unnoticed. Our approach is based on visually perceiving that the object starts to move, and is thus a good complement for force-based contact detection which fails e.g. in the case of grasping light objects that don't resist the applied force but are just pushed away.**

Our visual collision detection approach tracks the hand in the robot's camera images and analyzes the optical flow in its vicinity. When a collision is perceived, the most probable part of the hand to have caused it is estimated, and a corrective motion is executed. We evaluate the detection together with different reaction strategies on the humanoid robot ARMAR-III. The results show that the detection of failures during grasp execution and their correction allow the robot to successfully finish the grasp attempts in almost all of the cases in which it would otherwise have failed.

I. INTRODUCTION AND RELATED WORK

Grasping objects is an indispensable competence for humanoid robots. While grasp planning is a challenging problem that (for good reason) received and still receives a lot of attention, the actual execution of the planned grasps on a real robot frequently poses serious problems too. Those difficulties are due to imprecision in object localization, hand-eye calibration and execution of the planned grasping motion, as well as the planned grasps themselves which may sometimes be inappropriate. The authors in [1] and [2] have actually showed that the currently used grasp quality measurements often lead the grasp planners to solutions that are not reliable in the real world despite seeming good in the used mathematical models. The problem of grasp plans that are not or only approximately suitable arises in particular when no precise object model is available or an unknown object is to be grasped based on heuristics (like e.g. in [3] and [4]).

Visual servoing [5] is an important technique that helps to greatly reduce the effects of imprecise hand-eye calibration and inexact arm motion by localizing both hand and object in the same camera images. The position and orientation (pose) of the hand relative to the object is thus determined visually in the camera frame, and as long as the kinematic model is good enough to allow for an approximately correct motion, the hand can be visually guided towards the intended pose by continuous correction [6]. The degree of exactness that can be achieved using visual servoing is essentially limited by the precision of the perception components.

D. Schiebener, N. Vahrenkamp and T. Asfour are with the Institute for Anthropomatics and Robotics, High Performance Humanoid Technologies Lab (H^{2}T), at the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. `schiebener@kit.edu, vahrenkamp@kit.edu, asfour@kit.edu`

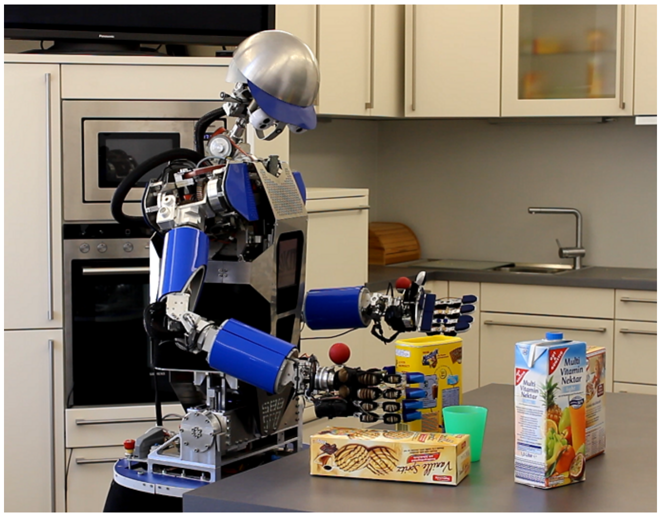


Fig. 1: The humanoid robot ARMAR-III grasping an object.

Thus, when we apply visual servoing, the remaining causes of imprecision are the object localization algorithm, the limited resolution of the vision system, the configuration of the fingers, and the grasp planner, especially when no perfect object model is available. In reality, these errors may be small but will always be present, and a frequent result is that the hand prematurely touches the object and moves it, which may cause the grasping to fail. Therefore, whenever the required accuracy of the grasp can not be guaranteed by the planning, perception and kinematic components, the robot should be aware of possible errors during the grasp execution and be able to detect and correct them.

Collision detection during grasp attempts has mostly been applied in the context of blind or reactive grasping, e.g. in [7], where objects from a box are grasped blindly. The torque detected by a force-torque sensor in the wrist of the robot arm is used to determine which finger touched the object and to correct the hand position accordingly. In [8], we reactively grasp unknown objects that have previously been segmented by vision and pushing actions. There, we use a force-torque sensor in the wrist, tactile pads in the fingers and the palm, and finger joint angle measurements to determine the contact location during the grasping approach and correct the hand position if necessary. In [9], tactile sensors in fingers and palm are used to adapt the hand position and finger configuration to the object pose and shape during the grasp execution. In [10], the tactile sensors in the fingers are used to reactively adapt the finger configuration while closing the hand during grasp execution.

However, all approaches based on force or tactile feedback require that the object resists the robot hand sufficiently so that a force can actually be measured. For top-down grasps, this is usually unproblematic as long as the object is not too easily deformable, but when light objects are grasped from the side, the sensitivity of the currently available sensors is far from being sufficient. One way to circumvent this is to use proximity sensors as in [11], another way is to use visual information, which is what we propose in this work.

To the best of our knowledge, the only other attempt to visually detect collisions in the context of grasping is [12]. They obtain an RGBD point cloud from a static depth camera observing the scene which consists of a table surface with only the object on it and the robot arm, of which a geometric model is available. The arm is tracked in the depth image and the object is segmented by removing the table surface. When the object moves while the arm is near it, a collision is assumed to have occurred. The most probable part of the hand to have caused the collision is determined based on the geometric model. This information would allow to implement a reaction strategy, although this has not been done yet in that paper. It is not obvious though how this approach could be generalized to more complex scenes and a non-static camera.

Our approach is related to [12] in the sense that it is also based on the idea of visually detecting the motion of the object when a collision occurs. We took some inspiration from [13], where a static camera observes a scene in which the robot arm approaches an object and, in the moment it collides with it, causes a sudden spread of optical flow in the image area occupied by the object. In our case the situation is more complex though, as the camera is located in the robot head and moves during the execution of the grasp.

