Visual Perception of Objects and their Parts in Artificial Systems

Dissertation

in order to obtain the doctoral degree in Mathematics and Natural Sciences
"Doctor rerum naturalium"

of the Georg-August-Universität Göttingen

in the Doctoral program of
the Georg-August University School of Science (GAUSS)

submitted by
Markus Schoeler of Sindelfingen, Germany

Georg-August-Universität Göttingen
Göttingen, Germany
October 2015

Betreuungsauschuss:

Prof. Dr. Florentin Wörgötter, Abteilung Computational Neuroscience, III Physikalisches Institut

Prof. Dr. Winfried Kurth, Abteilung Ökoinformatik, Biometrie und Waldwachstum

Mitglieder der Prüfungskommission:

Referent: Prof. Dr. Florentin Wörgötter, Abteilung Computational Neuroscience, III Physikalisches Institut

Koreferent: Dr. Frank Guerin, Department of Computing Science, University of Aberdeen, UK

Weitere Mitglieder der Prüfungskommission:

Prof. Dr. Winfried Kurth, Abteilung Ökoinformatik, Biometrie und Waldwachstum

Prof. Dr. Wolfgang May, Datenbanken und Informationssysteme, Institut für Informatik

Prof. Dr. Hansjörg Scherberger, Deutsches Primatenzentrum Göttingen

Prof. Dr. Carsten Damm, Theoretische Informatik, Institut für Informatik

Tag der mündlichen Prüfung: 12.10.2015

Visual Perception of Objects and their Parts in Artificial Systems

Abstract

Humans are able to perceive their surrounding apparently with ease. Without much thinking we can process the complex visual stream into meaningful entities which we call objects. How we do this remains an open question already addressed by years of research. Still, there exists a general consensus that (so-called) Visual Object Perception is one of the most fundamental abilities of intelligent agents to make sense of their environment.

In this thesis we advocate the idea that Visual Object Perception can be decomposed into three concurrent ways of perceiving objects: *Instance*, *category*, and *function* perception. This decomposition emanates from the idea that *perception* is inseparably intertwined with actions and tasks. If actions require a specific object (*e.g.*, fill this tea into my teddy-bear cup), one starts perceiving available objects at the *instance* level. If the task asks for a generic *cup* (*e.g.*, go to the supermarket and buy some cups), agents need to perceive objects at the *category* level, without caring for the exact *instances*. Finally, the *function* level is used when objects are defined by the task itself instead of a specific *category* name. For example, transport water from A to B (1) or bore a hole into the soil for seeding plants(2). Both tasks define objects by the role they have in the action context, *i.e.*, a fillable object (1) and an object to poke/bore into the soil (2), respectively.

Especially having mastered *function* level perception was a step in our cognitive evolution which enabled early hominids during the advent of humankind to make sense of their environment and use objects as tools. Eventually, this allowed us to build better tools driven by human ingenuity which separates us from all other animals.

In order to make a machine interact with objects in a "human-like" way, we see two questions which need to be addressed: First, what objects do I see and, second, how can I manipulate or use these objects? The former requires label assignment (*e.g.*, classification, recognition), the latter requires to estimate the orientation and location (pose) of recognized objects

in order to correctly apply motor behavior to use them. Depending on the required perception level (*i.e.*, *instance*, *category*, or *function*), both problems need to be treated with different approaches. Consequently, there is a total of 6 sub-problems (2 problems $\times$ 3 perception levels): Instance Recognition, Object Categorization, and Object Function Assignment; Pose Estimation of instances, Pose Estimation at the category level, and Pose Estimation at the function level. In this thesis we contribute to Instance Recognition, Object Categorization, Object Function Assignment, and Pose Estimation at the category level. While not published at the time of submission of this thesis, we also discuss a small preliminary study about Pose Estimation at the function level at the end of this thesis.

For Instance Recognition all objects in the environment are uniquely defined and need to be discriminated. This requires all objects to be recorded and learned before a system is able to recognize them. As a consequence, it limits agents to specific environments; moving a machine to a new environment would require a new training set and more training. To solve this problem, we present a method which is able to automatically record a training set from a scene with minimal human supervision. Moreover, to deal with highly visual similar objects (*e.g.*, two similar looking cups) we develop an algorithm which is highly discriminative, while being robust to illumination, scale, and object-rotation.

At the *category* level we treat Object Categorization as well as Pose Estimation. As rich models like Deep Convolutional Neural Networks have become de facto standard in modern Object Categorization systems, huge amounts of relevant training data are required. Our first contribution, the *TransClean* algorithm, is able to generate such large sets of relevant training images for *categories*, while also dealing with ambiguous *category* names. For example: The *categories* apple, nut, and washer are ambiguous (polysemes), because they can refer to different objects: Apple refers to a notebook or the fruit; nut to the hardware or the fruit; washer to the hardware or a washing-machine. The general idea is that this ambiguity usually does not exist in other languages. For example, washer translates to the German words "Waschmaschine" (the washing-machine) and "Unterlegscheibe" (the hardware) - the ambiguity does not exist here. *TransClean* uses this idea to sort out irrelevant images retrieved from online word-tagged image databases (*e.g.*, Google Image-Search) by comparing images retrieved for different lan-

guages. The second contribution aims at treating the challenging task of Pose Estimation at the category level. This is complicated, because the system cannot align stored models to recorded known *instances* in the scene (which is done for Pose Estimation of instances). We treat this by introducing a Deep Convolutional Neural Network which not only predicts the *category* but also the category *pose* of objects. The need for a large set of annotated training data is met by synthesizing cluttered indoor scenes.

Lastly, the *function* level is determined by treating objects not as a whole but, instead, as an aggregation of parts in specific constellations. First, we present three sequential algorithms for segmenting a scene into objects and objects into their parts. Second, we develop a framework which analyses the parts and part-constellations to learn the *function* of each part (*e.g.*, being a blade or a tip) together with the function of the object as a whole (*e.g.*, being something for cutting, drilling). Interestingly, objects and their parts can possess multiple *functions*. For example, a hammer-like object can be used to hit a nail or it can be used as a makeshift replacement for task (2), defined earlier: Bore a hole into the soil for seeding plants, now, using the handle as the tool-end.

All the work presented in this thesis has been systematically evaluated using existing or new benchmarks and proved better than state-of-the-art in their respective tasks.

The comprehensive treatment of Artificial Visual Object Perception which we introduce in this thesis has widespread application in various scenarios including robots in human healthcare, house-hold robots, and robots for emergency response (*e.g.*, disaster zones). For example, it allows for new problem solving strategies in agents. Instead of looking for a predefined and hard-coded object which solves a task, agents can perceive objects at, for example, the *function* level and propose creative solutions: Use a hammer to bore a hole into soil or push a button which is out of reach; use a boot or a helmet to transport water.

Contents

List of Acronyms

BoW Bag of Words

CPC Constrained Planar Cuts

CRF Conditional Random Field

DCNN Deep Convolutional Neural Network

DDVG Depth Dependent Voxel Grid

DoF Degree of Freedom

FPFH Fast Point Feature Histogram

GLOH Gradient Location and Orientation Histogram

ICP Iterative Closest Point

ILSVRC ImageNet Large Scale Visual Recognition Challenge

IR Instance Recognition

LCCP Locally Convex Connected Patches

MRF Markov Random Field

OC Object Categorization

OFA Object Function Assignment

OP Object Partitioning

PCL Point Cloud Library

PEC Pose Estimation at the category level

PEF Pose Estimation at the function level

PEI Pose Estimation of instances

RANSAC Random Sample Consensus

SfM Structure from Motion

SHOT Signature of Histograms of Orientations

SIFT Scale-invariant feature transform

SURF Speeded-Up Robust Feature

SVM Support Vector Machine

VCCS Voxel Cloud Connectivity Segmentation

VOP Visual Object Perception

Glossary

class Used in this thesis in a general sense to distinguish between objects with different labels. Which entities are regarded different depends on the level of perception. This is different to some works in the literature which use *class*, *object classification*, or *generic recognition* to describe what we refer to by *category* and *categorization*.

inter-class variance The variance among objects of different classes in a classification context. The bigger the variance, the more different the objects' appearance and the easier the classification.

intra-class variance Similar to inter-class variance, but describing variance among objects within the same class. The smaller the variance, the easier the classification and the pose estimation.

object signature In the context of Object Categorization (OC) a signature is often a vector in a high-dimensional *Hilbert space*. It is a numerical representation of the visual appearance of an object and allows for easy comparison to other objects' signatures using, for example, the dot-product, L1-norm, L2-norm, or min-operation. Machine learning algorithms like Support Vector Machines(SVMs) and decision trees aim to separate classes by segmenting this signature space. Borders (*e.g.* hyperplanes) are called decision boundaries. In the context of Instance Recognition (IR), graph-representations are commonly used as signatures to allow for better discrimination. This comes at the cost of more complicated and computational more expensive metrics and similarities.

polyseme A single word which has multiple meanings, e.g., orange being the fruit and the color or nut referring to the fruit and the hex-nut.

RGB-D Data representing the three color channels (Red, Green, Blue) together with the distance of the point to the sensor (Depth).

RGB-D sensor A sensor which records RGB-D data. In this thesis we use active sensors which project a structured light pattern onto the scene and analyze the scattered return for the depth calculation.

supervised method Methods or predictor models which use training data where the input as well as the desired response is known (labeled data). Using the training data, the model seeks to predict reasonable output for new input.

unsupervised method Similar to supervised methods with the difference that training data is not available or not labeled (e.g. desired responses are not known).

variance Used as a qualitative measure to describe how much the visual appearance of objects differ.

Acknowledgments

I HAVE TO THANK MANY PEOPLE which helped me bringing about this thesis. First of all, I shall thank my supervisors Prof. Dr. Florentin Wörgötter and Dr. Frank Guerin for their countless valuable advices, many hours of important discussions, and the creation of such a friendly, creative, open-minded working environment. My hearty thanks to my former and current vision colleagues: Dr. Jeremie Papon, Dr. Alexey Abramov, Dr. Eren Erdal Aksoy, Simon Reich, Timo Lüddecke, Simon Stein and Fatemeh Ziaeetabar for the many invaluable advices. Special thanks to Jeremie: Without our permanent discussions, the exchanges (sometimes quite vivid), the time we spent together as well as the many collaborations, it would have been much less fun and I would have hardly made it that far. I'd also like to thank our two robot-guys Dr. Tomas Kulvicius and Mohamad Javad Aein for putting my algorithms to practical use in our robotic applications.

A sincere thank-you to all of the other current and former group-members for creating such a pleasant atmosphere: Dr. Alejandro Agostini, Dr. Minija Tamosiunaite, Dr. Poramate Manoonpong, Dr. Christian Tetzlaff, Dr. Sakyasingha Dasgupta, Dr. Yinyun Li, Dr. Xiaofeng Xiong, Dr. Christoph Kolodziejski, Dr. Irene Markelic, Jan-Matthias Braun (thanks for revealing the secrets of Linux to me), Michael Fauth, Martin Biehl, Dennis Goldschmidt, Timo Nachstedt, Johannes Auth, and Juliane Herpich. A big appreciation also to our staff behind the scenes Ursula Hahn-Wörgötter, Nicole Rehbein, Thomas Geiling, Sabine Huhnold, and Elke Zech for having an open ear, being always helpful, and organizing the nice group retreats and summer-schools.

My dear friends Florian Linder, David Schmitz, Saskia and Thorsten Karbach: I am very grateful to have met you. Thanks for the good time we always had together and for being there when needed. Johannes Kaschel, Nils and Julia Brökers: Thanks for taking good care of me when I first arrived in Göttingen and for being so welcoming. Without you my thesis would have probably dominated my days to much.

My deep-felt gratitude goes out to my mother Meike Schoeler and to Helwig Schäfer, my grandparents Helga and Erwin Bretthauer, and my brother Hanno. Thanks for always being there. Thanks for all the priceless advices you gave me. Thanks for the constant support you lend to me.

Finally, I would like to give my deep-felt thanks to my wife Elisa Schoeler. Your motivation, your geniality, your genuine love, and your company were my driving force. I could have never made it here without you.

Thank you all, so very, very much!

Markus Schoeler
Göttingen, 2015.

It is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility.

Moravec, *Mind Children*, 1988.

1 Introduction

In July 1966, Seymour Papert of MIT proposed the summer vision project to students which started with the following sentence [12]: "The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system."

Nobody at that time assumed that 50 years later this "summer project" still occupies a significant number of researchers world-wide and is far from being solved. Since then many research areas in "computer vision" or "machine vision" have emerged and many applications been named which shall benefit from automated interpretation and "understanding" of sensory inputs by computer systems. One of the core applications are autonomous robotic systems. Here we are especially interested in systems which are able to operate in unstructured environments[1] like households, disaster-zones, and unknown territories.

Such agent's ability to make decisions and to plan actions very much depends on their ability to perceive their surrounding and process the complex visual stream into meaningful entities like available objects, potential dangers, landmarks for localization, or goals. Interestingly, the process of visual understanding of our surrounding is so fundamental to us, humans, that it is even difficult for us to name the steps which lead to our perception of a scene or an object. In this thesis we advocate the idea that object perception is in two ways inseparably intertwined

[1] The term unstructured describes an environment which does not need to be well defined, unlike the operational workspace of robots in industrial assembly with predefined tool and object locations.

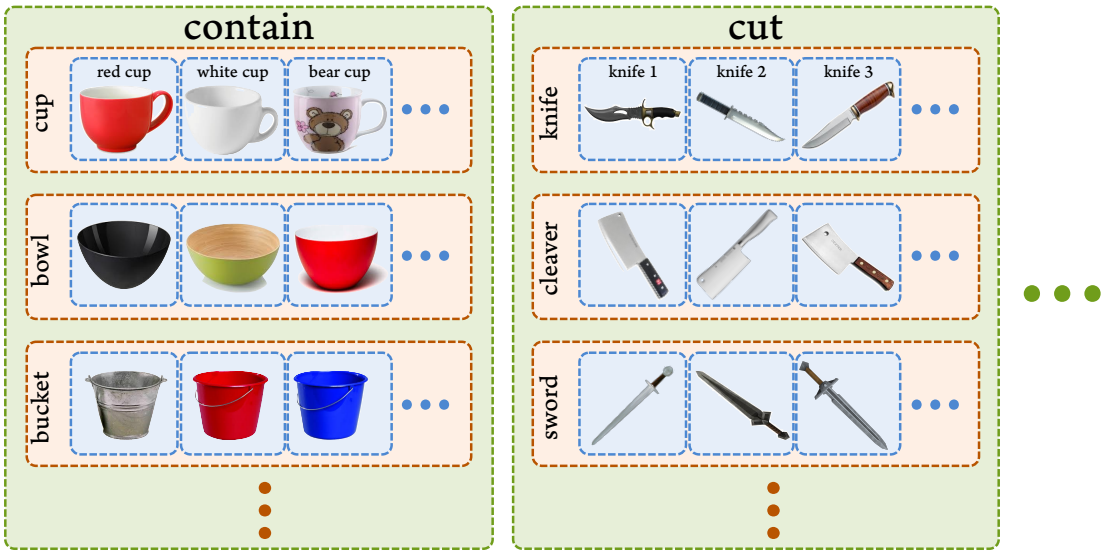


Figure 1.1: Three levels of perception. Each object is a unique *instance* (blue boxes). The combination of many *instances* results in *categories* (red boxes). Again combining several *categories* which allow for a certain *functionality* results in the third perception level, that of *function* (green boxes). Note, the mapping from an object to the function label is not necessarily surjective as shown here. An object can possess multiple *functions* at the same time as shown in Fig. 3.2.

with the concept of actions, planned tasks, and set goals (this is similar to the idea which led to Object-Action-Complexes [13–15]).

First, the level of granularity or specificity of object-perception is dictated by the task at hand: (1) "Filling the blue-and-white-striped mug", (2) "filling *a* cup" or (3) "filling *something* for transporting a liquid" all require a different perspective on the involved objects. (1) requires a specific object (there is only one existing in the world). (2) requires a more general understanding of the *category* "cup" and (3) finally defines an even more general set of objects solely by their purpose or use.

Second, an object is perceived in different ways depending on the action-context : A hammer, usually used for hitting a nail, can also be perceived as a borer and used for drilling a hole into soil for planting seeds (using the handle as the tip). The human ability to think of makeshift replacements, eventually, led to tools being specifically designed to possess multiple functionalities (*i.e.*, so-called multi-tools): A swiss army knife is designed to be used as a screwdriver, a corkscrew, a knife, a bottle-opener, and so on. Some examples of improvised tools and designed multi-tools are shown in Fig. 3.2.

In line with the literature, we denote the level of perception for the problems (1) and (2) as the *instance* and *category* level, respectively [16]. We promote that problem (3) requires and defines an even more general level of perceiving objects, that of *functionality*.

The effect of the three perception levels on object labeling is shown in Fig. 1.1.

2

Overview of the Thesis

For easier orientation we define color-codes in this thesis. Color-codes are used for the level of perception in figures and visible at the outer side of each page: For *instance* perception we use blue, for *category* perception red, and for *function* perception green. General chapters, including the introductory as well as the closing chapters, have a gray color-code.

In Chapter 1 we already mentioned that we consider object perception as a stack of three task-dependent levels: The *instance*, the *category*, and the *function* level.

In this chapter we give an overview of the problems arising in the three perception levels and which are treated in this thesis. Figure 2.1 illustrates the general problem areas and the terminology we use. Figure 2.2 shows a more detailed overview of the problems and contributions in this thesis. The chapter ends with Section 2.1 which lists the research published in the course of this work with individual contributions.

In Chapter 3 we start to describe the subproblems which arise from the three perception levels. Although we introduce and discuss important related works in the broader field around object perception, detailed and specific reviews of the literature are in the respective papers throughout this thesis.

Chapter 4 deals with classifying objects at the first level - the *instance* level - also known as Instance Recognition (IR). It introduces a paper about automatically training an IR system which consists of two parts. First, a method for automatically generating training sets for the work environment of an agent using its own sensors. Second, a classifier which is able to create highly discriminative signatures of objects in order to achieve high classification accuracy.

	Why needed for artificial systems?	Instance Perception	Category Perception	Function Perception
Object Partitioning	For Function Perception which requires partitioned objects.	Not needed	Not needed	Object Partitioning (OP) Treated in Sections 6.1 - 6.3
Classification	Recognition of an object. What objects do I see, have available?	Instance Recognition (IR) Treated in Chapter 4	Object Categorization (OC) Treated in Section 5.1	Object Function Assignment (OFA) Treated in Section 6.4
Pose Estimation	A precursor to object manipulation and use. How can I use the objects I see?	Pose Estimation for instances (PEI) See [11]	Pose Estimation at the category level (PEC) Treated in Section 5.2	Pose Estimation at the function level (PEF) Discussed in Chapter 8

Figure 2.1: An overview of the problem areas treated in this thesis.

Chapter 5 deals with perception at the *category* level. While the first part (Section 5.1) focuses on the same problem of training-set generation and classification, it does so for *categories*. The second part (Section 5.2) introduces a paper in which we describe how to do concurrent Object Categorization (OC) and Pose Estimation at the category level (PEC) using synthesized training data with a Deep Convolutional Neural Network (DCNN).

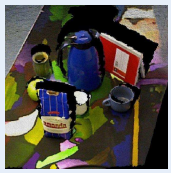
Chapter 6 introduces the third level - the *functionality* level - of object perception which aims at assigning *function* to objects rather than *category* labels. Object-parts play a crucial role here. First, they help to reduce the huge variance between objects for one functionality and, second, parts possess a function in their own right. For example, many objects for cutting consist of two parts: one handle (for grasping) and one blade (for the cutting). Therefore, we first introduce several algorithms which aim at segmenting scenes into objects and objects into parts (Sections 6.1, 6.2, and 6.3), and second, we show how to retrieve *functionality*, at both the part and the object level (Section 6.4).

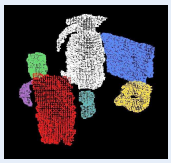
Finally, in Chapters 7 and 8 we summarize our findings and conclude with a discussion in respect to our goal and promising directions of future extensions.

Problem (Chapter 4): IR
Training data for environment specific objects needs to be recorded

Contribution:
Automatic training set recording by extracting unknown objects

Paper:
[1] Fast Self-Supervised Online Training ...

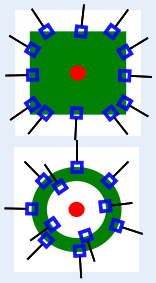




Problem (Chapter 4): IR
Similar objects need to be discriminated

Contribution:
Novel Radial key-point orientation scheme

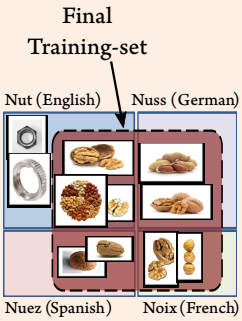
Paper:
[1] Fast Self-Supervised On-line Training ...



Problem (Section 5.1): OC
Large amounts of labeled training data are required for OC

Contribution:
Automatic training set generation using multiple language searches

Papers:
[2] Automated generation of training sets ...
[3] Unsupervised generation of context-relevant training-sets ...



Problem (Section 5.2): OC & PEC
Concurrent Classification and Pose estimation for categories

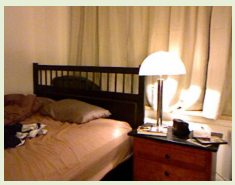
Contribution:
Synthesizing scenes for training a DCNN for categorization and pose estimation

Paper:
[4] Semantic Pose using Deep Networks Trained on ...

Problem (Section 6.1): OP
Computational expensive processing in 3D

Contribution:
The VCCS algorithm for combining similar points into entities called supervoxels

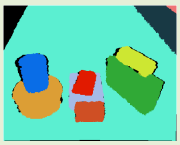
Paper:
[5] Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds



Problem (Section 6.2): OP
Segmentation of objects in a scene

Contribution:
The LCCP algorithm which uses concavities to separate objects

Papers:
[6] Covexity based object partitioning for robotic ...
[7] Object Partitioning using Local Convexity



Problem (Section 6.3): OP
Segmentation of parts for objects

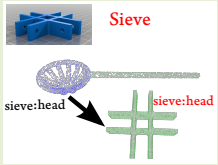
Contribution:
The CPC algorithm which uses local concavities to induce cuts through objects

Paper:
[8] Constrained Planar Cuts - Object Partitioning for Point Clouds

Problem (Section 6.4): OFA
Assigning function to novel objects and their parts

Contribution:
Algorithm which can assign functions to objects based on their part constellation

Paper:
[9] Bootstrapping the Semantics of Tools ...



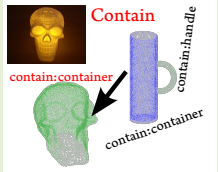


Figure 2.2: A detailed list of the problems and the contributions in this thesis sorted by the level of perception: Instance, category, and function (top, middle, bottom).

2.1 List of Publications and Contributions

The following is a list of my contributions to each publication described in this thesis, sorted by their order of appearance:

[1] **Schoeler, M.** and Stein, S. and Papon, J. and Abramov, A. and Wörgötter, F.: "Fast Self-Supervised On-line Training for Object Recognition Specifically for Robotic Applications", *9th International Conference on Computer Vision Theory and Applications (VISAPP)*, 2014. See page 23.

Ideas, methods, evaluation, figures, tables, main text, literature research

about **80%** contribution

[2] **Schoeler, M.** and Wörgötter, F. and Aein, M. and Kulvicius, T.: "Automated generation of training sets for object recognition in robotic applications", *IEEE/RSJ 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, 2014. See page 39.

Ideas, methods, evaluation, figures, tables, main text, literature research

about **70%** contribution

[3] **Schoeler, M.** and Wörgötter, F. and Papon, J. and Kulvicius, T.: "Unsupervised generation of context-relevant training-sets for visual object recognition employing multilinguality," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2015. See page 47.

Ideas, methods, evaluation, figures, tables, main text, literature research

about **75%** contribution

[4] Papon, J. and **Schoeler, M.**: "Semantic Pose using Deep Networks Trained on Synthetic RGB-D," *IEEE International Conference on Computer Vision (ICCV)*, 2015 (in press). See page 57.

Ideas, evaluation, network training, model in scene rendering, figures, tables, text, literature research

about **40%** contribution

[5] Papon, J. and Abramov, A. and **Schoeler, M.** and Wörgötter, F.: "Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. See page 71

Ideas, figures

about **15%** contribution

[6] Stein, S. and Wörgötter, F. and **Schoeler, M.** and Papon, J. and Kulvicius, T.: "Convexity based object partitioning for robot applications", *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. See page 81.

Master supervision, ideas, methods

about **30%** contribution

[7] Stein, S. and **Schoeler, M.** and Papon, J. and Wörgötter, F.: "Object Partitioning using Local Convexity", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. See page 89.

Master supervision, ideas, methods, text, Publishing of the algorithm within Point Cloud Library (PCL)

about **40%** contribution

[8] **Schoeler, M.** and Papon, J. and Wörgötter, F.: "Constrained Planar Cuts - Object Partitioning for Point Clouds", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. See page 99.

Ideas, methods, evaluation, figures, tables, main text, literature research

about **80%** contribution

[9] **Schoeler, M.** and Wörgötter, F.: "Bootstrapping the Semantics of Tools: Affordance analysis of real world objects on a per-part basis," *IEEE Transactions on Autonomous Mental Development (TAMD)*, 2015 (in press). See page 111.

Ideas, methods, evaluation, figures, tables, 90% text, literature research

about **80%** contribution

[10] Papon, J. and **Schoeler, M.** and Wörgötter, F.: "Spatially Stratified Correspondence Sampling for Real-Time Point Cloud Tracking", *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2015. See page 147.

Ideas, methods

about **15%** contribution

ICRA is considered the premier robotics venue having an h5-index[1] of 64. It ranks before IJRR and IEEE Trans. Robot. CVPR is the top computer vision venue with an h5-index of 128, ranking before PAMI, IEEE Trans. Image Process., ECCV, ICCV and IJCV. Furthermore, CVPR is the highest ranking conference across the field of Engineering & Computer Science, just placed behind the journals Nature Nanotechnology and Nature Photonics. ICCV ranks number three among the computer vision conferences with an h5-index of 68.

The research leading to this thesis was supported with 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, grant agreement no. 269959, Intellact, and grant agreement no. 600578, ACAT.

[1] The h5-index is the h-index for articles published in the last 5 complete years. It is the largest number h such that h articles published in 2010-2014 have at least h citations each. All metrics were taken from Google Scholar (https://scholar.google.de/). Accessed on 6th of July 2015.

3

Problem Decomposition and State-of-the-Art

In Chapter 1 we already introduced the three perception levels: The *instance*, *category* and *function* levels. Objects are perceived at these three levels depending on the task a human or robot is set to solve. In some tasks specific *instances* are required, in other tasks, we do not care about specific objects but more about objects of a specific *category*, or which allow for a certain *functionality*.

Two fundamental problems arise in all tasks: First, how can one recognize perceived objects in a scene, thus assign them a class[1]? Second, what is the pose/orientation of the recognized objects in the scene in order to apply learned motor behavior to use them? For example, when filling a cup you should not hold it upside-down. If you want to sit down on a chair, you cannot do so from the backrest's side. Pose of a rigid body in 3D is described by 6 Degrees of Freedom (DoF). There are three translational and three rotational DoF. The problem of finding the transformation between an object's intrinsic reference frame (where actions are being defined) and the object recording in the world's reference frame (*e.g.*, room-axis aligned) is addressed by pose estimation.

For the first problem, classification, we define three subproblems: For the *instance* level, we call this Instance Recognition (IR), at the *category* level Object Categorization (OC), and at the *function* level Object Function Assignment (OFA). Depending on the perception level, we

[1] We use the term class in a general sense to distinguish between objects with different labels. Which entities are regarded different depends on the level of perception. This is different to some works in the literature which use class, *object classification*, or *generic recognition* to describe what we refer to by *category* and *categorization*.

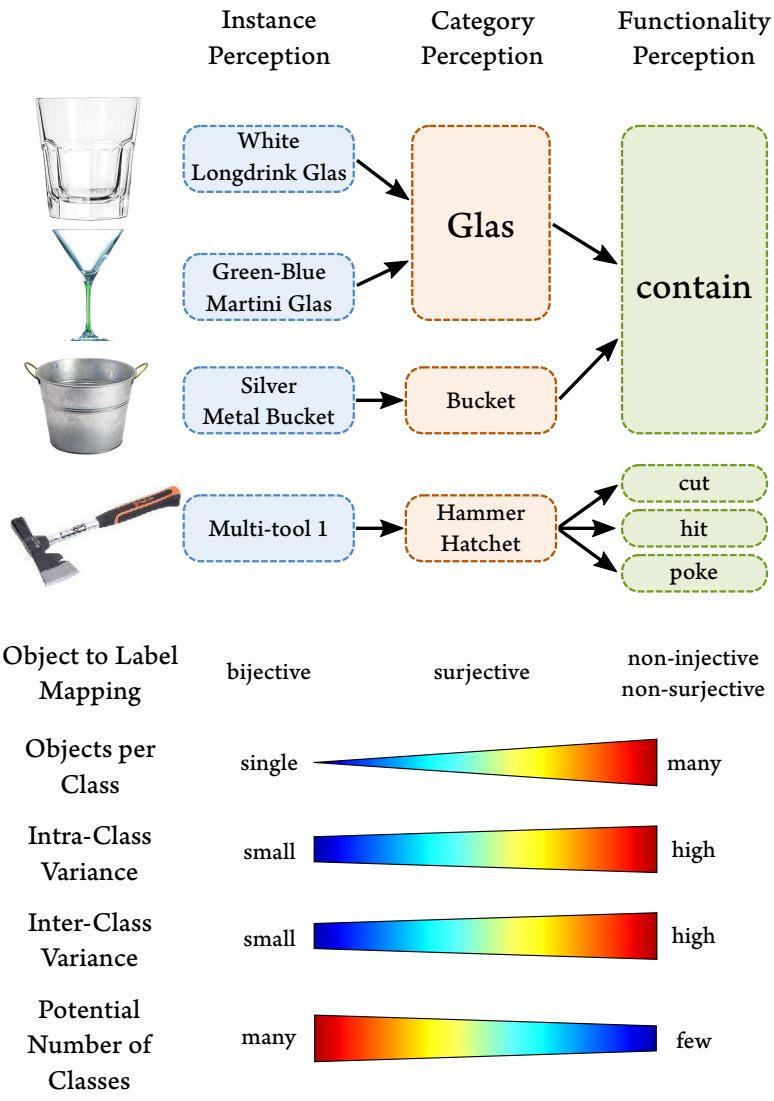


Figure 3.1: A comparison of the three perception levels. Top: Labeling for specific objects. Bottom: Effects of the perception level on the classification. Note that *functionality* is always defined in relation to *parts* of objects (handle, blade, ...).

can also define the three pose estimation problems: Pose Estimation of instances (PEI), Pose Estimation at the category level (PEC), and Pose Estimation at the function level (PEF).

Although related problems, IR, OC, OFA as well as PEI, PEC, and PEF impose different constrains on the set of algorithms used to address them. This is best explained by looking at how the different perception levels define a class. At the *instance* level a class is only a single object, at the *category* level a class represents a set of similar objects, and at the *function* level classes represent objects compatible to certain *functionalities*.

Therefore, methods for the six problems have to deal with vastly different magnitudes of intra-class variance, inter-class variance, potential number of classes, and number of objects per class (see Fig. 3.1). Intra-class variance describes the degree of visual variation of objects' appearances within a single class, whereas inter-class variance describes the variation between objects of different classes. In general the smaller the intra-class variance, the easier the classi-

fication (in the extreme case of no intra-class variance, we could simply use a nearest neighbor label assignment for the classification). The smaller the inter-class variance, the harder the classification, since the algorithms need to discriminate very similar objects with potentially complex decision boundaries.

Variance is caused by two effects: First, viewpoint, occlusion, and illumination alters a single object's appearance[2]. Second, different objects have different appearance. The latter is responsible for the increase in intra-class variance when going from the *instance* level to *categories* and to *functionalities*, because we combine more and more objects into single classes. At the same time we increase the inter-class variance as we merge similar objects and remain with fewer and (hopefully) visually better distinguishable classes.

3.1 Instance Perception

Using an agent to fetch your favorite coffee-cup among other cups requires fine-grained labeling like the *red cup*, the *bear cup*, or the *blue two handle cup*. Each unique item is called an *instance*. Two problems arise: *What* do I see and *how* to use it ? While the former requires Instance Recognition (IR), the latter requires (among others) Pose Estimation of instances (PEI).

3.1.1 Instance Recognition (IR)

To recognize individual objects, one needs to distinguish potentially very similar *instances* of a more general *cup*-category. Due to the low inter-class variance algorithms developed for IR need to be very discriminative. This is the reason why researchers tend to use very constrained algorithms like template matching [17] or matching of local descriptors which geometrically align in a known constellation using for example a Hough transform [18]. Modern approaches primarily use robust local features like the Scale-invariant feature transform (SIFT) [18], Gradient Location and Orientation Histogram (GLOH) [19], DAISY [20] or Speeded-Up Robust Feature (SURF) [21] in 2D as well as Spin-Images [22], Signature of Histograms of Orientations (SHOT) [23], Fast Point Feature Histogram (FPFH) [24] in 3D, just to name a few. An overview is given by Mikolajczyk and Schmid [19] as well as Alexandre [25] for 2D and 3D, respectively.

Besides the low inter-class variance there is another main problem in IR: Instances are most of the time specific to an environment, therefore, training data is not publicly available. This is why training data needs to be recorded before IR pipelines can be employed. As soon as objects in the environment change or new objects are introduced, this process needs to be repeated. This leads to an inflexible system. In Chapter 4 we contribute to the solution of

[2] This is the only contributer to intra-class variance in case of IR

this problem by introducing a recognition pipeline which can detect and train novel objects on-the-fly with minimal human supervision.

3.1.2 Pose Estimation of Instances (PEI)

PEI is one of the precursors to manipulation of known objects by artificial agents. It is normally solved by aligning full object models to the partial recordings of objects in a scene. This can be done by extracting local features for both models, finding matches, and identifying the transformation which aligns all features best. Algorithms used for this are, for example, Random Sample Consensus (RANSAC) [11, 26, 27] or Geometric Hashing [28]. For refinement, Iterative Closest Point (ICP) [29, 30] is usually used. It starts with the coarse pose and iteratively converges the model to the scene by minimizing the distance between pairs of closest points using a mean-squared-error cost function. In this thesis we do not not contribute to PEI. A comprehensive overview of algorithms is given in the Ph.D.-Thesis by A. Buch [11].

3.2 Category Perception

The huge number of objects in the world causes humans to group similar entities into meaningful *categories* starting in their second year [31]. This process is, at first, much facilitated by color or shape of objects and, at a later stage, by higher-level features. According to Piaget, one of the leading scientists on child development in the 60s, this *Adaptation*-process can be split into two complementary sub-processes: *Assimilation* and *Accomodation* [32]. *Assimilation* describes the mechanism by which perceived and familiar objects are sorted into existing *categories*. If the visual-impression of a new object is too different from existing *categories*, the *Accomodation*-process forms a novel *category* [33]. *Assimilation*, for example, is a powerful acquisition during a child's development as it allows transferring knowledge from a group of known objects to new objects. If you encounter a new knife and recognize it as belonging to the knife *category*, you can recall trajectories and grasping points for cutting and quickly handle it without relearning it from scratch. While *recognizing* the knife demands Object Categorization (OC), using it requires (among others) Pose Estimation at the category level (PEC).

3.2.1 Object Categorization (OC)

Being able to recognize the belonging *category* for a given object is called OC. As different objects are being combined into single classes, this can lead to complex decision boundaries in the object signature space.

While generalizability for Instance Recognition (IR) is limited to recognizing a known object under different viewpoint, occlusion, and/or illumination, OC algorithms are expected to generalize to novel objects (of known *categories*). This is why methods usually trade-off

some discriminative power for generalizability. Constrained algorithms like template matching or Hough transforms do not work well anymore, because objects in one *category* can have remarkable differences in their appearance, such that local image patches have little spatial coherence across different objects. This is why a geometric verification stage is less common in OC. Instead, objects are more frequently described by object signatures using for example Bag of Words (BoW) [34], Fischer-Vectors [35–37], or sparse coding [38, 39].

Machine learning algorithms (*e.g.*, Support Vector Machines(SVMs) [40–42], decision trees and random forests [43–47], or boosting [48]) are then used for training on signatures with known labels (supervised learning). After learning, the predictive model should be able to merge signatures of objects of the same class (by assigning same labels) and differentiate them from object signatures of other classes (by assigning different labels).

While these classical methods tend to work well for a small number of classes, they do not scale to large object categorization problems with up to 1000 classes and hundreds of thousands of *instances* like the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[3] [49]. This growth in data led to the advent of Deep Convolutional Neural Network (DCNN) architectures[4] with thousands of learnable parameters. DCNNs take a special role in that they replace the traditional pipeline <*feature extraction*> → <*signature generation*> → <*class learning*> (with fixed *feature extraction* and *signature generation* steps) by a pipeline which starts at the signal level provided by, for example, an RGB-D camera[5]. They use a stack of consecutive layers (the first is the input signal layer, the last is the output layer) with each layer being the input to the next layer. A very powerful property of DCNNs is the fact that the last layer can predict any kind of output (*e.g.*, in Section 5.2 we predict not only the *category*, but also the pose of objects). Layers are connected by neurons which are only applied in local regions of their input layer (receptive field) and share weights across image regions (using convolution). Interestingly, lower layer neurons automatically tune to local image features based on gradients (like edges and corners) whereas neurons in later layers adapt to characteristic higher-level features for *categories* [50]. Thus they learn the steps of traditional *feature extraction* and *signature generation*[6].

While DCNNs can be tailored to datasets by pure learning, they need an enormous amount of labeled training data to avoid over-fitting. While there are popular ways like augmenting training data (shifted, rotated, and/or flipped versions of training images) or drop-out [55] (randomly deactivating neurons in the network to prevent over-fitting), DCNNs are not practical if training data is scarce.

---

[3] http://image-net.org/challenges/LSVRC/

[4] A comprehensive introduction and tutorial on DCNNs for Visual Recognition by Li and Karpathy is available at http://vision.stanford.edu/teaching/cs231n/syllabus.html.

[5] A camera which records red, green, and blue as well as distance to the camera for each pixel.

[6] This is why DCNNs with stripped *Fully-Connected* layers are often used as generic signature extractors [50–54].

We contribute to solving this dilemma for learning rich models in two ways: In Section 5.1 we introduce our *TransClean* algorithm. It is able to automatically retrieve large amounts of OC training data when given a *category* name (*e.g.*, *nut*) together with a descriptive context (*e.g.*, *crack* or *delicious*). In this example *nut* has a double-meaning: It can refer to either a *food-nut* or a *hex-nut*. Downloading training data directly from large word-based image-databases like Google-Image-Search results in many irrelevant images in the training set. Using the context, *TransClean* can disambiguate the name of the *category* and retrieve task-relevant images.

In Section 5.2 we randomly generate cluttered indoor scenes in 3D. From those we can synthesize an unlimited number of training images. Compared to the *TransClean* algorithm, this approach is limited to *categories* where full 3D models are available. It quickly makes up for this disadvantage by being able to automatically annotate all objects with a full 6 Degrees of Freedom (DoF) pose. This allows for the training of methods for category level pose estimation.

3.2.2 Pose Estimation at the Category Level (PEC)

In order to execute tasks, the agent needs to determine exactly where and in which pose an object is located in a scene. For example, when filling a cup you should not hold it upside-down. If you want to sit down on a chair, you cannot do so from the backrest's side. The problem of finding the transformation between an object's intrinsic reference frame (where actions are being defined) and the object recording in the world's reference frame (*e.g.*, room-axis aligned) is addressed by pose estimation. Interestingly, we, as humans, can infer pose of an object even if we have never seen it before. For example, if you see a new object of known *category* (a cup, a knife, and so on), you immediately know how to use it. While trivial for us, artificial agents cannot accomplish this, yet.

A lot of research has been conducted to solve pose estimation for known instances in a scene (see Section 3.1.2). Here it is normally solved by aligning full object models to the partial recordings of objects in a scene. However, this can only be done with known *instances*. Doing pose estimation at the *category* level is much harder. For example, aligning two similar, but different objects usually fails. Comparing each object in a scene to a huge collections of stored models (from one *category*), in order to increase the chances of finding a good match, would be very inefficient. While the variance within a *category* as well as the complexity of the problem calls for rich models (*e.g.*, DCNNs), scarcity of training data with annotated 6 DoF pose usually prevents those models from being trained. In Section 5.2 we show how to circumvent this dilemma.

3.3 Function Perception

Describing objects by their functional properties is not a novel concept. Gibson coined the term *affordance* in his work *The theory of affordances* [56]. *Affordance* of the environment is "what it offers the animal, what it provides or furnishes, either for good or ill" [56, p. 127, ln. 11]. Properties of objects always need to be defined in relation to the perceiver. According to Gibson, *affordances* exist even if the actor is not able to perceive them. They are independent of an actors experience, knowledge, and cultural background [57]. Diverging from this view, Norman specifically includes subjective perception into the *affordance* term [58, p. 9, ln. 19]: "Affordance refers to the perceived and actual properties of a thing, primarily those fundamental properties that determine just how the thing could possibly be used."

In this thesis we abstain from using the term *affordance*, because it not only describes visually perceivable properties, but a comprehensive view on objects and environment including physical properties like weight, flexibility, being a solvent, being nutritious, conducting vibrations, and so on. Inclusion of all these properties into a framework for artificial vision is beyond the scope of this thesis. Consequently, we only determine the visually perceivable *functionality* of objects from the viewpoint of a *humanoid* agent.