II. OVERVIEW

The execution of a grasp in general comprises the motion of the robot's arm, hand and fingers from an initial pose to a configuration in which the object is held firmly inside the hand. Collision-free motion planning in this high-dimensional space is challenging [14]. A common approach is to separate the grasp and the motion planning step by using precomputed grasp tables which are applied on localized object poses in order to allow for efficient processing. Such grasping pipelines (see e.g. [15] or [16]) usually comprise a motion execution component which is responsible for moving the end effector along a planned path.

Within this work, we assume that a grasping pose p_g together with a corresponding pre-grasp pose p_{pre} are available for the target object. Further, we assume that the straight trajectory between p_{pre} and p_g is collision-free. For our experiments, p_{pre} and p_g were defined manually, but in general this approach can seamlessly be integrated as a grasp execution module within the robot's grasping pipeline. In that case, the grasping poses will be computed by grasp

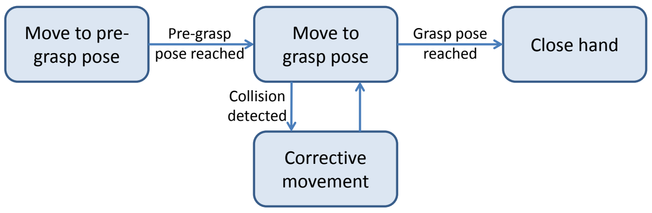


Fig. 2: Schematic overview of grasping with collision detection and correction: First, the robot moves the hand to a pre-grasp pose relative to the object. Then the actual grasp pose is approached while continuously checking for collisions. If a collision is detected, a corrective movement is executed and the grasp pose adapted. When the grasp pose is reached, the fingers are closed.

planning components, and the pre-grasp pose is equivalent to a point on the approach trajectory. Fig. 2 shows a schematic overview of the proposed grasp execution procedure. During the critical last part of the approach, we continuously check for collisions and if one is detected, a corrective reaction is performed.

Fig. 3 shows the processes running during the critical approach phase. The hand is guided towards the grasp pose by visual servoing. At the same time, our visual collision detection continuously checks for indications that the hand has unintendedly collided with the object, in which case the approach is interrupted.

The details of the collision detection algorithm are explained in the following section III, and the different reaction strategies we implemented are described in section IV. The detection and the different strategies are tested on our robot and quantitatively evaluated in section V.

III. VISUAL COLLISION DETECTION

The main idea of our approach is to detect the motion of the object that is caused by the unintended collision with the hand. To this end, we track the hand in the camera images and observe the optical flow next to it in the direction in

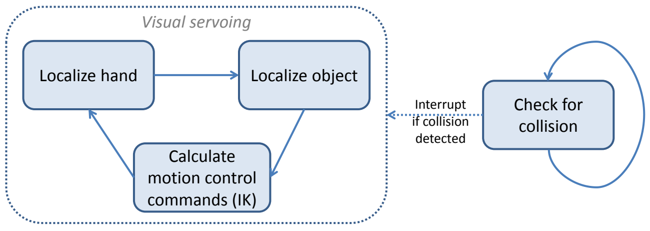

Visual servoing
Localize hand
Localize object
Calculate motion control commands (IK)
Interrupt if collision detected
Check for collision

Fig. 3: Visual servoing with collision detection: The intended grasping pose relative to the object is approached using visual servoing, i.e. the robot continuously localizes object and hand visually, calculates the required relative motion of the hand in Cartesian space and an appropriate joint motion to realize it using inverse kinematics. Concurrently, it checks for collisions, and if one is detected, the approach is interrupted and a correction initiated.

which it is moving. The individual components are described in the following subsections.

A. *Hand Tracking*

Visual tracking of the robot's hand is necessary for both the visual servoing and the collision detection. For the visual servoing, we use a simple and very fast method in which we localize the red spherical marker fixed to the robot's wrist which can be seen in fig. 1 (see [6]). The orientation is obtained from the forward kinematics. For collision detection, we use a tracking algorithm based on the particle filter approach [17] which estimates the position, orientation and finger configuration of the hand. This comprehensive information is particularly important when trying to determine which part of the hand has collided with the object.

The end effector of our humanoid robot ARMAR-III [18] is a five-finger hand which is pneumatically actuated (i.e. with air pressure), has two DoF in each finger and one in the palm. Tactile sensor pads are installed in each finger tip and the palm, and a force-torque sensor in the wrist. However, these sensors are not used in this work[1]. The head is equipped with a stereo camera system.

The particle filter estimates the position and orientation of the hand, and a reduced set of the finger DoF, which results in a 12-dimensional state space. The particles are initialized with the position from the localization of the spherical marker, the orientation from forward kinematics, and the measured finger joint angles. In each iteration of the particle filter algorithm, the particles are perturbed by adding random Gaussian noise, and then the plausibility of the hand configurations defined by the particles is evaluated based on the current camera images. In the next iteration, particles are redrawn with a probability proportional to their rating, the relative hand motion since the last iteration is applied to them, random noise is added and they are evaluated again. Robustness of the tracking is enforced by only allowing particle configurations that are within an empirically determined interval around the configuration obtained from forward kinematics and joint value sensor readings.