Figure 3.2: Tools which possess multiple *functionalities*. Top row: Tools specifically designed to be used in multiple applications (*i.e.*, multi-tools). Bottom row: Makeshift-replacements with objects being used in a different way from what was intended by the designer.

3.3.1 Object Function Assignment (OFA)

Most work found in the computer vision literature focuses on one of the two aforementioned levels of perception (*i.e.*, Instance Recognition (IR) and Object Categorization (OC)). In this thesis we advocate the idea of adding a new even more general level to the labeling hierarchy, which we denote as Object Function Assignment (OFA). OFA ignores the concept of traditional OC-classes completely. Instead, it assigns *functionality* or usage to objects. *Categories* like cups, mugs, glasses, goblets, and bowls would consequently be combined into a functional class of *contain*; knifes, saws, cleavers into a class of *cut*. The interesting observation at this point is that labeling by verbs instead of nouns becomes much more natural for OFA as compared to IR and OC. This already shows that OFA combines objects in a very general sense going away from the traditional object description by label to an entirely different descriptive level by *functionality*. Objects at this level do not even need to have traditional OC-labels - one can think of an artistic object which is hard to describe by name, but has *functionality*. Furthermore, objects can possess multiple *functionalities* at once. A hammer-like object can be used to hit a nail or to drill holes into soft materials using the handle. In the latter example, the former hammer-head becomes the handle, and the hammer-handle becomes the borer of the improvised tool. If we attach a blade to the other end of the hammer-head, it could also be used for cutting.

It becomes evident that *function* perception is, therefore, strongly linked to object perception at the *part level*. For each *functional* context, parts have a *function* in their own right (*e.g.*, cutting tools can consist of *blades*, *handles*, a *motor*, a *cord*, a *switch*, and so on). This viewpoint from the action and part domain rather than the *category* and full-object domain allows humans to bootstrap tool usage and even - by ways of human ingenuity - create makeshift replacements for tools. This eventually led to tools being designed for various tasks (*i.e.*, so-called multi-tools). The most prominent examples are swiss army knifes which possess dozens of *functionalities*[7]. Some examples of make-shift replacements versus designed multi-tools which are better described by their *functionalities* and parts rather than a single *category* for the full-object are depicted in Fig. 3.2.

In the next Section 3.3.2 we give an introduction how to extract parts of objects, which is a prerequisite for the OFA introduced in Chapter 6.4.

3.3.2 Object Segmentation and Partitioning (OP)

The word *segmentation* is widely used to describe the process of dividing an image or a point cloud into entities. It is used for naming the processes of dividing a scene into objects (*i.e.*, *object and instance segmentation*), a scene into *categories* (*i.e.*, *semantic segmentation*), and ob-

[7] The swiss army knife "The Giant" of Wenger in Delemont, Switzerland, set the Guinness World Record of *most functions on a penknife* in 2008 with 141 different functions [59].

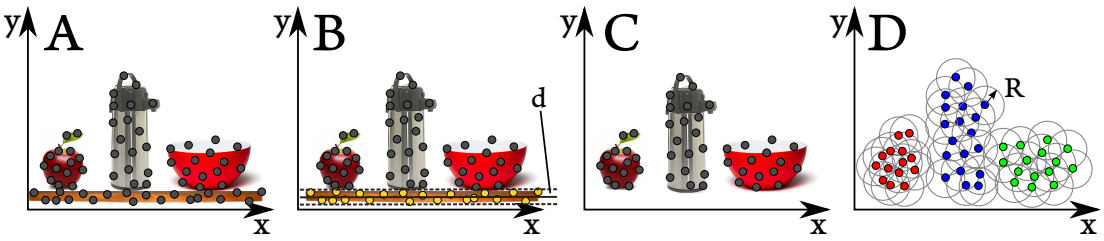


Figure 3.3: Simple object segmentation pipeline: *Ground-Plane Subtraction* and *Euclidean Clustering*. A: Unlabeled point cloud showing 3 objects on a table. B: Using RANSAC one can fit a plane (black line) and extract the table points within a distance d to the plane (yellow points). C: Point cloud after removing the table points. D: *Euclidean Clustering* clusters all points which have a distance smaller than R. This results in three separate clusters (red, blue, and green).

jects into their parts (*i.e., part segmentation*). While *partitioning* is often used interchangeably, we reserve this word in this thesis for describing the process of dividing full-objects into their parts (also known as *part segmentation*).

One can divide segmentation and partitioning methods into two groups: Supervised and unsupervised methods. The former group is often combined with classification (semantic segmentation) and generally dominated by trained object- or part-detectors using sliding-windows detectors, Markov Random Fields(MRFs), Conditional Random Fields(CRFs), or template matching methods [60–62]. While supervised methods yield good performance, they need to be tuned to known objects or *categories*.

Using very broad classes as in the case of OFA, we increase the intra-class variance in such a significant way that it becomes harder to train supervised methods in an appropriate way.

This naturally leads to unsupervised data-driven methods, which do not need training data. Thus they can be applied to arbitrary and novel objects. Some of the simpler unsupervised methods are *Ground-Plane Subtraction* as well as *Euclidean Clustering* (Fig. 3.3) which have their origin in the *Similarity* and *Proximity Gestalt-Laws* [63]. While simple, these algorithms still serve as first steps in many modern systems (this is especially true for *Ground-Plane Subtraction*). Unluckily, they do not work well if trying to separate cluttered scenes or stacked objects (do not even think about separating parts of objects with it).

In 1987 Biederman [64, 65] proposed that objects should be described as an assembly of parts. He used primitive geometric shapes like cuboids, spheres, cylinders, and tori (so-called *Geons*) as parts. Unluckily, his model is far too minimalistic for being used with real objects. Motivated by the findings of Richards and Hoffman [66, 67], who indicated that part perception in humans is much facilitated by concavities (cups), we introduce the Locally Convex Connected Patches (LCCP) algorithm for object segmentation in cluttered scenes (Section 6.2) and the Constrained Planar Cuts (CPC) algorithm for the subsequent Object Partitioning (OP) into constituent parts (Section 6.3).

3.3.3 Pose Estimation at the Function Level (PEF)

In order to use an object according to its assigned *functionality*, robots need to do Pose Estimation at the function level (PEF). While we, as humans, can easily use the objects depicted in Fig. 3.2, this is not trivial for artificial agents. Just like OFA we belief that PEF needs to be addressed at the part level. For example, if we look at a "fillable" object. It probably consists of at least one container and maybe some handles. While the container poses are important to determine the way the object can be filled, the handle orientations are needed to determine potential grasps for the object. Therefore, instead of having one pose for the full object, each part needs to have its own pose. Although we did not publish research at the time of submission of this thesis we further discuss this potential PEF approach in Chapter 8.

4

Instance Recognition (IR)

Recognition of known objects in a scene is one of the fundamental tasks a machine has to master before being of any assistance to humans. Here all objects involved in the scene are specifically known to the system. Example applications are: Agents which are employed in industrial settings, or service robots which work in household environments. An example task involves the precise description of the involved objects: Pour into my blue-white striped mug. While there is a lot of complicated reasoning involved to generate a sequence of motor commands for such a task, the objects involved are well defined. The robot is asked to specifically perform the task with a specific *instance* of the cup class: The blue and white striped mug, which is located somewhere in the scene.

In *instance* perception each object is treated as a unique class. Since different classes represent visually similar objects, the algorithms employed need to deal with a low inter-class variance compared to Object Categorization (OC) and Object Function Assignment (OFA) (see Fig. 3.1). Even worse, training data is usually scarce as objects are highly specific to the individual operating environment. This is why training data needs to be recorded before Instance Recognition (IR) systems can be employed. As soon as objects in the environment change or new objects are introduced, this process needs to be repeated.

Consequently, we now contribute to solving two problems: First, the high object similarity which we treat by introducing a discriminative algorithm. Second, the inflexibility of recognition systems due to environment specific objects, which we treat by automatic recording of training sets:

[1] **Schoeler, M.** and Stein, S. and Papon, J. and Abramov, A. and Wörgötter, F.: "Fast Self-Supervised On-line Training for Object Recognition Specifically for Robotic Applications", *9th International Conference on Computer Vision Theory and Applications (VISAPP)*, 2014 (p. 23).

The recognition system is able to segment and learn unknown objects from scenes with minimal human intervention. It employs a two-level pipeline which combines the advantages of RGB-D sensors for the segmentation of unknown objects from the scene (called object extraction in the paper) and high-resolution RGB cameras for the object learning and recognition. It starts with unsupervised object extraction, for which it uses a combination of *Ground-Plane Subtraction* and *Euclidean Clustering* (see Fig. 3.3). Given a new object the agent first takes several images from different viewpoints and extract the objects.

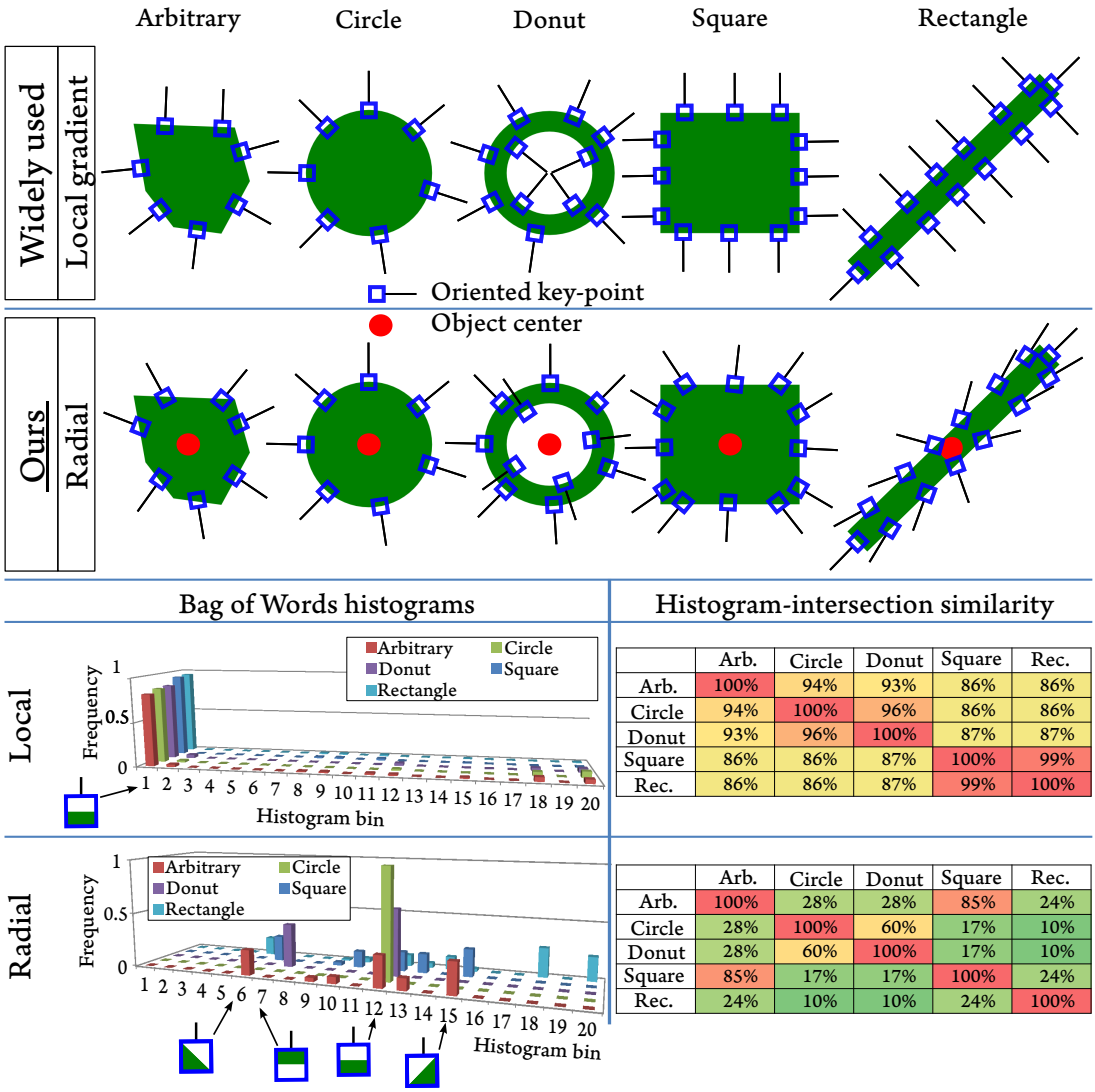
For the next step, object learning and recognition, we need to treat potentially similar objects. We now further discuss the novel *Radial* key-point orientation scheme, which we introduced in the paper. While it leads to highly discriminative object signatures, it is also robust to object rotation in the image plane as shown in Figure 4.1. In this experiment we compare the widely used Dominant Local Gradient (*Local*) [18, 43, 68–70] with our *Radial* orientation scheme on artificial objects, which only differ in their shapes. As signatures we use Bag of Words (BoW) histograms [34]. To determine similarity between two histograms, we use histogram-intersection, which is defined as:

$$\mathcal{H} = \sum_{i=1}^{N} \min(h_i, k_i), \tag{4.1}$$

with h and k being L1-normalized N-dimensional histograms:

$$\sum_{i=1}^{N} h_i = 1 \qquad \text{and} \qquad \sum_{i=1}^{N} k_i = 1. \tag{4.2}$$

As shown in Fig. 4.1, *Local* orients key-points always in direction of the dominant local gradient, thus important shape information is lost. Therefore, the BoW-signatures cannot discriminate between the objects (high histogram-intersection similarity between different objects). This is not the case for proposed *Radial* orientation. Consequently, we are able to outperform state-of-the-art algorithms, some even using full 3D information from multiple recordings. We additionally show that our proposed pipeline can easily train environment specific objects with minimal human supervision.



	Arb.	Circle	Donut	Square	Rec.
Arb.	100%	94%	93%	86%	86%
Circle	94%	100%	96%	86%	86%
Donut	93%	96%	100%	87%	87%
Square	86%	86%	87%	100%	99%
Rec.	86%	86%	87%	99%	100%

	Arb.	Circle	Donut	Square	Rec.
Arb.	100%	28%	28%	85%	24%
Circle	28%	100%	60%	17%	10%
Donut	28%	60%	100%	17%	10%
Square	85%	17%	17%	100%	24%
Rec.	24%	10%	10%	24%	100%

Figure 4.1: Comparing *Radial* and *Local* orientation schemes.
Top: Local features, like SIFT, capture the local appearance of an image at key-point locations. They only "see" a key-point's neighborhood (denoted by the blue squares) and in the key-point's reference frame (denoted by the black lines). To make the description invariant to object rotation in the image plane, *Local* orients key-points in direction of the dominant local intensity gradient (green-to-white transition). Our *Radial* orientation scheme achieves invariance to rotation by orienting key-points along the radial direction away from the object's center.
Bottom-left: Binning the features to histograms using the BoW approach. The blue boxes next to the bin numbers show the local appearance of the stereotypical feature (so-called visual word). Because *Local* oriented key-points are all pointing along the dominant local gradient (green-to-white transition), local features extracted at those key-points are all the same, thus represented by the first bin. *Radial* oriented features, on the contrary, do not orient features along the local gradient. This leads to diverse histograms with different bins being used.
Bottom-right: Pair-wise comparison of object histograms using the histogram-intersection similarity from Eq. (4.1). The high similarity of all objects for *Local* confirms that it is not able to discriminate shapes. *Radial* can discriminate the shapes and assign meaningful similarity scores. Circle and donut are, for example, found to be more similar (60 %) than circle and square (17 %).

Fast Self-Supervised On-line Training for Object Recognition specifically for Robotic Applications

Markus Schoeler, Simon Christoph Stein, Jeremie Papon, Alexey Abramov, Florentin Wörgötter
Georg-August University of Göttingen, III. Physikalisches Institut - Biophysik
{*mschoeler, scstein, jpapon, abramov, worgott*}*@physik3.gwdg.de*

Keywords: Object recognition, On-line training, Local feature orientation, Invariant features, Vision pipeline.

Abstract: Today most recognition pipelines are trained at an off-line stage, providing systems with pre-segmented images and predefined objects, or at an on-line stage, which requires a human supervisor to tediously control the learning. Self-Supervised on-line training of recognition pipelines without human intervention is a highly desirable goal, as it allows systems to learn unknown, environment specific objects on-the-fly. We propose a fast and automatic system, which can extract and learn unknown objects with minimal human intervention by employing a two-level pipeline combining the advantages of RGB-D sensors for object extraction and high-resolution cameras for object recognition. Furthermore, we significantly improve recognition results with local features by implementing a novel keypoint orientation scheme, which leads to highly invariant but discriminative object signatures. Using only one image per object for training, our system is able to achieve a recognition rate of 79% for 18 objects, benchmarked on 42 scenes with random poses, scales and occlusion, while only taking 7 seconds for the training. Additionally, we evaluate our orientation scheme on the state-of-the-art 56-object SDU-dataset boosting accuracy for one training view per object by +37% to 78% and peaking at a performance of 98% for 11 training views.

1 INTRODUCTION

Creating recognition systems which can quickly adapt to new and changing environments is not only a challenging but also highly desirable goal for the machine vision community. Solving this goal is especially important for creating machines (robots), which are able to assist humans in their daily life, as this task requires robots to interact with a multitude of objects it may encounter in a household. This, in turn, depends on successful detection and recognition of objects relevant for potential actions. Unluckily object recognition still remains one of the hardest tasks in computer vision, which leads to failures in todays robotic applications (Szeliski, 2010). One reason is that classification performance scales badly with the number of trained classes, which prohibits training the recognition system of a robot to deal with all possible objects it may encounter. One way to solve this problem is to reduce the objects to the most likely classes for a specific environment (a robot working in a kitchen will probably not need the knowledge about a hay-fork). However, this inevitably limits the robot to the most probable classes from the designers point of view. Furthermore recognizing specific instances (like the red coffee cup) is not possible. We, on the other hand, want to pursue a different path. We want to create a robot which is able to do quick, automatic and robust learning from scratch, enabling it to adapt to new or changing environments and only learning objects it encounters. Consequently our system needs to deal with the following problems in the training stage:

[1]

T1 Automatic detection and extraction of object candidates from the scene without prior object knowledge.

T2 Automatic training set generation with minimal human intervention.

T3 Dealing with a training set which is as small as possible and preferably just made of one observation per object (users should not spend their time rearranging objects for the robot to generate a large training set).

T4 Quick training of the recognition system.

For the recognition stage the system needs to deal with additional problems:

R1 Quick and robust recognition of objects in a scene (especially dealing with different distances, poses and occlusion of objects).

R2 Determining the 3D coordinates of all objects for subsequent manipulations.

We address these issues by providing:

- A new two stage vision pipeline combining low resolution 3D information for object detection and high resolution 2D information for object recognition. 3D information is needed to make extraction of unknown objects on textured background possible (see Section 3.1). In addition using a high-resolution camera does significantly improve object recognition due to the much higher quality visual information as we show in section 4.2.
- A novel orientation scheme for local keypoints, denoted as **Radial**, which is rotation invariant but includes information about the object shape and thus making object signatures much more discriminative. We show that it outperforms state-of-the-art orientation schemes on two benchmarks in section 4.2 and 4.3.
- A fusion of two classifiers using Gray-SIFT (Lowe, 2004) and a simple local color feature (**CyColor**), which is based on the hue and saturation channels of the HSV-colorspace. This combination, called **Fused**, is not only much faster to extract than color versions of SIFT, but also significantly boosts recognition performance on the benchmarked datasets.

This enabled us to build a system which works online and highly automatically. It starts completely untrained, continues with fully automatic object extraction and leads to reliable object recognition.

2 RELATED WORK

Although there are many recognition systems tackling some of the aforementioned problems, only few of them work fully automatic starting without object knowledge and with minimal human intervention. The reason is that most systems which try to extract objects from 2D images already need a trained classifier or rely on video streams and human manipulation to extract moving objects (Gall et al., 2011; Schiebener et al., 2011; Welke et al., 2010; Zhou et al., 2008). While there are methods which use a trained classification algorithm to semantically segment static images (Lai et al., 2012; Vijayanarasimhan and Grauman, 2011), few of them can extract unknown objects, like in (Iravani et al., 2011) where the authors threshold the spatial density of SIFT features or in (Ekvall et al., 2006) where a background subtraction algorithm is employed. Unfortunately both systems have their drawbacks. In the first case objects can only be placed on texture free ground and in the second case training requires a pick and place-back action by a human supervisor, thus being not fully automatic (see problem **T1** and **T2**). Furthermore, using just 2D images will not enable the robot to infer the absolute position of an object in the room, thus rendering it helpless when trying to execute an action and failing at problem **R2**.

Two other good approaches are presented in (Schiebener et al., 2011) and (Welke et al., 2010). The authors of the first work extract objects by physical robot interaction. Features are being tracked during the manipulation and simple geometrical models (planes and cylinders) are fitted to the point clouds for building object models. This method needs objects which are textured for reliable feature matching as well as objects which can be described by planes and cylinders. Furthermore, the robot needs to move all objects it encounters for training as well as for recognition, which dramatically slows down the system. In the second work objects are put into the hand of the robot and multiple images of the object are acquired while turning it. Since objects have to be segmented from the background using a stereo camera, problems with untextured objects or objects similar to the background emerge. Also holding an object in the hand can occlude important parts for the training, especially for small objects like the pen we use in our experiments.