The key component of the particle filter is the rating function which estimates for each particle s_i the conditional probability $p(z|s_i)$ that the input z (the camera images) was caused by the hand configuration defined by s_i. This probability is calculated based on five different cues, which are each determined in both of the stereo camera images. The first cue is $q_1(s_i) = 1/d$, where d is the distance between the positions of the red spherical hand marker in the model and in the camera images. The other four cues are based on the blue fingertips: They are projected into the images given the hand configuration of particle s_i. The cue $q_2(s_i)$ gives a rating proportional to the number of pixels in the area covered by the fingertips that have the correct color, $q_3(s_i)$ rewards if a large part of the area has the correct color[2]. The cue $q_4(s_i)$ checks for intensity edges in the image that correspond to those of the projected fingertip, and $q_5(s_i)$ takes the edge directions into account. The conditional probability of a given particle s_i is then

$$p(z|s_i) \;=\; \vartheta \; e^{\sum_{j=1}^{5} \omega_j \; q_j(s_i)}$$

where ϑ is a scaling factor and the ω_j are weights for the different cues.

On each pair of stereo camera images, two iterations of simulated annealing are performed to enhance the precision of the final localization result, which is the average of all particles weighted with their probability.

B. *Optical Flow*

Concurrently with the hand localization, we calculate the optical flow between the current camera image and the one taken at the last iteration of the collision check. The optical flow is determined using the algorithm proposed by [19] which is implemented in OpenCV. The idea of the algorithm is to approximate the neighborhood of each pixel by a quadratic function. If a quadratic function undergoes a translation, the displacement can be determined in closed form. By iteratively determining these translations first on a coarse and then on increasingly finer scales, larger displacements that exceed the direct neighborhood of the pixel can also be determined and refined. The algorithm provides a dense estimation of the optical flow between two images, although in larger monotone image regions it does not return any values. This is not a problem in our case, as we are interested in the image region around hand and object which offers enough visual information for the algorithm.

C. *Collision Detection*

In [13], the moment of the collision between robot arm and object is recognized by the fact that an area of significant optical flow appears next to the hand. In our case, the cameras are on the robot and moving with it during the grasp, and consequently there is optical flow throughout the whole image. Therefore we have to solve the more general problem of discovering if an object next to the hand moves in a way that is inconsistent with the rest of the scene. Note that for the static part of the scene, its projected motion is not equal throughout the image but depends on the distance to the camera.

To overcome this problem, we cluster the pixels of the camera image by their optical flow values. To this end, we apply x-means, a variant of k-means that automatically determines an appropriate number of clusters given a parameter that balances the number of clusters and their in-class variance [20]. The idea is to detect if there is a cluster of similar optical flow next to the hand which is different from the optical flow in the rest of the scene, which would indicate that an object is being moved by the hand.

[1]We used these sensors for reactive top-down grasping of unknown objects in [8].

[2]This way, we assure that the rating is not too good if the projected fingertips are extremely small or large, which would be the case if only one of the two criteria was used.

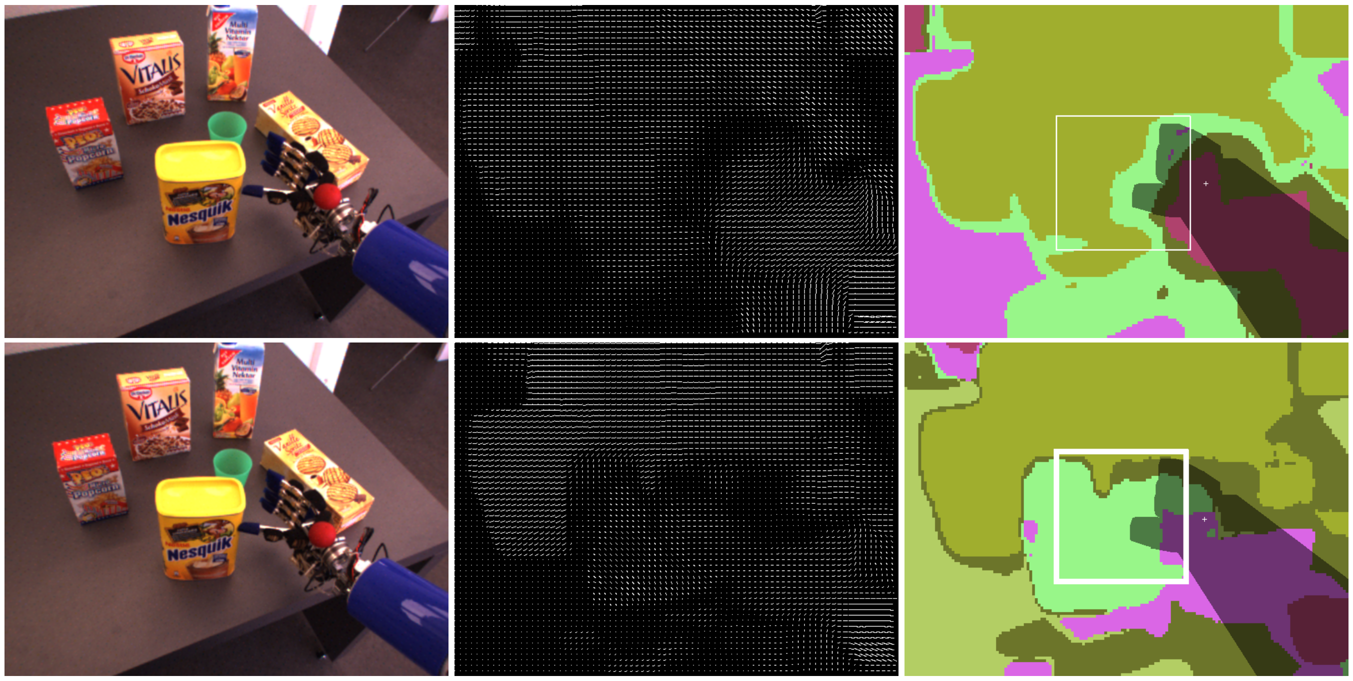


Fig. 4: Visual collision detection in the moment when the robot's hand touches the object: The left column shows the scene from the robot's cameras immediately before and after the collision. The central column visualizes the optical flow, the right column the clusters of similar optical flow, where each cluster has been marked with a distinct color. The darker area is occupied by hand and arm and therefore ignored. The white box marks the area next to the hand where we expect a possible collision to occur. If we find a cluster of optical flow that exists mostly within this area but not outside of it, this observation indicates that the hand collided with an object and caused it to move.