To compare object recognition pipelines, researches often rely on publicly available benchmarks like the *RGB-D Object Dataset* (Lai et al., 2011) or the *KIT ObjectModels Web Database* (Kasper et al., 2012). We did not use them, because results for comparison are only available for turntable recordings, where objects are placed in the same spot and recorded from different inclinations. This is a very constrained scenario as objects are always placed upright and in-plane rotation is minimal. Instead, we used the SDU-dataset (Mustafa et al., 2013), which consists of single objects in arbitrary poses, but in a fixed distance and without occlusion. Robots, however, specifically also face objects in random distances and with occlusion, while working in human environments. Therefore, we recorded a new publicly available benchmark based on cluttered, high-resolution scenes with multiple objects partially occluding each other in random distances and poses[1]. This benchmark has been created using our proposed object detection pipeline.

[1] http://www.dpi.physik.uni-goettingen.de/~mschoeler/public/42-scenes/

3 METHODS

To automatically detect, extract and recognize objects in the scene, and thus solving problems **T1** and **R1**, we implemented a vision system which consists of two sensors:

1. RGB-D sensor for **object detection and extraction** (Section 3.1).
2. High-resolution 2D camera for the **object recognition** (Section 3.2).

Starting at an untrained recognition system the robot makes use of 3D information provided by the RGB-D sensor to automatically extract the object in front of it. Hereupon the vision system creates a mask and warps it to the reference frame of the high-resolution camera, takes an image and saves it for the training. The only job of the human supervisor is to actually tell the robot the names of the encountered objects, which addresses problem **T2**.

3.1 Object detection and extraction

All data from the RGB-D camera is processed in the form of point clouds. Creating object masks is done in the following way utilizing functions from the point cloud library (Rusu and Cousins, 2011):

1. The point cloud (see Figure 1 A and B) is downsampled for faster processing using a voxelgrid-filter.
2. The groundplane is subtracted (see Figure 1 C and D) by using a RANSAC plane fit to the voxelized cloud and deleting the respective inliers (This leaves a set of disconnected object candidates in our cloud, see Figure 1 C and D).
3. An Euclidean clustering scheme with a fixed distance threshold is applied to the cloud and all voxels within a cluster are treated as belonging to one object.

For all experiments a voxel resolution of 5 mm, a groundplane separation threshold of 5 mm and a clustering threshold of 4 cm have been used. The resulting labeled voxel cloud is then projected onto the high-resolution camera frame (see Figure 1 F), and for each indiviual cluster a 2D mask is created using the positions of the projected points belonging to that cluster. Since the number of projected voxels for one object is much smaller than the actual pixel count on the high-resolution image covering the object (due to the difference in resolution), we extend each projected voxel on the image by the average distance to the nearest neighboring voxel with the same label. This allows us to create a full mask for each object in the high-resolution image instead of just having a sparse set of pixels from the projection. Note that this simple scheme can provide us with fast, robust and accurate segmentation even for scenes which are cluttered in 2D or with textured background, as long as the visible parts of the objects are not touching in 3D space. Since we already possess complete 3D information for all objects, unlike systems which are working in 2D solely, we automatically solved problem **R2** as well.

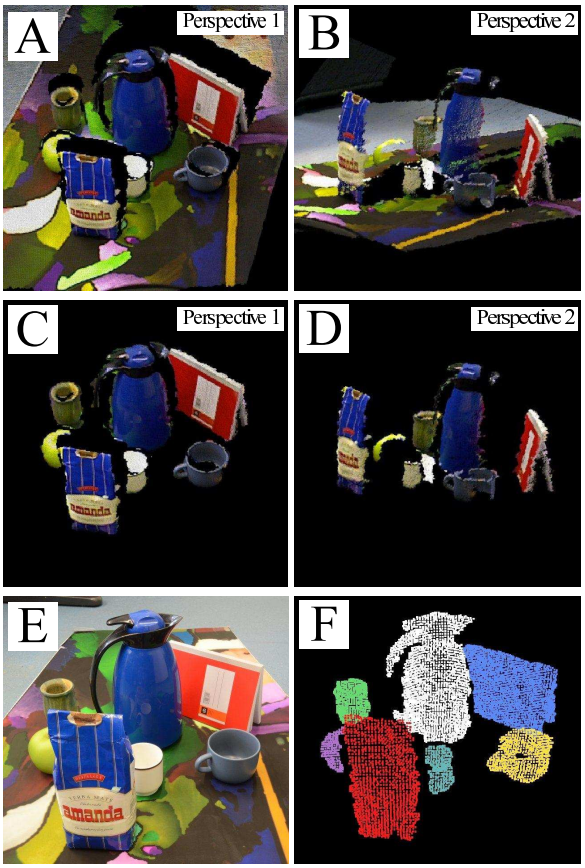


Figure 1: Process chain for extracting objects from the scene. A and B: The acquired point cloud from the RGB-D camera from different perspectives. C and D: The point cloud after groundplane subtraction. E: High-resolution camera image. F: Projected voxels on high-resolution image.

3.2 Object recognition

We implemented two recognition pipelines to incorporate full RGB information. One based on color versions of SIFT and a faster version fusing two disjoint classifiers: Gray-SIFT with a three dimensional **CyColor** feature. This combination will be denoted

as **Fused**. We chose SIFT features as they are considered state-of-the-art and are widely used in recent works of object recognition (Silberman and Fergus, 2011; Zhou et al., 2010; Van de Sande et al., 2010; Binder et al., 2011; Bo et al., 2011). In all cases a bag-of-words algorithm with k-means clustering (Csurka et al., 2004) is employed to generate compact signatures for the objects. We use all descriptors of up to 5 images per object for the vocabulary generation (about 4000 to 30000 for all objects). We cluster them to 300 visual words using k-means and generate signatures by binning each descriptor to the nearest visual word in L2-distance. The resulting signatures are used with one-versus-rest support vector machines (SVM) (Vapnik, 1998) using a histogram intersection-kernel (Barla et al., 2003) for the classification.

3.2.1 Radial orientation scheme

While we leave the SIFT descriptors untouched, we do adapt the detector step (determining the location, size and orientation of the keypoints) to leverage on the additional information provided by the first part of our pipeline 3.1. Keypoint locations are placed on a regular grid within each object mask with a stepsize of $\Delta D/d_{Step}$, with d_{Step} being a fixed number and ΔD being the diagonal of the mask size ($\Delta D = \sqrt{\Delta h^2 + \Delta w^2}$, Δh and Δw denoting the height and width of the mask's bounding box, respectively). In our experiments a value of 14 for d_{Step} yielded a good trade-off between classification performance and speed. An overview of how we are locating the keypoints can be seen in Figure 2.

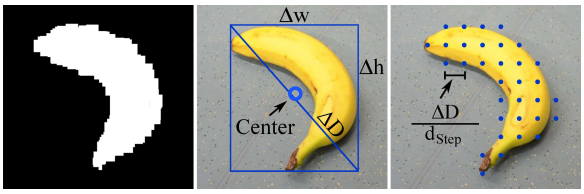


Figure 2: Defining keypoint location and object center. Left: Extracted Object mask, Middle: Determined object size and center, Right: Keypoint locations.

For each location we extract SIFT features on four different scales $\frac{\Delta D}{80} 2.5^l (l = 0,1,2,3)$. Using four different sizes makes our signatures more robust to small errors in object size estimation in case of occlusion and is a common technique in the literature (Gehler and Nowozin, 2009; Bosch et al., 2007b). In contrast to the aforementioned works we are not using a fixed scale but instead scale our SIFT features with the dimension of the mask, which makes our classification scheme completely robust to scale variations, even for unknown object, which addresses problem **R1**.

As the orientation of the keypoint decides, if the resulting signatures are invariant to in-plane rotation, we want to briefly discuss two popular approaches found in the literature: **Local** gradient and **Fixed** orientation. The **Local** gradient scheme orients features along the dominant local brightness gradient of the image patch around each keypoint. This is by far the most widely used orientation scheme (Lowe, 2004; Zhou et al., 2010; Silberman and Fergus, 2011; Bosch et al., 2007b; Bosch et al., 2007a) as it makes image signatures invariant to in-plane rotation. This unfortunately sacrifices discriminative power (Calonder et al., 2010), as important information about the object shape, encoded in the orientation of the dominant local gradient, is lost. Consequently an important cue, describing the object shape, is missing. The **Fixed** orientation scheme on the contrary orients all keypoints in a fixed direction (Calonder et al., 2010; Bay et al., 2008), thus incorporating the shape information into the signature and as a result making it more discriminative. This however comes at the cost of making it variant to in-plane object rotation. To make our signatures robust to inplane-rotation, but still keeping their discriminative power, we introduce a simple, but powerful novel orientation scheme named **Radial**. For this we approximate the center of the object by determining the middle of the object mask and orient all keypoints in a radial manner, pointing away from the center. An example of the three orientation schemes is depicted in Figure 3. Note that using the **Radial** orientation scheme requires knowledge about each object's location, which we retrieve by segmenting in 3D directly.

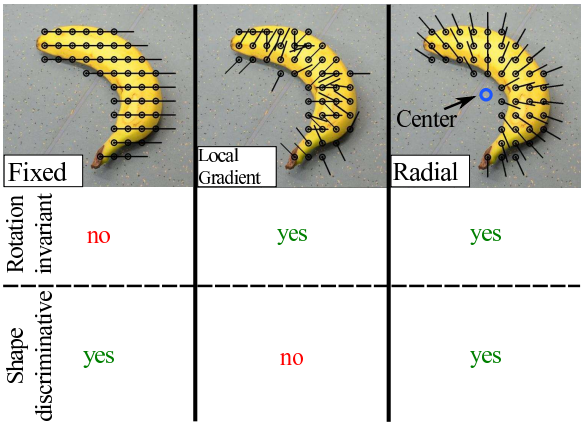


Figure 3: Comparing the three keypoint orientation schemes: **Fixed** orientation, **Local** gradient orientation and our **Radial** orientation.

3.2.2 CyColor feature

Traditionally SIFT descriptors are extracted on gray-scale images. One popular possibility to use full RGB information is to extract SIFT descriptors on all channels of an image separately and concatenating each channel's descriptors to one big descriptor (Van de Sande et al., 2010; Bosch et al., 2007a). While this generally boosts recognition performance all operations for SIFT have to be repeated on three channels, which makes the feature extraction slow. Additionally, SIFT-based feature can not deal with textureless objects as SIFT only considers gradients. To speed up the feature extraction and to cope with textureless objects, we employed a second much faster feature which we will call **CyColor**. This feature is extracted using the local pixel value at a keypoint location in HSV-colorspace. To account for the cyclic nature of the hue channel, we defined the three dimensional feature vector $\vec{f}_C$ in the following way:

$$\vec{f}_C = [sin(2\pi H), cos(2\pi H), S],$$

with H and S denoting the hue and saturation value [0,1]. Using this feature vector one easily gets rid of the problematic cyclic nature of the hue channel, while still being mostly robust to illumination variations, since we ignore the value channel. Since our **CyColor** feature itself does not cover shape of the object, we always fuse it with Gray-SIFT as described in the next section.

3.2.3 Fused classifier

There are multiple ways to fuse different classifiers (Rodriguez et al., 2007; Gehler and Nowozin, 2009), but most of these methods need a large training set to determine meaningful weights via cross-validation. We on the other side want to keep the training set as small as possible (see problem **T3**). Consequently one robust weighting scheme (when intra object variance for the individual features is unknown), is averaging the classification results. For this we train two independent classifiers: One using Gray-SIFT and one using aforementioned **CyColor** feature. Each classifier extracts features on the same keypoints. For the classification we use one-against-rest SVMs and average their scores such that class j gets the score:

$$Score(j) = \frac{1}{2}\left(Score_{CyColor}(j) + Score_{SIFT}(j)\right)$$

4 EXPERIMENTAL EVALUATION

Our main goal is to create a system which can be trained as fast as possible with minimal human intervention. Consequently we investigate how the different orientation schemes and features deal with a limited number of training samples. This is important as it shows how many observations the robot needs to robustly recognize the objects and therefore how fast the robot learns to distinguish between objects starting from an untrained system. We tested our system on two datasets: Our own publicly available scene benchmark and on the SDU-dataset which was kindly provided by the authors (Mustafa et al., 2013).

4.1 Experiment on 42-Scenes Benchmark

For the 42-Scenes Benchmark we recorded about 60 images per object in different poses and under different lighting conditions using the proposed object extraction pipeline. All objects are shown in Figure 4. For object recognition the robot was only allowed to select a fixed number of images per object from this pool for the training. After the classifier was trained, we exposed it to a new scene with several objects being placed in random orientation, distance and pose with partial occlusion up to 50%. Each object was shown in 15 scenes. Example scenes together with masks and classification results are depicted in Figure 5.

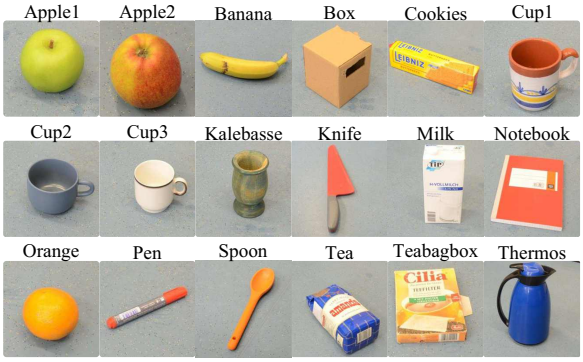


Figure 4: All objects used in the experiments.

To simulate the effect of a reduced image resolution, as one would encounter when directly using the RGB-D camera images for the recognition, we reduced the high-resolution images and masks from their original size $R_{full} = 2464 \times 1632$ pixels to the maximum Microsoft Kinect resolution $R_{low} = 1280 \times 1024$ pixels using bilinear interpolation. First note that the result of this operation still yields much higher quality images (less noise, sharper contrasts) as compared to the images you can retrieve with the Microsoft Kinect and second that using the depth channel limits your resolution to 640×480 pixels.

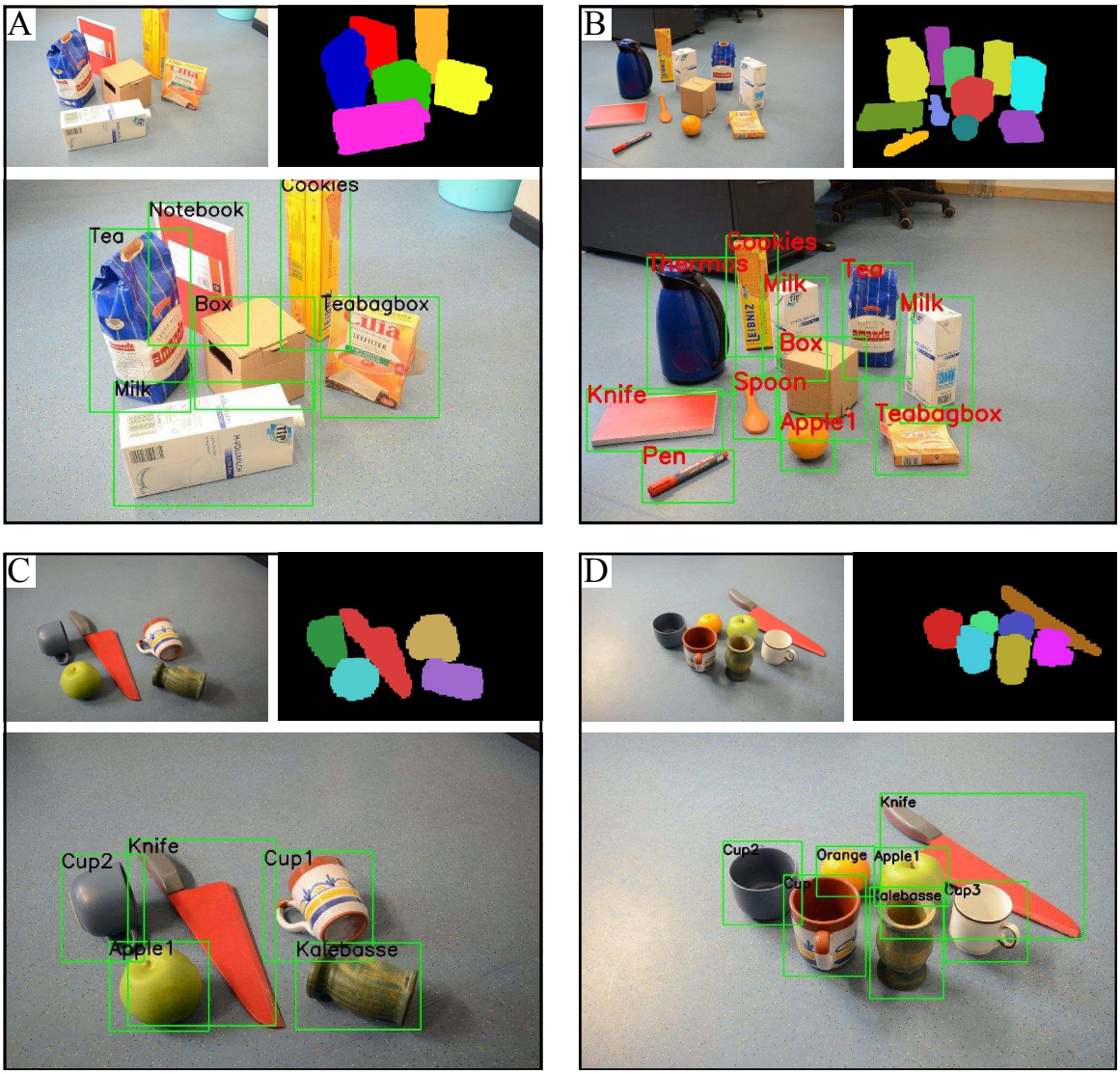


Figure 5: Four scenes from our 42-Scenes Benchmark with cluttered objects in various poses which the robot had to solve. (A-D): Top left: High-resolution scene image, Top right: Automatically extracted masks. Bottom: Example outcome using **Radial-Fused**.

We compare three different features HSV-SIFT (Bosch et al., 2007b; Bosch et al., 2007a; Gehler and Nowozin, 2009), Opponent-SIFT (Van de Sande et al., 2010) and **Fused** (Section 3.2.3), as they all incorporate color information. For HSV-SIFT, features are extracted on each of the three HSV-channels separately and concatenated to form a $3 \times 128 = 384$ dimensional vector. Opponent-SIFT does the same but on the three CIELAB channels. Altogether we compared the six following classification algorithms:

- **Fixed-HSV-Low**: Fixed keypoint detector with HSV-SIFT features on reduced scene resolution R_{low}.
- **Fixed-HSV**: Fixed keypoint detector with HSV-SIFT features on full scene resolution R_{full}.
- **Local-HSV**: Local gradient detector with HSV-SIFT features on full scene resolution R_{full}.
- **Radial-HSV**: Radial keypoint detector with HSV-SIFT features on full scene resolution R_{full}.
- **Radial-Opponent**: Radial keypoint detector with Opponent-SIFT features on full scene resolution R_{full}.
- **Radial-Fused**: Radial keypoint detector with Gray-SIFT features combined with **CyColor** features (as described in Section 3.2.3) on full scene resolution R_{full}.

To compare each classifier's performance we aver-

aged the F1-score (Hu et al., 2009) across all objects and across all scenes for 20 runs with a random draw of training images (see Figure 6). We decided to use the F1-score, as it puts equal weights on precision and recall and therefore describing the overall performance of the recognition system. It ranges from 0 to 1 (0 being worst and 1 being best) and is defined as

$$F1 = 2\frac{PR}{P+R},$$

with P and R denoting precision and recall, respectively. For comparing the processing time of the different classifiers we used an Intel i7 hexacore processor with 3.2 GHz per core.

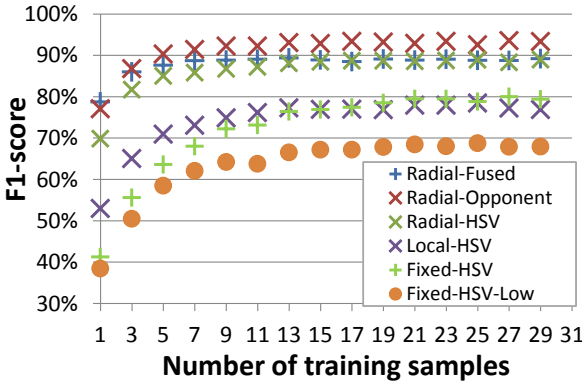


Figure 6: Averaged F1-score across all objects in all 42 scenes using different classification algorithms versus number of training images per object.

4.2 Discussion 42-Scenes Benchmark

Influence of image resolution
When comparing **Fixed-HSV-Low** and **Fixed-HSV**, the lower resolution R_{low} in general decreased the classification results significantly. While the difference for one training image per object is negligible (as both have a bad performance), an increasing number of training images shows the influence of the lower resolution. Due to the low resolution visual information of the object is lost for the SIFT features, which decreased their discriminative power. We found out, that reducing the image resolution to 640×480, the average F1-score again decreased by roughly 6% compared to the resolution R_{low}. This is consistent with the findings of Ekvall et al. (Ekvall et al., 2006) who also noted a decrease in performance when decreasing resolution. This strongly emphasizes the importance of image resolution and justifies our approach to combine a low resolution RGB-D sensor and a high-resolution RGB camera.

Influence of orientation scheme
When comparing the three orientation schemes (**Fixed-HSV**, **Local-HSV**, **Radial-HSV**), two regions can be analyzed separately:

1. Few training images per object (≤ 11 images per object)
2. Enough training images per object, such that an increase does not improve classifier performance significantly (> 11 images per object)

Few training images: Having only a few images puts very high emphasis on the signature itself. Orientation schemes which produce signatures invariant to in-plane rotation like **Radial** and **Local** are superior to orientation schemes which are fragile to object rotation like **Fixed**. Accordingly one needs only a few images to generalize to the full object using the former scheme. The distinction between **Radial** and **Local** is caused by the poor discriminative power of the **Local** scheme as described in Section 3.2.1.
Many training images: Using many images the robustness of the signature becomes less important, because different object poses are known for the training. Here the performance is more dictated by the power of the SVM, which uses all signatures as input to separate the classes. Consequently orientation schemes which lead to discriminative signatures (**Radial** and **Fixed**) work better in this regime. This is the reason why the performance of **Local** drops below the performance of **Fixed** when increasing trainingset size.

Since our orientation scheme is robust to in-plane rotation, but still discriminative (see Figure 3) , it is by far the best choice for robotic applications using local features and improves classification by 16.9% for one training image per object and about 10% for the saturated region. Please also note that the **Radial** orientation scheme is very fast to calculate (in average 4 ms per object).

Influence of feature selection
Comparing **Radial-HSV** and **Radial-Opponent** one clearly sees that the Opponent version of the SIFT descriptor is superior to the HSV version. This confirms the findings of (Van de Sande et al., 2010) where the authors compared several color extensions of SIFT. When only a few images are available the classifiers **Radial-Fused** and **Radial-Opponent** perform equally good, with **Radial-Fused** being slightly better for one training image (2%) and **Radial-Opponent** being better for more than 5 images per object (3%). The reason for the fused classifier to perform better for a very small number of training images is that the **CyColor** descriptor is fully invariant to any object rotation in 3D as long as the

same side is visible or the object color distribution does not change too much when rotating the object. Consequently we achieve results of 79% for **Radial-Fused**, 77% for **Radial-Opponent** and 70% for **Radial-HSV** when using only one image per object. In the saturated region ($>$ 11 images per object) the performance of **Radial-Fused** and **Radial-HSV** are identical, because we train already with a variety of different poses. This means that being completely pose invariant and being only invariant to in-plane rotation does not make a big difference any more. Here **Radial-Opponent** shows better results than **Radial-Fused** with an average score of 93.1%, but at the cost of being slower as shown below. Noteworthy is that using the red-green and yellow-blue channels of the Opponent-color space as base for our **CyColor** feature decreased the performance of the **Fused** classifier to 76% for one training image. The reason is, that the Opponent-color space is not invariant to lighting variations in contrast to the H and S channels of the HSV color space (Van de Sande et al., 2010). This is a severe problem when using the absolute values of these channels, whereas considering gradients, as SIFT does, circumvents this problem.

Time performance
Since we are also interested in the speed of the recognition system (problems **T4** and **R1**), we measured the time for object extraction and compared the training and recognition time for the three best scoring classifiers. The average time for the object extraction in a scene with 6 objects is 30 ms. Table 1 shows that **Radial-HSV** and **Radial-Opponent** are 3 times slower for the recognition and about 2 times slower for the training compared to the combination of Gray-SIFT and **CyColor**. This result is not surprising as the SIFT feature extraction step is by far the slowest part of the whole recognition process and consequently doing it on three channels instead of just one increases the processing time significantly. The training time grows approximately linearly with the number of images used for the training, again showing the advantage of a classification algorithm which can deal with a small number of training images. To reach the maximum performance (**Radial-Opponent** with 13 images per object) training takes 201 s. Consequently a decision has to be made, whether a high recognition rate or a fast system is preferred.

Table 1: Comparing training time and average object recognition time for **Radial-HSV**, **Radial-Opponent** and **Radial-Fused** using a single training image per object.

Classifier	Training [s]	Recognition [s]
Radial-Fused	6.8	0.23
Radial-Opponent	14.7	0.66
Radial-HSV	15.7	0.68

4.3 Experiment and discussion SDU-dataset

For the SDU-dataset (Mustafa et al., 2013) we followed the same experimental procedure as described in the paper. Since we are interested to see how well our classifier deals with a limited number of training samples, we mainly compared to Figure 7 in their paper. Figure 7 shows accuracy (mean of the confusion matrix) of our three highest scoring **Radial** classifiers on the SDU-dataset as well as results of their best scoring classifier in the paper (named SDU-Best which uses a combination of a point cloud feature and a hue-saturation histogram, (Mustafa et al., 2013)-Figure 7-pink curve). All parameters are left unchanged. As can be seen, all **Radial** classifiers are supperior to the SDU-Best classifier, altough we did not use the depth channel. **Radial-Fused** is by far the best scoring classifier (especially for few training samples) with an accuracy increase of 37% compared to SDU-Best (78% versus 41%) for a single training view. Using 11 and more training views per object, accuracy increases to 98%, which is a big improvement over the state-of-the-art as presented in the paper. This is a significant result as it shows how valuable absolute color information (only provided by the **CyColor** descriptors) is for object recognition especially for few training samples.

As stated by the authors knowing about the shape is indispensable for robust recognition. While they need 3D information to calculate shape descriptors, our **Radial** orientation scheme includes shape information in a natural way directly from 2D data.

5 CONCLUSION

This work presented a recognition system, which can adapt to new and changing environments. This is especially important for robots assisting humans in their daily life. To achieve this a system needs to deal with problems **T1** - **R2** as described in our introduction. Therefore we designed a two stage pipeline, featuring fast, automatic and robust learning of objects

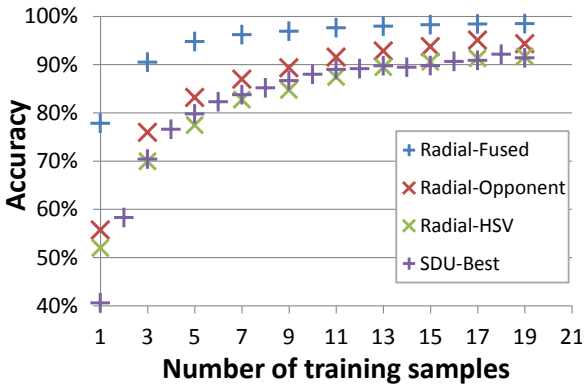


Figure 7: Averaged classification accuracy versus number of training images for the SDU-dataset.

with minimal human intervention. In the first stage (**Object detection and extraction**) the robot uses 3D information from the RGB-D sensor to automatically retrieve objects from cluttered scenes. Projecting all object masks to a high-resolution camera, we were able to provide the second stage of the recognition system (**Object recognition**) with accurate and detailed visual information.

We tested our recognition system in two scenarios: First with 18 objects with varying poses, illumination and distances in 42 scenes with partial occlusion and second on the SDU-dataset with 56 objects in arbitrary poses. The former is made publicly available. Comparing results of the SDU-Benchmark to our 42-Scenes Benchmark, one can see that our benchmark is more challenging, although the SDU-dataset uses more objects. The reason is twofold: First, we did not put any constrains on object pose, distance as well as illumination, and second, we evaluate on a collection of labeled and masked scenes which show occlusion, making the recognition more difficult. In both benchmarks our novel **Radial** orientation scheme achieved better than state-of-the-art results. This is because our orientation scheme leads to signatures which do incorporate shape information in contrast to widely used local gradient orientation schemes. Furthermore, using a simple fusion of Gray-SIFT and our three dimensional **CyColor** feature did not only speed up the recognition pipeline (7 s for the full training in our 42-Scenes Benchmark), but also boosts classification accuracy for the SDU-dataset significantly. This shows the value of absolute color information for object recognition, especially for few training samples. The combination of our **Radial** orientation scheme with our **CyColor** features leads to an improvement over the state-of-the-art on the SDU-dataset by +37% to a total of 78% for only a single training view and to 98% for 11 training views.

Acknowledgment

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

REFERENCES

Barla, A., Odone, F., and Verri, A. (2003). Histogram intersection kernel for image classification. In *Image Processing(ICIP), 2003 International Conference on*, volume 3, pages III–513–16 vol.2.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Computer Vision Image Understanding*, 110(3):346–359.

Binder, A., Wojcikiewicz, W., Müller, C., and Kawanabe, M. (2011). A hybrid supervised-unsupervised vocabulary generation algorithm for visual concept recognition. In *Proceedings of the 10th Asian conference on Computer vision - Volume Part III*, ACCV'10, pages 95–108, Berlin, Heidelberg. Springer-Verlag.

Bo, L., Ren, X., and Fox, D. (2011). Depth kernel descriptors for object recognition. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 821–826.

Bosch, A., Zisserman, A., and Munoz, X. (2007a). Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, CIVR '07, pages 401–408, New York, NY, USA. ACM.

Bosch, A., Zisserman, A., and Muoz, X. (2007b). Image classification using random forests and ferns. In *Computer Vision (ICCV), 2007 IEEE 11th International Conference on*, pages 1–8.

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: binary robust independent elementary features. In *Proceedings of the 11th European conference on Computer vision: Part IV*, ECCV'10, pages 778–792, Berlin, Heidelberg. Springer-Verlag.

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

Ekvall, S., Jensfelt, P., and Kragic, D. (2006). Integrating active mobile robot object recognition and slam in natural environments. In *Intelligent Robots and Systems (IROS), 2006 IEEE/RSJ International Conference on*, pages 5792–5797.

Gall, J., Fossati, A., and Van Gool, L. (2011). Functional categorization of objects using real-time markerless motion capture. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1969–1976.

Gehler, P. and Nowozin, S. (2009). On feature combination for multiclass object classification. In *Computer*

Vision (ICCV), 2009 IEEE 12th International Conference on, pages 221–228.

Hu, X., Zhang, X., Lu, C., Park, E. K., and Zhou, X. (2009). Exploiting wikipedia as external knowledge for document clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 389–396, New York, NY, USA. ACM.

Iravani, P., Hall, P., Beale, D., Charron, C., and Hicks, Y. (2011). Visual object classification by robots, using on-line, self-supervised learning. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1092–1099.

Kasper, A., Xue, Z., and Dillmann, R. (2012). The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics. *The International Journal of Robotics Research (IHRR)*, 31(8):927–934.

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

Lai, K., Bo, L., Ren, X., and Fox, D. (2012). Detection-based object labeling in 3d scenes. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1330–1337.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110.

Mustafa, W., Pugeault, N., and Krger, N. (2013). Multi-view object recognition using view-point invariant shape relations and appearance information. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*.

Rodriguez, B., Peterson, G., and Agaian, S. (2007). Multi-class classification averaging fusion for detecting steganography. In *System of Systems Engineering, 2007 IEEE International Conference on*, pages 1–5.

Rusu, R. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4.

Schiebener, D., Ude, A., Morimotot, J., Asfour, T., and Dillmann, R. (2011). Segmentation and learning of unknown objects through physical interaction. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 500–506.

Silberman, N. and Fergus, R. (2011). Indoor scene segmentation using a structured light sensor. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 601–608.

Szeliski, R. (2010). Computer vision: Algorithms and applications. In *Computer Vision: Algorithms and Applications*, page 657. Springer.

Van de Sande, K. E. A., Gevers, T., and Snoek, C. G. M. (2010). Evaluating color descriptors for object and scene recognition. 32(9):1582–1596.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley, 1 edition.

Vijayanarasimhan, S. and Grauman, K. (2011). Efficient region search for object detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1401–1408.

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

Zhou, F., Torre, F., and Hodgins, J. (2008). Aligned cluster analysis for temporal segmentation of human motion. In *Automatic Face Gesture Recognition, 2008 IEEE International Conference on*, pages 1–7.

Zhou, X., Yu, K., Zhang, T., and Huang, T. S. (2010). Image classification using super-vector coding of local image descriptors. In *Proceedings of the 11th European conference on Computer vision: Part V*, ECCV'10, pages 141–154, Berlin, Heidelberg. Springer-Verlag.

Summary: Instance Perception

In this chapter we treated object perception at the *instance* level and contributed to solving two problems:

Problem 1: Training data for environment specific objects is not available. Therefore, it needs to be recorded by the human operator before an agent can be employed and whenever objects in the environment change.

Our contribution: We proposed a method for training environment specific objects on-the-fly. It segments unknown objects from a scene (using 3D information) and trains on them for the later recognition (using high-resolution 2D images).

Problem 2: For Instance Recognition (IR) similar objects need to be discriminated by the algorithms.

Our contribution: We introduced the novel *Radial* orientation scheme for local descriptors. It captures the shape of objects much better than the widely used *dominant local gradient* orientation schemes. Therefore, object signatures are more discriminative.

An agent equipped with our system is able to adapt to changing environments by learning newly introduced objects on-the-fly. Learning and training requires only minimal human supervision. Due to our novel *Radial* orientation scheme the agent is, additionally, able to tell similar objects apart.

Outlook: Category Perception

By going from *instance* perception to *category* perception, we increase the intra-class variance, which requires large datasets for training. Furthermore, while Pose Estimation of instances (PEI) has been thoroughly researched (see [11] for a comprehensive overview), Pose Estimation at the category level (PEC) needs to be addressed, in order to allow for manipulating objects.

5 Object Categorization (OC) and Pose Estimation (PEC)

In this chapter we make the transition from *instance* perception to *category* level perception. While methods for Instance Recognition (IR) aim at being as discriminative as possible in order to distinguish potentially very similar objects, Object Categorization (OC) aims at using such similarities to combine objects into meaningful *categories*.

Two problems arise: First, as objects with varying appearance are being combined into single classes, this leads to complex decision boundaries for machine learning algorithms. We therefore need to employ very adjustable rich predictor models with up to thousands of tunable/learnable parameters (*e.g.*, Deep Convolutional Neural Networks(DCNNs)). Those models tend to over-fit unless provided with large amounts of relevant training data. Generating such data is the focus of Section 5.1. Second, we do not have information about single *instances* when working at the *category* level. As pose estimation usually requires a comparison between stored models and perceived known objects, this presents a challenging problem when dealing with *categories*. A possible solution is introduced in Section 5.2.

5.1 Generating Relevant Training Data

Recording training-images for rich models using the robot's sensors (similar to our IR approach in Chapter 4) or by a human operator obviously is a tedious procedure. Retrieving

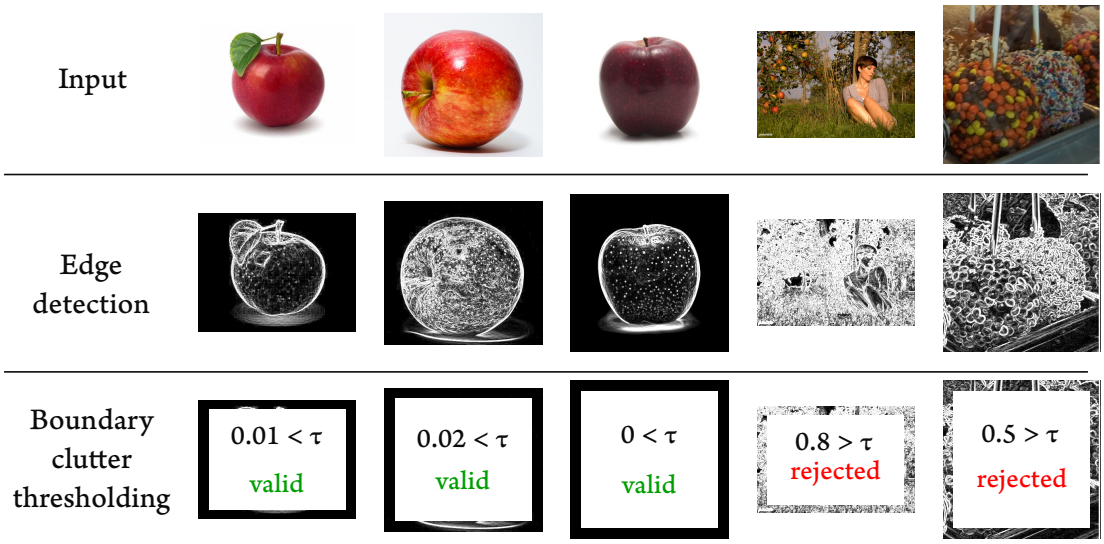


Figure 5.1: Clutter detection step from [3]. Idea: Cluttered images have edges at the image borders. In contrast, uncluttered images have fewer edges at the borders, because they are recorded with plainer backgrounds. Top: Example images for the clutter detector; Middle: Edge detection using the Sobel-Filter [72]; Bottom: By averaging the edge response within a fixed border and thresholding with $\tau = 0.1$, the algorithm filters out cluttered images.

indexed images from word-based databases like Google-Image-Search, while automatic, suffers from the tagging with words in the proximity of the image in a document. This is especially problematic in the case of images for polysemic *categories*, which cannot be retrieved using the ambiguous class name as a search term. For example, the noun washer refers to the metal plate you put under a screw, but it is also used as a synonym for a washing-machine; apple refers to the fruit, but it is also the name of a brand. Therefore, images returned for plain class-name queries do not necessarily show the relevant object or may show the object in a cluttered scene. In fact, Griffin *et al.* [71] measured that only 30 % of the top 100 images returned from Google for the Caltech-256 benchmark generation showed the relevant object in a good way. To alleviate this problem, we introduced the *TransClean* algorithm:

[2] **Schoeler, M.** and Wörgötter, F. and Aein, M. and Kulvicius, T.: "Automated generation of training sets for object recognition in robotic applications", *IEEE/RSJ 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, 2014. See page 39.

[3] **Schoeler, M.** and Wörgötter, F. and Papon, J. and Kulvicius, T.: "Unsupervised generation of context-relevant training-sets for visual object recognition employing multilinguality," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2015. See page 47.

TransClean operates on class+context pairs, which are easily extractable from spoken/written commands or manually set by a human supervisor. The general idea is that ambiguity of classnames (*e.g.*, apple, nut, washer) usually does not exist in other languages. For example, washer translates to the German words "Waschmaschine" (washing-machine) and "Unterlegscheibe" (hardware) - the ambiguity does not exist here. Apple translates to the

German word "Apfel" (only the fruit), brands, names, etc. are usually not translated - again, the same ambiguity does not exist in other languages.

By committing searches in word-based search engines (*e.g.*, Google) for each combination of translated context + translated noun, we can tell relevant from irrelevant translations (*e.g.*, "Waschmaschine" versus "Unterlegscheibe") simply by the number of retrieved documents. Next, the algorithm downloads images for each relevant noun translation (one set per language) using word-based image-databases (*e.g.*, Google-Image-Search). Finally, by using visual image-to-image comparison between images from different languages we can define the relevance of each image by the number of good matches in other sets. While irrelevant images do exist in single sets, only the relevant images find corresponding matches in other sets.

In [2] we integrated this algorithm into a tabletop application using a KUKA LWR robot-arm. For the object segmentation, we used the object extraction and recognition pipeline from Chapter 4. While originally developed for IR, it is also applicable to OC when trained with *category* classes. Only this time we did not use the extracted objects for the training, but the images retrieved by *TransClean*.

In [3] we extended this algorithm and performed an in-depth analysis. One of the extensions, the clutter detection step, proved to be especially valuable in improving the quality of the training set. It filters out images with objects in cluttered scenes, like an apple on a tree in a garden. Since we want our agents to recognize objects in indoor tabletop applications (*e.g.*, kitchen and workshop settings) such images are usually obstructive for the training. The clutter detector applied to example images is shown in Fig. 5.1.

Using our algorithm in [2], we were able to show that our robot automatically disambiguates commands like "crack the nut" and "tighten the nut", retrieves and trains on the relevant training sets, recognizes the objects, and finally, successfully executes the task on the relevant object (in the first case a food-nut, in the second case a hex-nut). The enhanced *TransClean* version [3] is able to increase the fraction of relevant images in automatically generated training sets from under 40 % to over 80 %, which in turn boosted performance of several classifier systems by more than +20 %.