The image area that is checked for such an outstanding cluster is determined by taking the hand position, adding a translation into the direction into which the hand is currently moving, and projecting this point into the image. A quadratic area around that point which has roughly the size of the object is then analyzed[3]. For each cluster of similar optical flow c_i, we count the number n_i of pixels belonging to it in the whole image, and the number a_i of pixels belonging to it in the area in front of the hand. If for one of the clusters the ratio $\frac{a_i}{n_i}$ is more than 0.5, i.e. most of the pixels of the cluster occur inside that small area, this is a strong indication that this unique motion has been caused by an object that is being moved by the robot hand.

It is very probable (yet not certain) that the object will move in a similar way as the robot's hand and parts of its arm. Therefore the image area covered by hand and arm, which is determined based on the results of the hand tracking, is not taken into account when the values n_i and a_i are determined. Fig. 4 visualizes the optical flow, its clustering and the relevant image regions just before and during a collision.

Note that due to the restricted area in which we expect collisions to happen, individual motion in the background (which most of the time moves as a whole due to the camera's motion) can theoretically cause false collision detections, but only when it occurs within the image area next to the hand in the direction into which it is moving as described above.

[3]The size of the object in the image can be estimated from its model and the distance to the camera.

IV. CORRECTIVE REACTION

When the robot detects an unintended collision during the grasp execution, it should react in a way that allows to successfully complete the grasp. Optimally, the robot would have all relevant information about shape and pose of the hand and the object and could just re-plan a collision-free grasp trajectory. But obviously this information is not available, otherwise the collision wouldn't have occurred in the first place. Thus we have to use robust heuristics that can deal with incomplete and uncertain information and create a reaction that has a good chance of correcting the execution error that the robot committed.

A. *Collision Localization*

One piece of information that is necessary for a reasonable corrective reaction is which part of the hand collided with the object. A random change of the hand pose may sometimes be successful, but as shown in our experiments in section V the informed reaction strategies are clearly superior to random modifications of the grasp.

The information which part of the hand touched the object is immediately available when one uses tactile sensors, but if the collision was detected visually it has to be determined in another way. Although this depends on the kind of hand that is used, one can assume that for a majority of grasping motions the fingers and in particular the fingertips are the primary causer of premature collisions. In the case of the conducted experiments, they are virtually always caused by the fingertips. We therefore obtain their positions from the hand localization and check which fingertip is closest to the object. This measurement is of course subject to errors in the perception of hand and object, but seemed to be always correct in our experiments.

B. Reaction Strategies

We implemented and evaluated different reaction strategies to correct the hand pose in the case of a premature collision. We limited ourselves to modifying the position and orientation of the whole hand, although we are aware that there are cases in which it would be necessary to correct the configuration of individual fingers.

The general reaction scheme is the same for all our proposed strategies: When a collision is detected, the hand retreats 2 cm into the direction that it came from with an absolutely straight motion to avoid disturbing the object any more. A corrective offset for the hand pose is calculated according to the respective strategy. The hand retreats another 2 cm during which half of the corrective offset is already applied, to make sure that the reaction has already taken effect before approaching the object again. The corrective offset is then permanently applied to the grasp pose definition. From that point on, the robot moves towards the intended grasping pose again as usual.

If the robot collides with the object again, another corrective reaction takes places. Thus, the corrective offsets add up, and the robot repeatedly tries to grasp and corrects the hand pose as often as necessary until the grasp is successful. In practice it would probably make sense that if the grasp doesn't succeed after a certain number of corrections, a totally different grasp is planned.

Within our reaction scheme, the key to a helpful correction is to determine an appropriate corrective offset. As a baseline, we implemented a strategy where the orientation of the hand is modified by a small random rotation. Such a purely exploratory approach would probably be the only possibility if there were no further information available about the collision, and there is a certain chance that the grasp will eventually succeed after one or more random modifications of the hand pose.

As in our case the information which finger collided with the object is available, we can determine a more constructive correction offset. The obviously useful kinds of motion are to either translate the hand into the direction of the finger that caused the collision, thus aligning the palm with the closest part of the object, or to rotate the hand such that the finger is turned away from the object, or a combination of both. We implemented all three variants and comparatively evaluate them in section V.

In the case of the hand of ARMAR-III, the thumb opposes the four other fingers, therefore we only need to distinguish whether the thumb or one of the other fingers collided with the object. Thus, our results can directly be transferred to simple grippers or precision grasps with two fingers. For more general hand configurations the reaction strategies have to be adapted to the individual hand geometry following the above principles.

TABLE I: Collision detection rate depending on the distance that the object has moved

2 mm	5 mm	10 mm
76 %	92 %	96 %

TABLE II: Collision detection rate depending on the angle between image plane and direction of movement (over a distance of 5 mm)

$0°$	$45°$	$70°$	$90°$
92 %	96 %	84 %	72 %

V. EXPERIMENTAL EVALUATION

We evaluated our approach on the humanoid robot ARMAR-III [18]. First, we tested the sensitivity of our visual collision detector by manually moving the object over a fixed distance while the robot hand was close to it. Table I shows the detection rate for the object to have moved, depending on the distance over which it moved. As can be seen, even if the object was shifted only by 2 mm, this is already detected most of the times, and when the translation is 5 mm or more the detection rate is clearly over 90%. The rare cases in which the object motion is still not detected when it moved more than 5 mm occur when less than half of the object lies within the observed area in front of the hand (see section III-C). We did not observe significant differences in the detection rate between the robot head being static or in motion, or when people were moving in the background.