Automated generation of training sets for object recognition in robotic applications

Markus Schoeler, Florentin Wörgötter, Mohamad Javad Aein and Tomas Kulvicius
Bernstein Center for Computational Neuroscience (BCCN)
III. Physikalisches Institut - Biophysik, Georg-August University of Göttingen
Email: (mschoeler, worgott, maein, tkulvic)@gwdg.de

***Abstract*—Object recognition plays an important role in robotics, since objects/tools first have to be identified in the scene before they can be manipulated/used. The performance of object recognition largely depends on the training dataset. Usually such training sets are gathered manually by a human operator, a tedious procedure, which ultimately limits the size of the dataset. One reason for manual selection of samples is that results returned by search engines often contain irrelevant images, mainly due to the problem of homographs (words spelled the same but with different meanings). In this paper we present an automated and unsupervised method, coined Trainingset Cleaning by Translation (*TCT*), for generation of training sets which are able to deal with the problem of homographs. For disambiguation, it uses the context provided by a command like "tighten the nut" together with a combination of public image searches, text searches and translation services. We compare our approach against plain Google image search qualitatively as well as in a classification task and demonstrate that our method indeed leads to a task-relevant training set, which results in an improvement of 24.1% in object recognition for 12 ambiguous classes. In addition, we present an application of our method to a real robot scenario.**

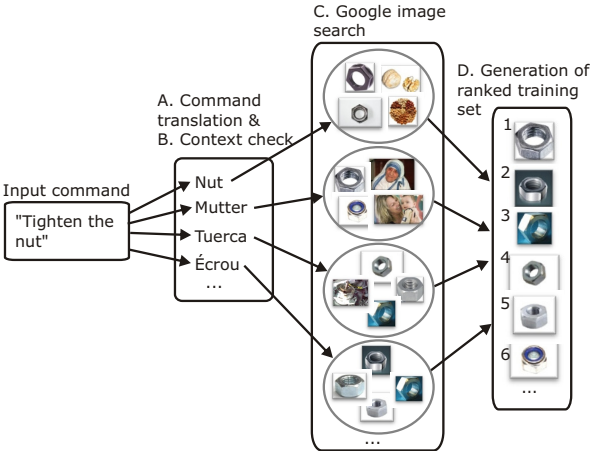


Fig. 1. Flow diagram of the proposed algorithm exemplified on the class "nut" in the context of "tighten". The training set is ranked according to the number of subsets matches occured. Only images which have a match in at least one other subset are further considered.

I. Introduction

In the field of robotics object recognition plays an important role and is crucial for object manipulation tasks, since task specific objects/tools first have to be found and identified correctly before they can be used. To demonstrate, suppose we have a robot-scenario where we tell the robot to "fill the cup with water" as shown in Fig. 6. In order to recognize the bottle and the cup in the scene, the robot has to be trained on these objects beforehand. The training procedure is typically done by off-line training of a classifier with a pre-selected set of classes (images), where images are gathered manually by a human ([1], [2], [3], just to name a few), thus, in a supervised way. Some new approaches make use of Internet searches in order to get information about objects and instructions [4], [5], [6], [7]. Although modern search engines like Google or Yahoo can return a large number of images within milliseconds, not all of the returned images are task/context-relevant, especially due to the problem of homographs (polysemes), i.e., words that are spelled the same but which correspond to different meanings or objects. For example, the word "cup" can correspond to a cup for drinking, the world-cup or bra's cup. "Apple" could mean the fruit, the brand logo or an Apple product. Nut could refer to a hex-nut or the food-nut (see Fig. 2 for an example).

In general, the performance of recognition systems heavily depends on the quality of the training data, thus, only task-relevant images should be collected. This is mostly done by searching for the plain class name or the class name with some context in huge image databases (e.g., Google image search, Bing image search) and by selecting the most-relevant images. As this is especially non-trivial for search terms which are homographs, most recognition methods are trained using manually cleaned or even hand-made training-sets, the creation of which is a time consuming and tedious procedure. Moreover, if a certain task (like "tighten the nut") requires knowledge about an object which is not in the training set, execution is not possible and, even worse, new training images need to be taken or collected and cleaned manually before the robot is able to execute the task.

A lot of research exists on trying to solve this problem of dirty image search results, for example by making use of additional visual cues, e.g., local image patches, edges, texture, color, deformable shapes, just to name a few [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18]. All of these approaches use textual information, too. Either implicitly, by using the first results of text-based image search engines [9], [10], by constructing their own image search engine [12], [13], [18], or explicitly, by making use of image tags and labels as found in photo-sharing websites like Flickr [11], [12], [16]. To our knowledge, all of the above presented approaches achieve an improvement with respect to the quality of the result set. However, none of these methods can automatically cope

[2]

with the problem of homographs (polysemes), which would be required in automated robotic applications like [4], [5], [6].

In this paper, in order to address the problem of homographs, we present a method for automatic (without human supervision) generation of task-relevant training sets for object recognition by using the information contained in a language-based command like "cut the apple" or "fill the cup". We ground our approach based on two facts: 1) homographs rarely occur for one word in multiple languages at the same time and 2) context information (action) provided by the command can be used in order to get rid of ambiguous and non-task relevant translations. In order to create such an automatic system we will employ a combination of publicly available image search engines, text search engines and translation services.

The paper is organized as follows. First, we present our algorithm in detail in Section II. Then, in Section III, we show a qualitative comparison for selected classes (Section III-A) and evaluate the performance of our method quantitatively in an object classification task (Section III-B). Additionally we present an implementation of our method in a real-robot scenario (Section III-C). Finally, we conclude our study in Section IV by discussing our approach and comparing it to other existing methods.

II. Proposed Algorithm

The algorithm consists of four sequential steps: 1) command translation, 2) context check, 3) image download and 4) subset matching (see Fig. 1). In this section we will use the example of nut as it will be important for our robot scenario in Sec. III-C. Nut is a homograph and can mean either a hardware-nut or a food-nut. Generating a training set using a plain Google search for "nut" will not work. In this case even humans cannot infer which object nut refers to. However, the command "tighten the nut" or "crack the nut" provides valuable context information for disambiguation which we want the robot to use.

A. Command translation

The first step of the algorithm is to translate the command to different languages ignoring the articles "the","a","an". For this we translated nouns and verbs separately. Note that in our study we used a fixed command syntax: verb/action + noun/object. A more general command syntax would require the usage of grammar analysis methods (i.e., parsers [19]). In this paper we used four languages: English, German, Spanish, French and Portuguese. Here, Portuguese was only used in the case when translations into the other languages resulted in less than three different terms (e.g., orange is the same word in English, French and German). As an example we will show the generation of the German subset. The first three translations for nut, tighten and crack are shown in Table I. "Mutter" and "Schraubenmutter" correspond to the hardware-nut. "Nuss" corresponds to the food-nut. As one can see the double-meaning of nut is not present in German.

TABLE I. First three English to German translations for nut, tighten and crack returned by www.dict.cc.

nut	**tighten**	**crack**
Nuss	anziehen	zerbrechen
Mutter	verschärfen	knacken
Schraubenmutter	straffen	zersplittern

B. Context check

If the translation service returns more than one translation for the noun this step will perform a context check using Google text search. The idea here is that Google will return significantly less results for a phrase which does not make sense like "Nuss anziehen" (tighten the food-nut), compared to a reasonable phrase like "Mutter anziehen" (tighten the hardware-nut). We forced exact matches using the "as_epq=" search parameter in the Google search. Since the order of the words influence the number of results for exact searches, we searched in both orders (noun verb as well as verb noun) and took the maximum number of results as the score. To retrieve the right noun in the specific context the algorithm uses the noun which gets the highest score with any verb combination. Table II shows how the context relevant German translations for "nut" can be reliably determined. The relevant translations in German, French and Spanish for "crack the nut" are Nuss, Noix, Nuez. The translations for "Tighten the nut" are Mutter, Écrou and Tuerca.

TABLE II. Context check for "tighten the nut" and "crack the nut" using the number of exact matches returned by Google text search. The noun with most matches is chosen (marked bold).

"tighten the nut"		"crack the nut"	
Term	**Matches**	**Term**	**Matches**
Nuss anziehen	445	Nuss zerbrechen	114
Nuss verschärfen	5	**Nuss knacken**	**13500**
Nuss straffen	256	Nuss zersplittern	7
Mutter anziehen	**6500**	Mutter zerbrechen	570
Mutter verschärfen	26	Mutter knacken	476
Mutter straffen	6	Mutter zersplittern	1
Schraubenm. anziehen	218	Schraubenm. zerbrechen	3
Schraubenm. verschärfen	4	Schraubenm. knacken	2
Schraubenm. straffen	4	Schraubenm. zersplittern	1

C. Google image search

This step downloads images for all relevant translations. In the "tighten the nut" context it downloads images for Nut, Mutter, Écrou and Tuerca into 4 separate subsets. In the context of "crack the nut" it downloads images for Nut, Nuss, Noix and Nuez.

D. Generation of ranked training set

Task-relevant images can be found in all subsets, whereas images which correspond to irrelevant context can usually be found only in one set. Nut in the hardware context is a good example as it translates to the German word "Mutter" which is also a homograph meaning the hardware-nut as well as "mother" (see Fig. 2). While mother images are only found in the German and food-nut images only in the English subset, images of hardware-nuts are found in all subsets. For similarity matching we used the procedure proposed by Kulvicius et al. [7]. The pseudo-code in Fig. 3 shows how the score is assigned to each image I_i^k : the number of subsets where a match has been found SM_i^k. Only images which have a match in at least one other subset are considered, i.e., $SM_i^k > 0$. Images are then sorted in descending order by the number of subsets they

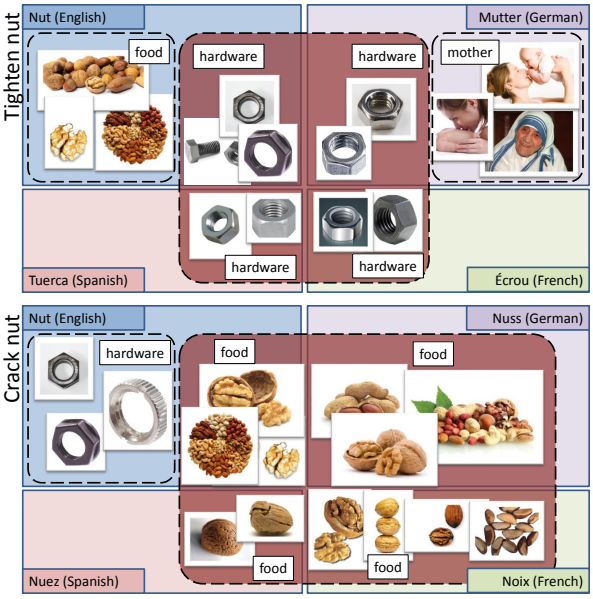


Fig. 2. Example word "nut" which is a homograph in English (food and hardware) and German (mother and hardware). By combining multiple languages and using the context check the proposed algorithm is able to retrieve the task relevant images for "nut" in both tasks ("tighten the nut" and "crack the nut"), the intersection marked with the dark red rectangle)

matched. This assures that most task-relevant images are found at the beginning of the list whereas borderline cases are found at the end.

Get images I_i^k $(k = 1 \dots m, i = 1 \dots n_k)$, where
m is the number of subsearches/languages considered and
n_k is the number of images in subsearch k;
Set similarity threshold θ;
Initialize matches $SM_i^k = 0$.

```
FOR k = 1 to m
    FOR i = 1 to n_k
        FOR l = 1 to m
            IF k! = l
                FOR j = 1 to n_l
                    Compare images I_i^k and I_j^l by
                      calculating similarity s
                    IF s > θ
                        increment(SM_i^k);
                        quit outer loop;
```

Fig. 3. Pseudo-code for the subset matching to determine image relevance.

For similarity calculation the algorithm generates signatures using radially aligned gray-SIFT features as described in previous work [20] (the center is set to the middle of the image). Features are sampled on a dense grid on three scales. A bag-of-visual-words algorithm with 100 visual words is used to generate image signatures. As similarity measure we used the histogram intersection over all visual word bins. The similarity threshold θ was set to 0.7.

III. Results

In order to evaluate the performance of the our algorithm we used 12 homographic classes. All classes, their possible meanings and action contexts are depicted in Table III. All classes have been used in the classification experiment in Sec. III-B. From now on we will denote the class in a specific action context as "class-action" (e.g., nut-crack and nut-tighten). For classification we used the method proposed by [20] which uses a combination of gray-SIFT and CyColor features. Local descriptors are extracted on a dense grid and oriented along the dominant local gradient (the latter using the SURF detector). Three hundred visual words were used for the signature generation. A support-vector-machine with a histogram intersection kernel is used for the machine learning.

TABLE III. The 12 classes used in the evaluation. All classes have multiple meanings (not all are shown). The relevant meaning for the context is marked bold. The last column shows the translations after context check. These are used as the subsets for the image retrieval (see Sec. II-C).

Term-context	Meanings of noun	Translations
apple-cut	**food**, laptop, logo	manzana, pomme, apfel
axe-chop	**hardware**, brand	hacha, hache, axt
bolt-tighten	**hardware**, athlete, movie	tornillo, boulon, bolzen
cup-fill	**drinking**, trophy, bra	taza, tasse
hammer-hit	**hardware**, brand	martillo, marteau
nut-crack	hardware, **food**	nuez,noix, Nuss
nut-tighten	**hardware**, food	tuerca, ecrou, mutter
oil-eat	**food**, mineral-oil	aceite, huile, oel
orange-cut	**food**, color	laranja, naranja
pan-fry	**hardware**, movie, god	sarten, poele, pfanne
peach-eat	**food**, computer character	molocoton, peche, pfirsich
pot-cook	**hardware**, drug	cacerola, casserole, topf
saw-cut	**hardware**, movie	sierra, scie, saege

A. Qualitative comparison

To visualize the qualitative performance of the algorithm Fig. 4 shows the first 10 images retrieved by Google, searching for the plain classname (Google Class only) as well as for the noun together with the action-context (Google Class+Action). Additionally we show the 10 highest ranking images retrieved by our algorithm. The problem of homographs is especially obvious in the case of plain classname searches, since no context is provided which could help to disambiguate. This is why we retrieve the same image sets for "nut" in either context. Consequently bolt and nut in the tighten context show solely irrelevant images except one. Using the action together with the classname does not yield much better results, since images are very affected by image clutter and irrelevant content showing the action instead of the isolated object. In contrast, our algorithm yields a much cleaner image set for all classes.

B. Image classification

Additionally we tested the performance of the algorithm quantitatively in an image classification experiment. We wanted to prove that training a classifier with images obtained by *TCT* results in significantly better classification accuracy as compared to training with uncleaned Google images. For comparison we generated three training sets: One returned from Google search using searches for the plain noun (C200), one with searches for the noun together with the action verb (CA200) and one created by proposed algorithm (TCT). For

[2]

Fig. 4. Images retrieved by Google image search and by our algorithm (*TCT*) for 4 example classes. Only the first 10 highest scoring images are shown.

the latter we used all images which got at least one match in another subset $SM_i^k > 0$. The number ranged from 80 (apple-cut) to 381 images (pan-fry).

The sets C200 and CA200 consist of the 200 highest scoring images. For testing we manually created a disjoint set containing only task-relevant images obtained from Google searches using other languages. Fig. 5 shows the confusion matrices for all three training sets. We can observe that using the context for the Google search yields on average better accuracy than the plain search, since it can disambiguate classes. This however comes at the cost of a high fraction of clutter and irrelevant object in the images returned for "Class+Action" which leads to worse results for "cup-fill", "hammer-hit", "nut-tighten" and "oil-eat". Using our algorithm we are able to increase the recognition accuracy by 24.1 % from 45.9 % to 70 %. Even more important: Using the TCT training set, the classifier can tell "nuts" in the context of "crack" from the ones in the context of "tighten". This is a requirement for

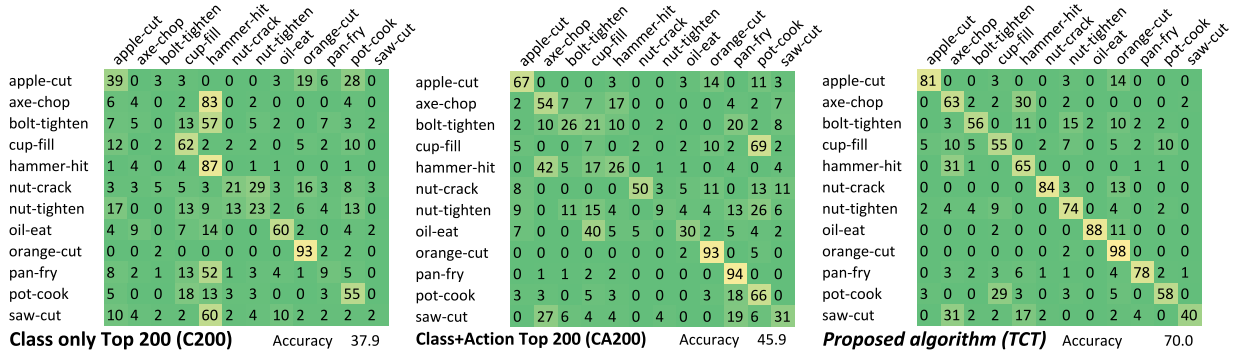


Fig. 5. Confusion Matrix as well as accuracy in percent. Rows correspond to the actual class label and columns to the predicted class labels returned by the classifier. Only if we train the classifier with images returned by *TCT*, we can disambiguate the classes needed in the robot scenario.

the robot scenario in the next section. None of the image sets returned by Google could be used instead. Please note, we also trained a classifier with the 300 highest scoring images (CA300 and C300), but this decreased the classification performance from 45.9 % to 44.7 % for "Class+Action" and from 37.9 % to 33.9 % for "Class only".

C. Robotic application

Last but not least, we applied our method to a robot application where we let a KUKA LWR robot-arm [21] perform three actions (see Fig. 6):

1) "fill the cup" (with water from a bottle)
2) "crack the nut" (with the stone)
3) "tighten the nut"

For each action only one object is task relevant. Since our method was the only one which can discriminate "hardware-nuts" from "food-nuts" we used that one for the training-set generation. In all cases the robot needs to ignore all distractors and choose the right object depending on the action context. Several aspects, like object recognition and robot movement execution, rely on published works and will not be described here in detail. To extract objects from the scene we used the object extraction pipeline of [20] using RGB-D data for segmentation and high resolution images (4928×3264 pixels) for object recognition. We additionally trained a background class which consisted of images of the table, the robot arm as well as the zucchini and the spoon.

For action execution we used the library of manipulation actions from [22], which is based on semantic event chains [23] and modified dynamic movement primitives [24]. Here, specifically, we used pouring, picking-up and putting-down actions. Object positions came directly from the object extraction by averaging all points in the pointcloud belonging to the object. The action "tighten" is a complex action sequence and consists of "pick up", "put on" and "turn". "Put on" and "turn" are difficult actions which require detailed knowledge about the objects and high precision on performing the action (including sensory feedback). As this is not in the focus of this paper, we only required the robot to execute the first step of this action.

In case 1) the robot finds out that cup refers to the coffee-cup and ignores the trophy-cup. Using the context "crack" in case 2) the robot detects the food-nut and ignores the hardware-nut. In case 3) the food-nut is ignored since we generated training images for the context relevant hardware-nut. Note that in our case the commands were typed directly into the computer program with a predefined syntax (action + article + noun). Additionally, we started with the bottle and stone grasped by the robot hand. Consequently, the task for the robot was to find out and recognize which cup and nut the commands refer to and to execute the corresponding action.

In Fig. 6 we show snapshots of the experiment. The robot successfully recognized the cup for filling, the hardware-nut for tightening and the food-nut for cracking. Please refer to the supplementary material for the full video.

IV. Discussion

In this paper we presented a method for automated generation of task-relevant training-sets for object recognition by combining image search engines, text search engines and translation services. The method is useful for obtaining "cleaner results" in image searches. While this is already a valuable property of the algorithm, it is of particular importance in the case of homographs. We showed that the presented approach indeed leads to cleaner search results and better recognition rates as compared to plain Google search. The method was developed with autonomous robotic systems in mind, where a robot has to collect (without human supervision) relevant images from the internet, in order to disambiguate and execute human instructions. In this section we will discuss our approach and how it relates to other existing methods.