One concern we had about our approach was whether it would be able to detect the object motion when it occured in the direction perpendicular to the image plane. As we are using optical flow, motions within the two dimensions spanned by the image plane should create the clearest signal, while e.g. a motion straight away from the camera would only cause the object to shrink in the image. Therefore, we tested moving the object by 5 mm in different angles relative to the image plane. The results can be found in table II, the experiment at an angle of approximately $0°$ is identical to the one for 5mm in table I. At $45°$ there seemed to be no difference, at an angle of around $70°$ the detection rate dropped slightly to 84%. This is the biggest angle that may occur in practice on our robot, as the tables etc. on which objects might be placed are lower than the cameras, therefore a motion on the table plane can never be exactly perpendicular to the image plane when the robot looks approximately towards the object. For the sake of completeness, we also tested a motion exactly into the depth direction and still obtained a reasonable detection rate of 72%.

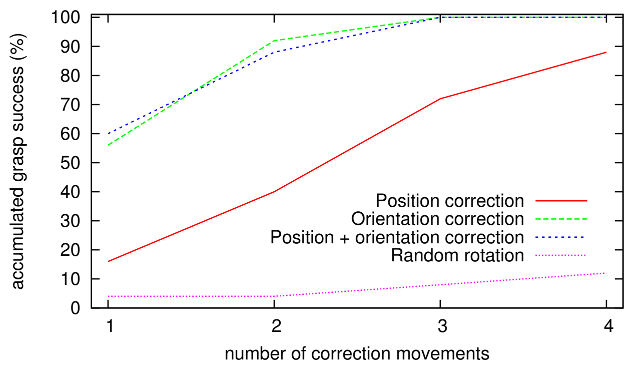


Fig. 5: Percentage of successful grasps after a certain number of correction movements, depending on the applied strategy. E.g., for the orientation correction strategy, in 56% of the cases one corrective motion was enough, another 36% of the grasps were successful after two corrections, and the remaining 8% of the attempts required three corrective movements.

In the actual grasping experiments, we had virtually no problems with undetected collisions. However, although the experiments above suggest that a very small motion is sufficient to detect the collision between hand and object, the objects were usually pushed a few centimeters. The two reasons for that are that our detection algorithm runs only at 2-3 frames per second on a standard PC due to the relatively high computational intensity of hand localization, optical flow calculation and clustering. More importantly, when a collision is detected, it takes some time until the hand actually stops moving forward. For these reasons, depending on the speed of the arm motion, the objects were usually pushed over 1-4 cm. Therefore, when for some reason the impact on the object position has to be kept minimal, the approach speed would need to be relatively slow, depending on the responsiveness of the used robot arm and, when this is very good, also on the available computational capacity.

Finally, we tested the performance of the overall system with the collision detection and the different reaction strategies we proposed in section IV-B[4]. We used five different test objects, and for each of them manually defined a grasp that seemed reasonable but failed in the real world when executed on the robot. We evaluated every reaction strategy by placing each of the five objects at five different reachable poses in front of the robot. We ignored grasp attempts that were immediately successful, so every strategy was tested with 25 grasps during which at least one collision with the object occurred.

Fig. 5 depicts the results of these trials. It shows how many of the grasping attempts had succeeded after a given maximal number of corrections. The baseline strategy where after a collision the hand hand pose was modified by a random rotation of 25° performs rather badly, as was to be expected. In only one case a single correction movement lead to a successful grasp, and another attempt succeeded after three and four corrections respectively.

The proposed strategies that take into account which finger seems to have caused the collision perform significantly better. The one where the position is modified by a translation of 25 mm towards the finger that caused the collision manages to successfully grasp the object after one correction in 16% of the cases, and in another 24% two corrective movements are sufficient. In 32% of the attempts three corrections were necessary. The overall success rate after at most four corrections is 88%, which is already quite an achievement regarding the fact that without the reactions all those grasps would have failed. In two of the remaining three cases the object was still grasped after further correction movements, but in one case it was finally pushed out of reach of the robot.

However, the two reactive strategies where the orientation of the hand was changed by 25° to turn the colliding finger away from the object, or the orientation changed by 15° and the position by 10 mm, turned out to be very successful. Both managed to grasp the object after one correction in about 60% of the attempts, and had an overall success rate of around 90% after one or two and 100% after at most three corrective movements.

We believe that the reason why the strategies that apply a rotational correction are so much more effective than the one that corrects only the position is that in our implementation of visual servoing, only the position of the hand is visually corrected, but its orientation is obtained from the forward kinematics of the robot. Thus, the orientation error during execution is much bigger than the position error. Additionally, when localizing an object, its position is usually determined more reliably than its orientation. Therefore, it is entirely possible that on other robotic platforms, or when the visual servoing can also correct the hand orientation, the comparison between the three strategies might turn out differently.

VI. CONCLUSIONS AND FUTURE WORK

We have presented an approach for visually detecting undesired premature collisions between a robot's hand and the object that is being grasped. The detection is based on analyzing the optical flow next to the hand in the direction into which it is moving, and detecting when the optical flow there is different from the rest of the scene, which indicates that the robot has caused the object to move.

We also proposed different strategies how to react to such a collision, which take into account an estimation of which finger has caused it and correct the hand position and/or orientation appropriately. Experimentally we showed that the detection works very reliably and that the proposed reaction strategies allow to correct a failed grasp attempt and virtually always conclude it successfully.

As the next step, we plan to complement this visual collision detector with classical tactile and force feedback sensors to cover both the cases in which the object is moved

[4]A video of the experiment is submitted with the paper, a high quality version can be found on https://www.youtube.com/watch?v=MkNIFWth5D4.

Convexity based object partitioning for robot applications

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

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

I. Introduction

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

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

II. Methods

A. Method overview and basic definitions

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

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

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

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

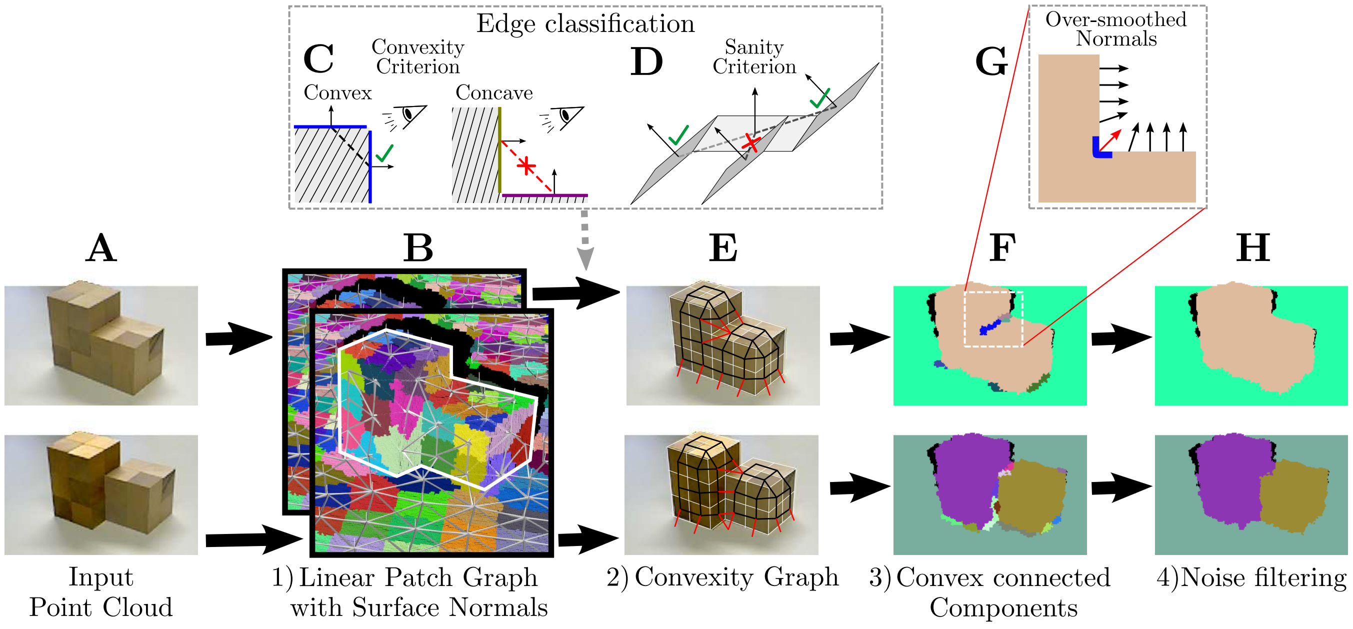


Fig. 1. Flow diagram of the segmentation algorithm. Two example cases are shown: single-object case (upper panels) and two-objects case (lower panels). Illustration of **A)** RGB images corresponding to the point clouds (not shown) of the scenes, which serve as an input to the segmentation algorithm.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

B. *Method flow-diagram*

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

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

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

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

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

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

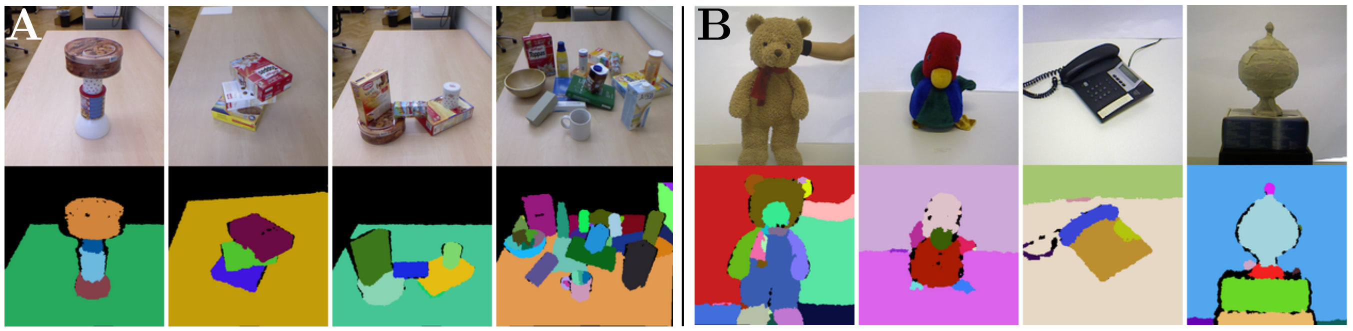


Fig. 2. Examples of segmentation: **A)** images from OSD dataset and **B)** images from our data set. Top and bottom panels show original and segmented images, respectively. Points beyond a distance of 1.3 m were cropped for visualization.

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

C. *Benchmarking and Measures*

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

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

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

III. Results

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

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

A. *Segmentation Examples*

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

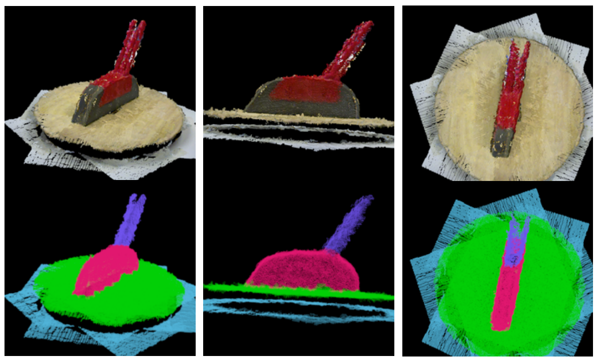


Fig. 3. Example segmentation of a point cloud combined from multiple views showing a foam hand broom. The input point cloud and segmentation result from different view points are shown in the top and bottom images respectively (also see supplementary video).