In the field of artificial intelligence and computer vision object classification is considered one of the hardest tasks. Due to its importance for many applications, including robotic systems, a lot of effort has been made in order to improve the performance of recognition methods. As shown above, it also highly depends on the quality of the training-set. Generating such training-sets for robotic applications by a human operator is a very time consuming and tedious procedure, which also limits the size of the training set. On the other hand, keeping only the first pages returned by Google [10] limits the size of the training set even more, and worse, will not work

[2]

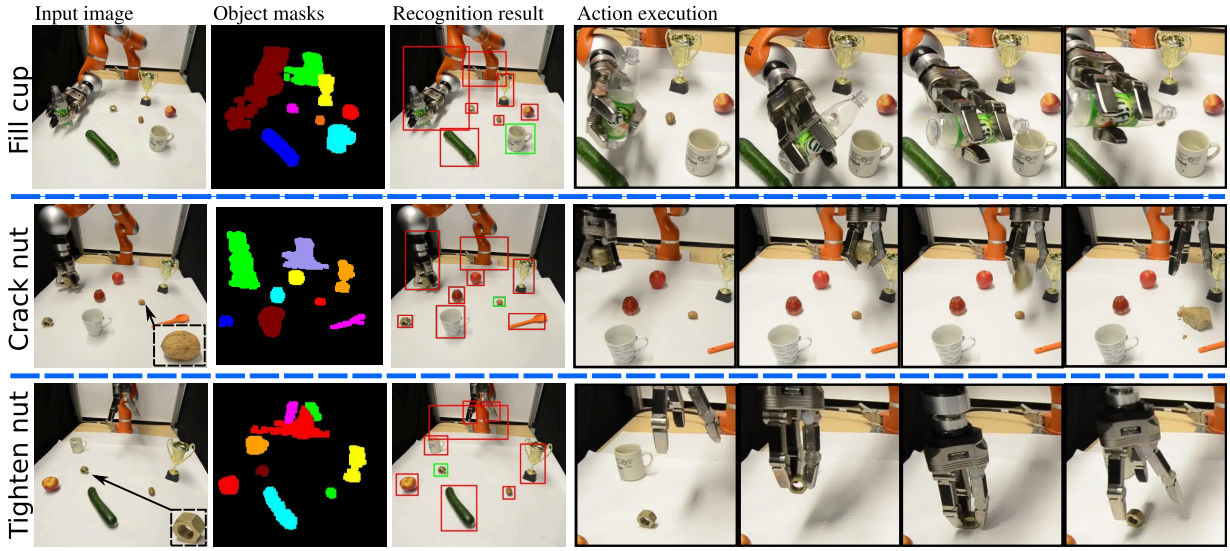


Fig. 6. Three example scenes where the robot had to perform the actions "fill the cup", "crack the nut" and "tighten the nut". The robot starts without knowledge about cups and nuts. In addition to the objects involved in the action we put other items as distractors into the scene. One of them being a different type of cup (the trophy) (see Fig. 4). Even though two items can be referred to by the word nut, only one of them is relevant for the specific action. The robot uses our algorithm to determine the context relevant objects and generates a training set on-the-fly. RGB-D information is used to generate object masks and a high resolution image is used for the classification (see [20] for details). The green box marks the object which gets the highest score from the classifier.

at all for homographic classes (see Fig. 4). The approach presented here provides a solution to solve such problems, based on the additional context information provided by the task (command) and four different subsearches (languages) to automatically retrieve clean training sets. Additionally, adding more languages/subsets (especially with different roots) and several search engines should lead to larger datasets, a more fine-grained relevance score and therefore an even greater improvement in object recognition performance. One could also improve the results by using state-of-the-art image retrieval algorithms like [25] for the image matching.

We have shown that our method performs well as long as the actions allow inference of context, as with fill, crush, crack, pour, cut, screw on, tighten, nail down, and so on. However, performance will drop if actions are used which can be applied to many objects in different contexts, as is generally the case with actions like give, put, move, place, lift and throw. Nevertheless, even humans would experience this problem and would require additional information (if the context is not known beforehand) in cases like "give me the nut".

Using our algorithm for the classifier training we were not only able to boost recognition accuracy by 24.1 % to 70 % (compared to 45.9 % when using images from Google). More importantly using this classifier the robot was able to detect the right objects for "tighten the nut", "crack the nut" and "fill the cup" using the provided context and to successfully execute the command.

Our approach most closely relates to the approaches of Kulvicius et al. [7] and Tamosiunaite et al. [6]. In [7] additional language cues are used in order to perform several sub-searches based on specific context. For example, to generate a task-relevant dataset for the class cup it could use "coffee cup", "tea cup", "full cup", "empty cup", etc. Such context-dependent cues can be obtained from language analysis. However, this requires knowledge about the domain as well as collecting a text-corpora for each specific context. In contrast, in the current approach there is no need for such information, and the context is provided by the action (verb). Similar to our approach, Tamosiunaite et al. [6] make use of language and actions together with Google text search in order to boot-strap in the object domain and to find out which other objects could be used as a replacement. If the command is "cut the cucumber", then the algorithm would return that carrots, potatoes, apples, etc. can be cut, too. Unlike [6], we use the action for a different purpose, i.e., in order to generate the relevant subsets.

As explained above, our approach requires textual (language-based) cues in order to perform image searches. In our study these cues were entered manually in a computer program as a text-command. However, such cues could come from human-robot interaction using natural language communication [26], [27], [28]. Thus robots would obtain language-based commands from humans (e.g., "fill the cup with water"). The other example of language-enabled robots are robots executing instruction sheets based on natural language [4], [5]. The algorithm presented in this paper, as discussed above, is developed having such robotic systems in mind as well.

In summary, we believe that this is a promising approach for automated and unsupervised generation of task-relevant training-sets for object classification/recognition, which has potential for use in many different kinds of robotic applications.

ACKNOWLEDGMENT

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

REFERENCES

[1] M. Muja, R. B. Rusu, G. Bradski, and D. G. Lowe, "REIN-A fast, robust, scalable REcognition INfrastructure," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

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

[3] Y. Sun, L. Bo, and D. Fox, "Attribute based object identification," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[4] M. Tenorth, U. Klank, D. Pangercic, and M. Beetz, "Web-enabled Robots – Robots that Use the Web as an Information Resource," *Rob. & Automat. Magazine*, vol. 18, no. 2, pp. 58–68, 2011.

[5] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth, "Robotic Roommates Making Pancakes," in *IEEE-RAS Int. Conf. on Humanoid Robots*, October, 26–28 2011, pp. 529–536.

[6] M. Tamosiunaite, I. Markelic, T. Kulvicius, and F. Wörgötter, "Generalizing objects by analyzing language," in *IEEE-RAS Int. Conf. Humanoid Robots*, oct. 2011, pp. 557–563.

[7] T. Kulvicius, I. Markelic, M. Tamosiunaite, and F. Wörgötter, "Semantic image search for robotic applications," in *Int. Workshop on Robotics in Alpe-Adria-Danube Region (RAAD2113)*, 2013.

[8] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003, pp. 264–271.

[9] ——, "A visual category filter for google images," in *Europ. Conf. Computer Vision*, May 2004, pp. 242–256.

[10] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman, "Learning object categories from google's image search," in *IEEE Int. Conf. Computer Vision*, vol. 2, oct. 2005, pp. 1816–1823.

[11] M. Guillaumin, J. Verbeek, and C. Schmid, "Multimodal semi-supervised learning for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[12] T. L. Berg and D. A. Forsyth, "Animals on the web," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006, pp. 1463–1470.

[13] F. Schroff, A. Criminisi, and A. Zisserman, "Harvesting image databases from the web," in *Int. Conf. on Computer Vision*, Oct. 2007, pp. 1 –8.

[14] I. Khan, P. M. Roth, and H. Bischof, "Learning object detectors from weakly-labeled internet images," in *35th OAGM/AAPR Workshop*, 2011.

[15] L. Li, G. Wang, and L. Fei-fei, "Optimol: automatic online picture collection via incremental model learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[16] G. Wang, D. Hoiem, and D. Forsyth, "Building text features for object image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2009, pp. 1367 –1374.

[17] Y. Jing and S. Baluja, "Visualrank: Applying pagerank to large-scale image search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1877 –1890, Nov. 2008.

[18] A. D. Holub, P. Moreels, and P. Perona, "Unsupervised clustering for google searches of celebrity images," *IEEE Int. Conf. Automatic Face and Gesture Recognition*, 2008.

[19] D. R. Dowty, L. Karttunen, and A. M. Zwicky, *Natural language parsing: Psychological, computational, and theoretical perspectives*. Cambridge University Press, 2005.

[20] 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.

[21] Kuka Robot Systems. [Online]. Available: http://www.kuka-robotics.com

[22] M. J. Aein, E. E. Aksoy, M. Tamosiunaite, J. Papon, A. Ude, and F. Wörgötter, "Toward a library of manipulation actions based on semantic object-action relations," in *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, 2013, p. in press.

[23] 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.

[24] 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.

[25] S. Paschalakis, K. Iwamoto, N. Sprljan, R. Oami, and M. Bober, "The mpeg-7 video signature tools for content identification," *IEEE Trans. Circuits Syst. Video Technol.*

[26] H. Holzapfel, D. Neubig, and A. Waibel, "A dialogue approach to learning object descriptions and semantic categories," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 1004–1013, 2008.

[27] M. Bollini, S. Tellex, T. Thompson, N. Roy, and D. Rus, "Multi-view object recognition using view-point invariant shape relations and appearance information," in *International Symposium on Experimental Robotics (ISER)*, 2012.

[28] R. Deits, S. Tellex, P. Thaker, D. Simeonov, T. Kollar, and N. Roy, "Clarifying commands with information-theoretic human-robot dialog," *Journal of Human-Robot Interaction*, vol. 2, no. 2, pp. 58–79, 2013.

[2]

[2]

Unsupervised generation of context-relevant training-sets for visual object recognition employing multilinguality

Markus Schoeler Florentin Wörgötter Jeremie Papon
Tomas Kulvicius
III. Physikalisches Institut - Biophysik, Georg-August University of Göttingen
{mschoeler, worgott, jpapon, tkulvic}@gwdg.de

Abstract

Image based object classification requires clean training data sets. Gathering such sets is usually done manually by humans, which is time-consuming and laborious. On the other hand, directly using images from search engines creates very noisy data due to ambiguous noun-focused indexing. However, in daily speech nouns and verbs are always coupled. We use this for the automatic generation of clean data sets by the here-presented TRANSCLEAN *algorithm, which — through the use of multiple languages — also solves the problem of polysemes (a single spelling with multiple meanings). Thus, we use the implicit knowledge contained in verbs,* e.g. *in an imperative such as "hit the nail", implicating a metal nail and not the fingernail. One type of reference application where this method can automatically operate is human-robot collaboration based on discourse. A second is the generation of clean image data sets, where tedious manual cleaning can be replaced by the much simpler manual generation of a single relevant verb-noun tuple. Here we show the impact of our improved training sets for several widely used and state-of-the-art classifiers including Multipath Hierarchical Matching Pursuit. All tested classifiers show a substantial boost of about +20 % in recognition performance.*

1. Introduction

Classifiers are ubiquitous in modern vision applications, spanning various areas including autonomous vehicles, photo classification on image hosting websites or robotic systems in unstructured environments. As such, there has been a significant effort to improve classification pipelines, using more discriminative image features (*e.g.* SIFT [21]) and image signatures (*e.g.*, Bag of Words [8], Fisher Vectors [25]) or by using better machine learning algorithms (*e.g.*, Support-Vector-Machines). Additionally, new approaches like Deep Belief Networks [16] and sparse coding [6] have improved recognition performance in recent years.

Nevertheless, all of these approaches have one thing in common: They are supervised methods which heavily depend on the quality and size of the training set used. Compiling such a dataset typically involves humans in time consuming and tedious procedures [23, 26, 28]. To avoid this, some researchers use the highest ranking images returned by image search engines (*e.g.*, Google) using the class name as a search term [10, 11]. Unfortunately, Griffin *et al.* [12] discovered that only 33 % of the top-100 images from Google are relevant, when using the class name as a search term. This makes it unfeasible for collecting large datasets. One reason for such a low relevance are polysemes (a single spelling with multiple meanings). "Nail", for example could refer to the piece of anatomy or the object one hits with a hammer. Clearly, a command like "hit nail" provides a context to instantly disambiguate the meaning of "nail", and while indexed images for "hit nail" will be more relevant, they also contain a lot of clutter or irrelevant objects. This is a consequence of actions like "tighten bolt" or "hit nail" involving tools in addition to the actually searched for "bolt" and "nail". Figure 8 shows this problem on some examples for plain Google search without context (*GP*) and for Google searches with action context (*GC*).

To address these issues, we have developed TRANSCLEAN (*TC*), which uses verb-noun tuples from sentences. As verb-noun tuples are part of all sentences, it specifically provides a generic solution to the problem of command-disambiguation. This arises in human-robot-interaction applications such as [7, 9, 13, 17] or applications which execute commands from instruction sheets like [3, 19, 29]. On the other hand, TRANSCLEAN can be also be used for improving the relevance of Google Image Search results by manually providing context. While we show results for noun-verb tuples, the algorithm will work with descriptive adjective-noun combinations as well.

The algorithm accomplishes this by creating relevant noun translations for the provided context using the avail-

[3]

able Google services Google Translate, Google Text Search and Google Image Search. We should emphasize at this point that we could use other translation services (*e.g.* dict.leo.org, dict.cc) or search engines which index documents and images by word occurrences (*e.g.* bing.com, yahoo.com).

2. Related work

TRANSCLEAN combines word sense disambiguation with content-based image retrieval. Word sense disambiguation is the task aimed at discovering the meaning of single- and multi-words in texts and mapping occurrences to entries in a reference knowledge database [1, 22]. Image retrieval is the task of retrieving query relevant images from a database. This query can either be a phrase, or (as is the case with content-based image retrieval) another image, which requires computer-vision algorithms to be employed. Multilinguality has been used in various ways, *e.g.* by using parallel text corpora to build multilingual contexts [2, 15] or by exploiting complementary sense evidence from translations in different languages [24]. While these approaches stay in the text domain and require a semantic knowledge base like BabelNet, we exploit multilinguality to eliminate polysemy by applying image retrieval techniques to a superset of images created by multilingual searches. We should note that we are not interested in the semantic meaning of the classes, as is the goal in word sense disambiguation, but rather in an unambiguous training set which can be fed into existing classifier pipelines respecting the context given by *e.g.* a verb. A related approach was proposed by Kulvicius *et al.* [20], who used search cues generated from domain specific text-corporas to create a superset of images which are then merged into one relevant dataset. In contrast to this we do not need to know the context explicitly or to create a text-corpora beforehand. Instead, we deduce the context implicitly from the verb. Other works also make use of visual and textual cues. Either implicitly, by using the first results of text-based image search engines [10, 11], by constructing their own image search engine [5, 27], or explicitly, by making use of image tags and labels as found in photo-sharing websites like Flickr [14]. None of these methods can automatically cope with the problem of polysemes.

This paper is organized as follows: First, in Sec. 3 we present the outline of our algorithm. Section 4 gives an overview of the methods we used for performance evaluation. In Sec. 5 we present quantitative results showing the superiority of our method compared to plain Google search without context (*GP*) as well as Google search together with the context (*GC*). Additionally, we give results for an image classification experiment using several popular and state-of-the-art classifiers including the Multipath Hierarchical Matching Pursuit (M-HMP) proposed by Bo *et al.* [6]. Finally, we summarize and discuss our approach in Sec. 6.

3. Proposed algorithm

The proposed algorithm, depicted in Fig. 1, consists of five sequential parts which we describe in this Section: 3.1 Noun and verb translation, 3.2 Context check, 3.3 Image retrieval, 3.4 Subset matching, and 3.5 Duplicate and clutter removal. The input for the TRANSCLEAN algorithm consists of an object/noun (like apple, orange or saw) and an action context (like cut, fill or prick). For clarity, we adopt the notation *class (context)* to denote a class in a given context, *e.g.*, nail (hit).

3.1. Noun and verb translation

In the first step we translate the noun and verb separately into multiple languages (we use French, Spanish and German unless noted otherwise). Having "cup (fill)" as input, this step would retrieve all translations for cup and fill. We will use https://translate.google.co.uk/ throughout this paper. For washer (clean) the translations in French are: *washer: rondella, machine à laver; clean: nettoyer, éplucher, faire nettoyage, ratiboiser, ravaler, vider*.

3.2. Context check

This step determines the most relevant noun-translation for each language using Google Text Search https://www.google.co.uk/. For each verb-noun combination we perform two exact searches: "noun verb" and "verb noun". For both searches we parse the number of results and take the maximum as the relevance score for that combination. The noun which gets the most matches combined with any verb is then selected as the relevant translation. We use the Google search parameter *lr* to only retrieve results from documents in a specific language.

In the washer (clean) example shown in Fig. 2 the translations which get selected are *machine à laver* in French, *Waschmaschine* in German and *lavadora* in Spanish.

3.3. Image retrieval

This step downloads the first 300 images for the translations which passed the context check (one per language) as well as for the English search. Again, we set the parameter *lr* to the respective language. For instance, to download images for "Waschmaschine" we use https://www.google.co.uk/search?q=Waschmaschine&tbm=isch&lr=lang_de.

3.4. Subset matching

In this step we are going to merge the different language subsets into one relevant dataset. We do this by pair-wise image comparison across the different subsets. To calculate similarity between two images we generate a histogram

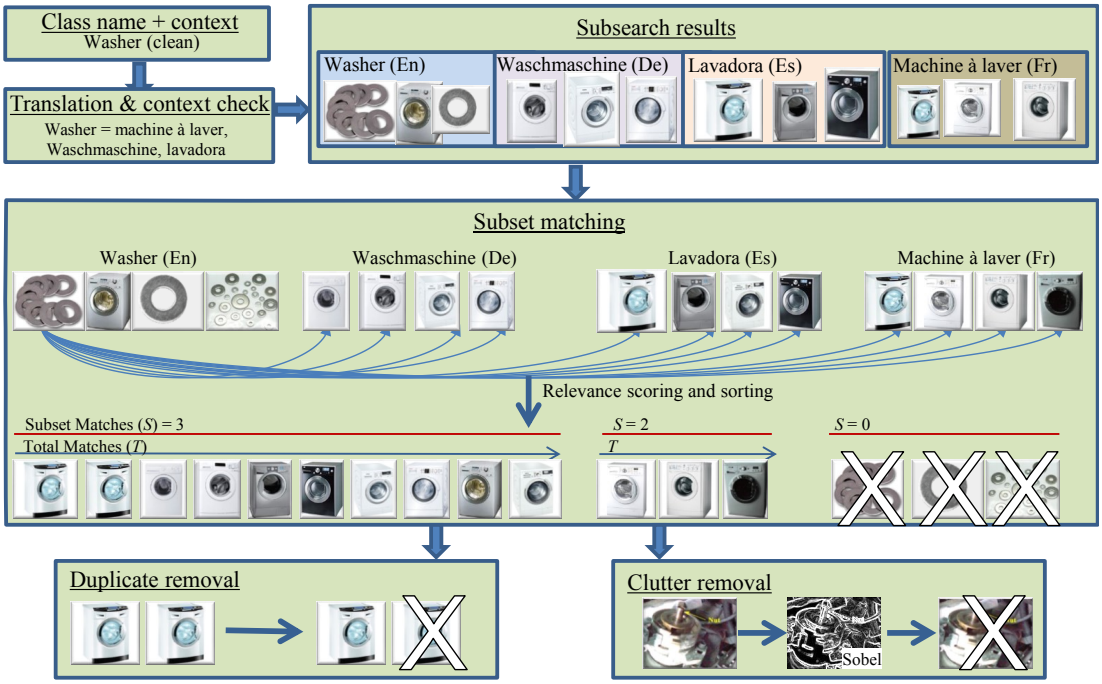


Figure 1. Flow diagram of the TRANSCLEAN algorithm exemplified on the class "washer" in the context of "clean". Subset Matches ($\mathcal{S}$) counts the total number of subsets in which a match has been found. Total Matches ($\mathcal{T}$) counts the total number of matches. $\mathcal{S}$ is our first order and $\mathcal{T}$ our second order relevance sorting criteria. Only images with $\mathcal{S} > 0$ are considered further. We used ISO639-1 language codes.

French

	nettoyer	éplucher	faire nettoyage	ratiboiser	ravaler	vider
rondella	0	0	0	0	0	0
machine à laver	30400	5	0	0	1	313

German

	reinigen	säubern	waschen	sauber machen	abwischen
Waschmaschine	39400	1430	126000	827	194
Scheibe	5390	2380	2530	7300	1200
Geschirrspülmaschine	3940	646	2270	7	8

Spanish

	limpiar	asear	mondar	hacer una limpieza
lavadora	2140	2	0	0
arandela	185	5	0	0

Figure 2. Context check for washer (clean). Rows: Translations for the noun; columns: Translations for the verb. Each cell shows the number of exact Google search results (relevance score) for the noun-verb combination. The noun with the highest response gets selected (marked with red).

for each image using SIFT features [21]. Since we reduce color images to gray-scale for the SIFT features, we are able to compare color to gray-scale images as well. The features are sampled on a 14×14 grid on the full image using four scales. Bag-of-words [8] with $\mathcal{N} = 100$ clusters is used to generate normalized image signatures. The histogram intersection (min-kernel) over all bins is used as a match score. We require the score $\mathcal{M}$ to exceed a empirically determined threshold $\theta_{\text{th}} = 70\,\%$ to count as a match:

$$\mathcal{M}(x, y) = \sum_{i}^{\mathcal{N}} \min(x_i, y_i) \geq \theta_{\text{th}}. \quad (1)$$

A procedure similar to that of Kulvicius *et al.* [20] is used to generate an overall image relevance score, improving upon their work by using an additional relevance score. Fig. 1 shows how each image is scored: The score consists of 1) the number of other subsets where at least one match has been found $\mathcal{S}$ and 2) the total number of matches $\mathcal{T}$. Images without subset matches $\mathcal{S} = 0$ are pruned. Relevance of images is determined first by $\mathcal{S}$ and second by $\mathcal{T}$.