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

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

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

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

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

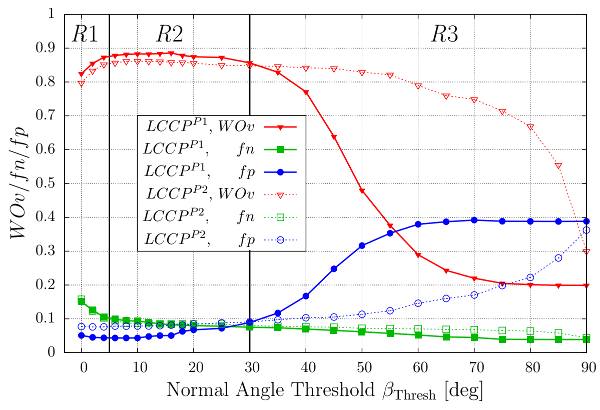


Fig. 4. Statistics obtained from segmentation of scenes from the OSD dataset using our method. Average results are shown. $LCCP^{P1}$ and $LCCP^{P2}$ stand for the different parameter sets with small (solid lines) and large (dashed lines) supervoxels respectively.

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

C. *Robotic application*

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

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

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

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

TABLE I
COMPARISON OF DIFFERENT SEGMENTATION METHODS.

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

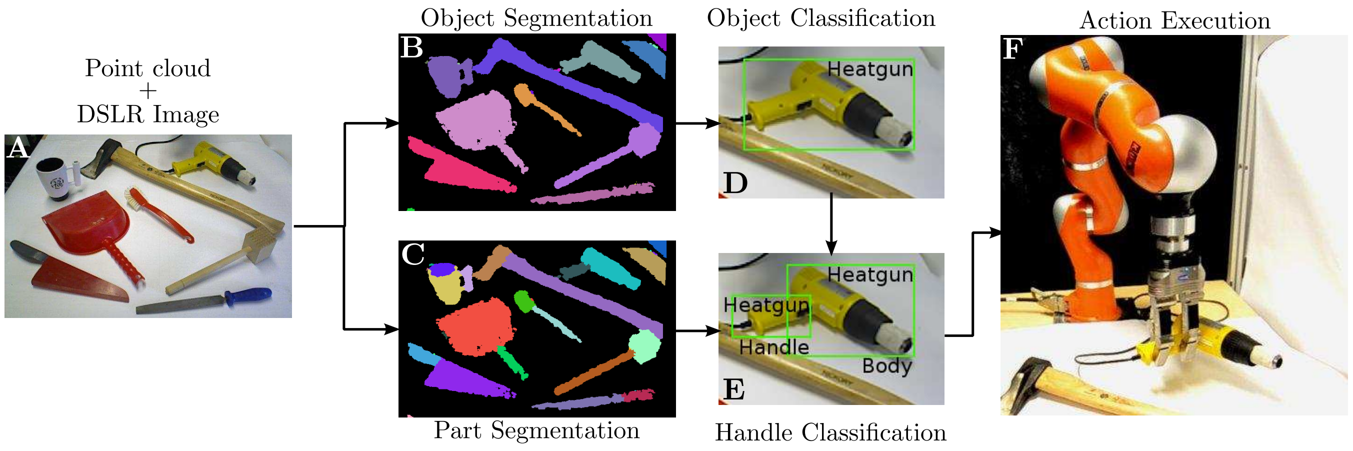


Fig. 5. Flow diagram of robotic application: **A)** Original image of the scene, **B, C)** object/part segmentation, **D, E)** object/part classification and **F)** action execution – robot grasping a heatgun by its handle. LCCP segmentation was applied after an initial ground plane subtraction. While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

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

IV. DISCUSSION

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

A. State of the Art

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

B. A compositional view on affordances and objects

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

APPENDIX

A. Formalism of the Segmentation Algorithm

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

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

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

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

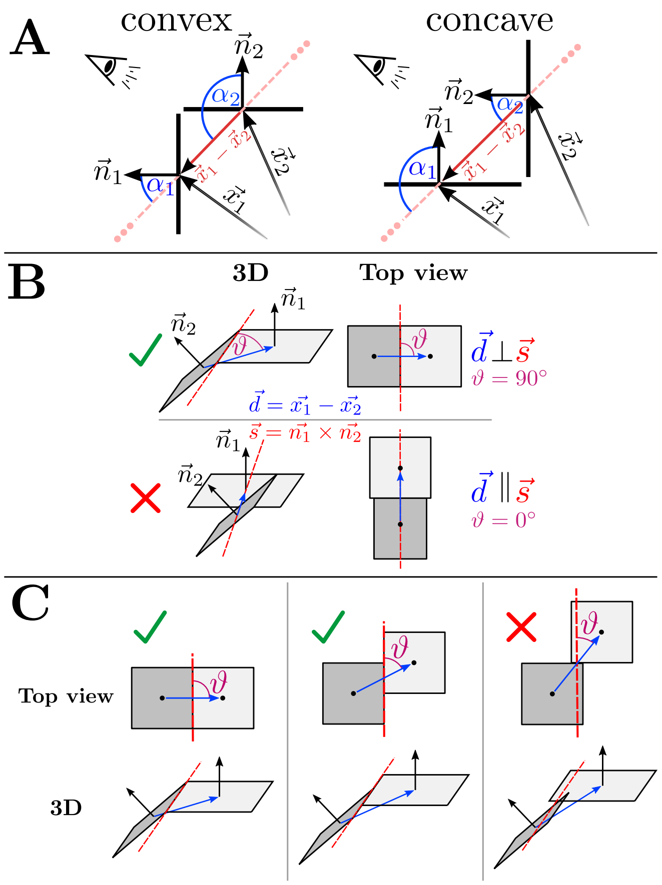


Fig. 6. **A)** Illustration of convex and concave connections between two linear patches. **B,C)** Illustration of the sanity criterion: **B)** A singular connection can be obtained by measuring the angle ϑ between the line of intersection $\vec{s}$ of the two planes represented by the patches and the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

The angle of the patch normals to the vector $\vec{d} = \vec{x}_1 - \vec{x}_2$ joining the centroids can be calculated easily using the identity for the dot product $\vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}| \cdot \cos(\alpha)$ with $\alpha = \measuredangle(\vec{a}, \vec{b})$. One can see in Fig. 6 A, that α_1 is smaller than α_2 for *convex* connections. This can be expressed as:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The sanity criterion SC is then defined as

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

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

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

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

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

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

TABLE II
PARAMETER SETS USED FOR PART SEGMENTATION.

Parameter set	v	s	n_{filter}
P1	0.5 cm	2 cm	3
P2	0.75 cm	6 cm	1

B. Definition of Measures

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

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

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

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

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

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

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

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

C. Evaluation problems

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

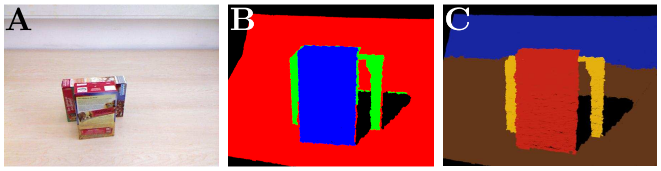


Fig. 7. **A)** Original image, **B)** ground-truth from point cloud perspective from the OSD dataset and **C)** segmentation result of our method.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

by the push and in which it resists it. How to combine these different sources in a constructive way to faster and more reliably detect collisions, and to determine the part of the hand that caused them, will be the central question here. This is of particular interest when executing more complex grasps or manipulative actions.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme FP7 under grant agreement 270273 (Xperience).

REFERENCES

[1] J. Kim, K. Iwamoto, J. Kuffner, Y. Ota, and N. Pollard, "Physically based grasp quality evaluation under pose uncertainty," *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1424–1439, 2013.

[2] J. Weisz and P. Allen, "Pose error robust grasping from contact wrench space metrics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 557–562.

[3] G. Kootstra, M. Popovic, J. Jørgensen, K. Kuklinski, K. Miatliuk, D. Kragic, and N. Krüger, "Enabling grasping of unknown objects through a synergistic use of edge and surface information," *ijrr*, vol. 31, no. 10, pp. 1190–1213, 2012.

[4] D. Fischinger and M. Vincze, "Empty the basket - a shape based learning approach for grasping piles of unknown objects," in *iros*, 2012, pp. 2051–2057.

[5] J. Hill and W. Park, "Real time control of a robot with a mobile camera," in *The 9th International Symposium on Industrial Robots*, 1979, pp. 233–246.

[6] N. Vahrenkamp, S. Wieland, P. Azad, D. Gonzalez-Aguirre, T. Asfour, and R. Dillmann, "Visual servoing for humanoid grasping and manipulation tasks," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2008, pp. 406–412.

[7] J. Felip, J. Bernabe, and A. Morales, "Contact-based blind grasping of unknown objects," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012, pp. 396–401.

[8] D. Schiebener, J. Schill, and T. Asfour, "Discovery, segmentation and reactive grasping of unknown objects," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012, pp. 71–77.

[9] K. Hsiao, S. Chitta, M. Ciocarlie, and E. Jones, "Contact-reactive grasping of objects with partial shape information," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 1228–1235.

[10] H. Dang and P. Allen, "Stable grasping under pose uncertainty using tactile feedback," *Autonomous Robots*, vol. 36, no. 4, pp. 309–330, 2014.

[11] K. Hsiao, P. Nangeroni, M. Huber, A. Saxena, and A. Y. Ng, "Reactive grasping using optical proximity sensors," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 2098–2105.

[12] J. Bernabe, J. Felip, A. P. del Pobil, and A. Morales, "Contact localization through robot and object motion from point clouds," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013.

[13] G. Metta and P. Fitzpatrick, "Grounding vision through experimental manipulation," *Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences*, vol. 361, no. 1811, 2003.

[14] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Simultaneous grasp and motion planning," *IEEE Robotics and Automation Magazine*, vol. 19, no. 2, pp. 43–57, 2012.

[15] D. Chen, Z. Liu, and G. Wichert, "Grasping on the move: A generic arm-base coordinated grasping pipeline for mobile manipulation," in *European Conference on Mobile Robots (ECMR)*, 2013, pp. 349–354.

[16] T. Asfour, N. Vahrenkamp, D. Schiebener, M. Do, M. Przybylski, K. Welke, J. Schill, and R. Dillmann, "ARMAR-III: Advances in humanoid grasping and manipulation," *Journal of the Robotics Society of Japan*, vol. 31, no. 4, pp. 341–346, 2013.

[17] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[18] T. Asfour, K. Regenstein, P. Azad, J. Schröder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann, "ARMAR-III: An integrated humanoid platform for sensory-motor control," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2006, pp. 169–175.

[19] G. Farneback, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis*, ser. Lecture Notes in Computer Science, J. Bigun and T. Gustavsson, Eds. Springer Berlin Heidelberg, 2003, vol. 2749, pp. 363–370.

[20] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *Proc. 17th Int. Conf. Machine Learning*, San Francisco, CA, 2000, pp. 727–734.