Given a fixed number of SIFT features per image and for the vocabulary generation, the complexity of the histogram generation algorithm is linear in the number of images $\mathcal{O}(kn)$ with k being the number of languages considered and n denoting the number of images per set. The complexity of the matching is $\mathcal{O}(k^2n^2)$. While the matching scales much worse, it is the fastest part of the algorithm when using four languages and 300 images per language, as it only consists of simple min operations. For $n = 300$ and $k = 4$ the running time of the image-to-image matching without parallelization is about 13 seconds on a 3.2 GHz processor.

[3]

Object	Given Context	Meanings
apple	cut	**food**, brand
axe	chop	**tool**, brand
bolt	tighten	**hardware**, athlete, movie
cup	fill	**drinking**, trophy, bra
fork	prick	**cutlery**, bike-part
glass	fill	**drinking**, material
hammer	hit	**tool**, brand
nail	hit	**hardware**, finger
nut	tighten	**hardware**, food
oil	eat	**food**, mineral-oil
orange	cut	**food**, color
pan	fry	**kitchenware**, movie, god
peach	fry	**food**, computer character
pot	cook	**kitchenware**, drug
saw	cut	**tool**, movie

Table 1. All classes together with their given context used for the experiments. Possible meanings as well as the relevant meaning (marked in bold) are shown.

3.5. Duplicate and clutter removal

Finally, in order to clean the result set of duplicate images and images with cluttered scenes we perform a duplicate and clutter search. To do this, we scale all images to exactly 150×150 pixels ignoring the aspect ratio and generate gradient magnitude images g_i using the sobel filter (the values g_i are in the range of 0 to 1). The similarity between image i and image j is calculated by L1-normalizing the gradient images g_i and g_j and calculating the histogram intersection. The duplicate threshold was empirically determined and set to 0.85 throughout all experiments. When a duplicate is found the image with lower relevance score is deleted. Additionally, we remove cluttered images by calculating the mean gradient magnitude within a five pixel image border of g_i. This value ranges between 0 (no clutter) and 1 (heavy clutter). Using a clutter threshold of 0.1 effectively removed all images which were recorded in cluttered scenes and therefore considered bad for the training (*e.g.*, an apple on a tree in a garden).

4. Evaluation methods

For the evaluation of our algorithm we benchmarked on the classes shown in Table 1. In all cases the noun itself is ambiguous and could refer to multiple meanings. In addition, we also provide context, in the form of verbs, which can be used for disambiguation. We used four languages: English, German, Spanish and French. Additionally, we used Portuguese for orange as the word is the same in German, French and English. We evaluate the proposed algorithm (*TC*) against plain Google search (*GP*) as well as Google search including the context (*GC*). For *GP* we conduct searches using the noun without the provided context to retrieve images. For *GC* we also provide the context together with the class label for the search. For example, with "pan" in the context of "fry", we retrieve images for *pan* (*GP*) as well as *fry pan* (*GC*). Image searches for *GC* are always conducted without quotes.

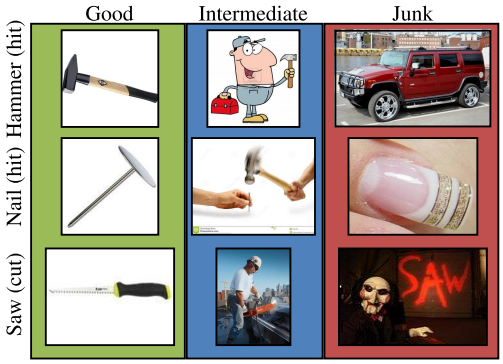


Figure 3. Three images showing the quality grading as introduced by [10] for three classes.

4.1. Quality of retrieved training-sets

Both our algorithm and Google provide results in an ordered list. We therefore first investigate how image-quality changes depending on list length. This is important as larger training image datasets should generally improve classifier performance. However, using more images may negatively affect the overall relevance of the set if more non-relevant images are added as the set expands, unnecessarily increasing intra-class variance. To quantify this effect we let a human grade each image retrieved by *GP*, *GC* and *TC*. We followed the categorization scheme introduced in [10]:

Good image Image containing the relevant object without major occlusions although there may be a variety of viewpoints, scalings and orientations.

Intermediate image Shows the relevant object, but may contain extensive occlusion, substantial image noise or the object is insignificant in the image.

Junk image Not relevant.

Stereotypic images for the three quality levels can be seen in Fig. 3. We use precision as well as quality distribution to assess the quality of the retrieved image sets. Precision is calculated counting only *Good* images as positives. Here we are especially interested in how precision changes depending on the number of images considered. This resembles the measure introduced by [12] in the Caltech-256 benchmark generation. The quality distribution, on the other hand, measures the ratio of the three quality categories, allowing deeper analysis of the properties of the methods.

4.2. Image Classification

To assess the performance dependency of classifiers on the training set we used three state-of-the-art object classification pipelines: Multipath Hierarchical Matching Pursuit (M-HMP)[6], the SIFT and Bag-of-Words based approach of Iravani *et al*. [18] combined with a Support-Vector-Machine (SIFT_SVM) and AdaBoost.MH (SIFT_BOOST). We decided on SIFT and Bag-of-Words as it has been and still is a very popular classification pipeline in the scientific community. Similar to the work of [18], SIFT features were extracted on a dense grid and 300 cluster centers were used to generate the signatures. We tested different kernels (χ^2, RBF and Histogram-Intersection) for the C-Support-Vector-Machine and found similar performance changes in all of them when trained with the 5 sets. For AdaBoost we used the multiclass capable Adaboost.MH from the MultiBoost library [4]. M-HMP achieved state-of-the-art results in many standard benchmarks by combining a collection of hierarchical sparse features to capture discriminative structures in the images[1]. For testing we use a manually cleaned image set containing only *Good* quality images which are disjoint from the sets used in the training.

5. Evaluation

5.1. Context check

We conducted the context check with the contexts given in Table 1. Additionally, we investigated more closely how stable the translations of "nail" and "nut" are against changing contexts.

For "nail" we chose four contexts: hit, pin, paint and cut. For the first two we expect our algorithm to retrieve hardware-nail images and for the last two fingernail images. Figure 4 shows the context check scores we retrieved from Google Text Search using the French translations provided by the Google translation service. All actions can be used to correctly disambiguate the meaning of "nail".

"Nut" shows an interesting case when using the context "eat" (see Fig. 5). We expected the algorithm to select only the food-nuts. Unluckily one translation in German for "nut" is "Mutter", which means hardware nut, but also mother. Since mothers do also eat (and there is a lot written about this in the documents indexed by Google) the context check found this to be the relevant translation for the German subset. The same polyseme, however, does not exist in French, Spanish or Portuguese. Therefore the overall performance of the algorithm is stable as long as one or more unaffected languages are used.

Table 2 shows all the translations of the 15 classes used in the following sections.

[1]We used the publicly available code http://homes.cs.washington.edu/~lfb/software/hmp/index.htm distributed by the authors.

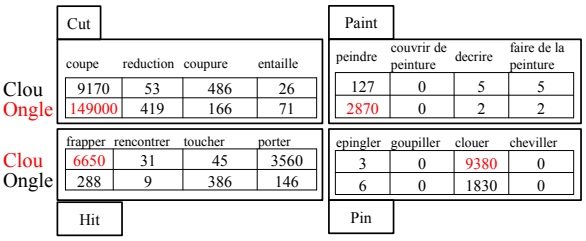


Figure 4. Context check for the class "nail" in different contexts. Shown are the number of matches from Google Text Search for the French set. Top rows correspond to responses for the hardware-nail (clou) and bottom rows refer to the fingernail (ongle). Using the noun which gets the most matches (highlighted in red) the algorithm manages to retrieve the fingernail in the context of "cut" and "paint", whereas the contexts "hit" and "pin" lead to the hardware nail being selected.

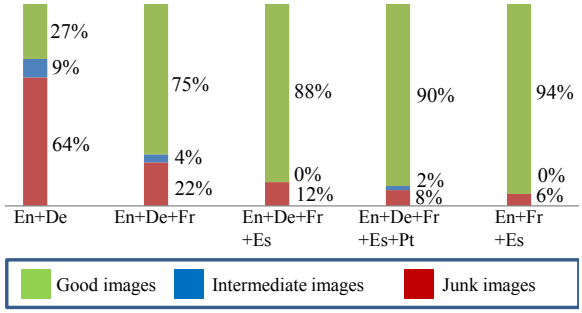


Figure 5. Quality distribution using the first 80 images of the *TC* generated sets for "nut (eat)" considering different languages depicted by their ISO639-1 language codes. Although the context check decided for the wrong translation in German, the algorithm only fails if no additional languages are used (En+De).

Class	Context	Translations
apple	cut	manzana, pomme, apfel
axe	chop	hacha, hache, axt
bolt	tighten	tornillo, boulon, schraube
cup	fill	taza, tasse
fork	prick	tenedor, fourchette, gabel
glass	fill	vaso, verre, glas
hammer	hit	martillo, marteau
nail	hit	clavo, clou, nagel
nut	tighten	tuerca, ecrou, mutter
oil	eat	aceite, huile, oel
orange	cut	laranja, naranja
pan	fry	sarten, poele, pfanne
peach	fry	molocoton, peche, pfirsich
pot	cook	cacerola, casserole, topf
saw	cut	sierra, scie, saege

Table 2. The final translations retrieved by the context check.

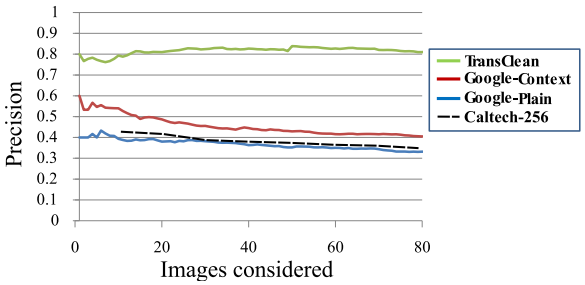


Figure 6. Average precision of the retrieved images (counting only *Good*) depending on the number of images considered. *TC* significantly outperforms the other methods. We included the Caltech-256 dataset curve [12] (dashed line, which closely resembles our results for Google-Plain) showing that our selected classes are representative of typical data.

5.2. Quality measure

Figure 6 quantifies how precision of the sets changes depending on the number of images considered. On average, our algorithm improves Google results significantly, doubling the precision to about 80 % (in contrast to 41 %) for the first 80 images. While the average precision of our algorithm stays constant, *GP* drops by 10 % and *GC* even by 15 % when increasing the list length from 15 to 80. Remarkably, the averaged curve for Google-Plain resembles the curve found when creating the Caltech-256 dataset [12]. This confirms our notion that the 15 classes selected as a demonstration are representative of typical performance of unfiltered Google Image Searches.

Interestingly, for some classes like "hammer", plain Google Search without context outperforms Google Search with context. The reason for this is that using the context in a Google Search often retrieves images which are more related to the action than to the object itself. That is, while these images show the relevant object they often have a lot of clutter or the object is not visible at all (they will consequently be rated as *Intermediate* or *Junk*) as shown in Fig. 7. Searches for nouns without the context verb show a small fraction of *Intermediate* results, since images show either the correct object or not. Examples of results from the three methods can be seen in Fig. 8.

Our method retrieves on average 80 % *Good* images compared to 41 % returned by Google. Even the worst class "oil" shows 50 % *Good* images, which is 9 % better than the average quality of the Google searches. We shall show in the next section how much classifier performance can be boosted using training images returned by our TRANSCLEAN algorithm.

5.3. Image classification

As a demonstration of the effectiveness of TRANSCLEAN, we conducted an experiment to show

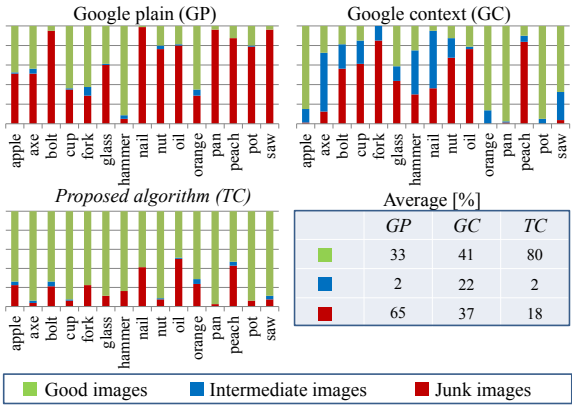


Figure 7. Quality distribution for all classes in the contexts given in Table 1. We evaluated the 80 highest ranking images using the measures introduced by [10] (see Sec. 4.1). Full bars correspond to a 100 %.

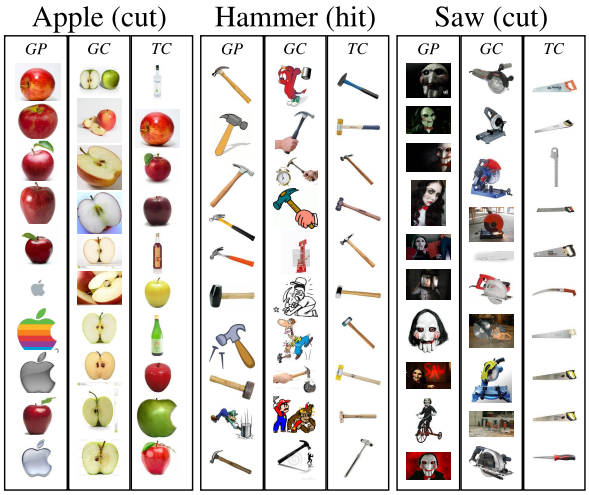


Figure 8. The top 10 images retrieved by Google Image Search (*GP*), Google Image Search with context (*GC*) and proposed TRANSCLEAN algorithm (*TC*).

the effect of different training sets on classifiers. We demonstrate that classifiers (see Sec. 4.2) are heavily dependent on the quality of the training sets and that our approach boosts them, resulting in better performance. To show this, we created five training sets: One training set generated by our algorithm (*TC*) and four sets consisting of plain Google search as well as context Google search results using the first 30 or 300 images (*GP30, GP300, GC30, GC300*).

As can be deduced from Figure 9, the state-of-the-art classifier M-HMP is far superior to the classifiers based on SIFT and Bag-of-Words showing about 20 % better accuracy for the same training set. The M-HMP classifier achieved 61 % accuracy when using Google search images

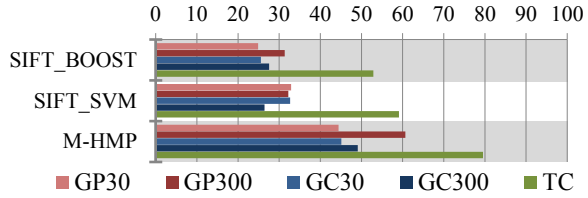


Figure 9. Performance impact of the training-sets on different classifier pipelines. All classifiers improve by roughly 20 % in accuracy using the *TC* set.

for only the noun (*GP*), 49 % when committing searches together with the verb (*GC*) and 80 % using images retrieved by our proposed method. Both Support-Vector-Machines and AdaBoost show poor performance across the Google search datasets. Nevertheless, using the TRANSCLEAN generated training sets improves performance across all classifiers by 20 %.

Figure 10 shows the per class recall and mean classification accuracy for the M-HMP. Remarkably, using the context for the search in general yields worse results than using the plain word for the search. The reason for this is the high percentage of *Intermediate* quality images in the *GC* sets (see Fig. 7). These images not only show the desired object in a bad way but typically also show other objects related to the action. This is far more destructive than *Junk* images, as the decision boundaries for one class may spread into the area of another class, *e.g.*, a hammer may be classified as a nail since it was shown in many nail (hit) images. Our method, in contrast, provides a high fraction of *Good* images and a low fraction of *Intermediate* images. This is clearly visible in the 19 % accuracy boost compared to the second best performance achieved (using *GP300*).

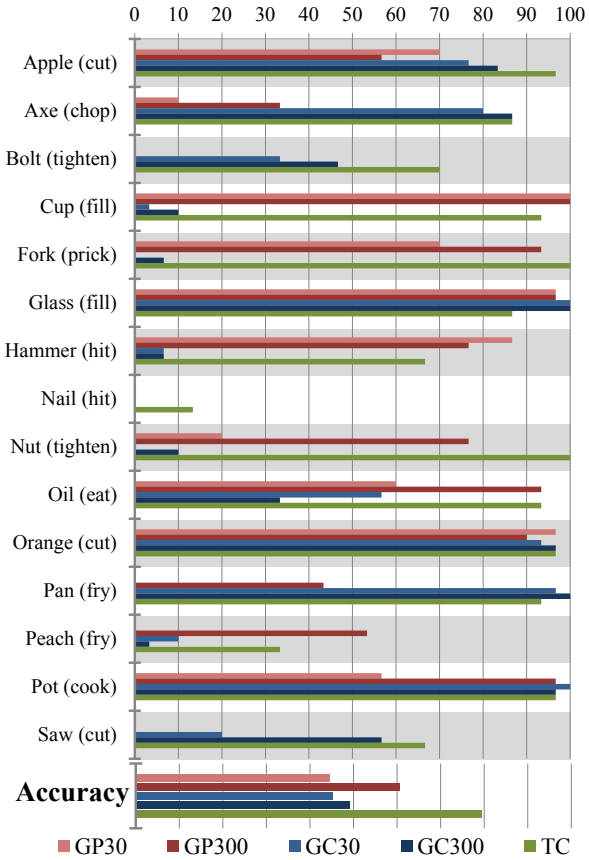


Figure 10. Per class recall and accuracy in percent of the M-HMP classifier using different training sets.

6. Conclusion

In this paper we presented an approach for unsupervised generation of training-sets for image classification algorithms. We proved using several experiments that the method outputs not only high quality sets of images when providing the class name and an action context but can also cope with polysemous classes. We also showed that while Google Image Search disambiguates quite well when provided the action context (*GC*), it suffers from a high fraction of images showing the object in a cluttered scene (*Intermediate* images). *GP* on the other hand shows a high fraction of images unrelated to the desired meaning. This is in agreement with the results reported by Griffin *et al.* [12]. Remarkably, *Intermediate* images are often far more destructive for a classifier's performance than *Junk* images, as wrong objects shown consistently with the right object may be learned instead. Our approach, however, yields few *Intermediate* images (2 %) and a high fraction of *Good* images (80 %).

For this paper we provided manually entered search terms (which is one of the reference applications), but this algorithm can be adapted to extract needed noun+verb (or noun+adjective) tuples from autonomous and/or interactive robotic systems, since a context is usually available as part of a spoken or written command. Since we combine multiple languages the algorithm is robust against incorrect translations or context check failure in single languages as shown in Fig. 5. Using very class-specific actions will result in better context check and therefore algorithm performance. Conversely, one potential pitfall is that very general actions like "put", "take" and "place" may not show an improvement due to their context being applicable to many objects. This is a general problem however, as even humans are unable to disambiguate a class based on very general context.

Due to the importance of image classifiers for many modern applications and their need for large, high quality training sets we hope that this method will be useful for researchers in various fields.

Acknowledgment

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

References

[1] E. Agirre, O. López de Lacalle, and A. Soroa. Random walks for knowledge-based word sense disambiguation. *Comput. Linguist.*, 40(1):57–84, Mar. 2014. 2

[2] C. Banea and R. Mihalcea. Word sense disambiguation with multilingual features. In *Int. Conf. on Computational Semantics*, pages 25–34, 2011. 2

[3] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth. Robotic Roommates Making Pancakes. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 529–536, 2011. 1

[4] D. Benbouzid, R. Busa-Fekete, N. Casagrande, F.-D. Collin, and B. Kégl. Multiboost: A multi-purpose boosting package. *Journal of Machine Learning Research*, 13:549–553, 2012. 5

[5] T. L. Berg and D. A. Forsyth. Animals on the web. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1463–1470, 2006. 2

[6] L. Bo, X. Ren, and D. Fox. Multipath sparse coding using hierarchical matching pursuit. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 660–667, 2013. 1, 2, 5

[7] M. Bollini, S. Tellex, T. Thompson, N. Roy, and D. Rus. Interpreting and executing recipes with a cooking robot. In *Experimental Robotics*, volume 88 of *Springer Tracts in Advanced Robotics*, pages 481–495. Springer Int. Publishing, 2013. 1

[8] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004. 1, 3

[9] R. Deits, S. Tellex, P. Thaker, D. Simeonov, T. Kollar, and N. Roy. Clarifying commands with information-theoretic human-robot dialog. *Journal of Human-Robot Interaction*, 2(2):58–79, 2013. 1

[10] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In *IEEE Int. Conf. Computer Vision*, volume 2, pages 1816–1823, 2005. 1, 2, 4, 6

[11] R. Fergus, P. Perona, and A. Zisserman. A visual category filter for google images. In *Europ. Conf. Computer Vision*, pages 242–256, 2004. 1, 2

[12] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007. 1, 4, 6, 7

[13] S. Guadarrama, L. Riano, D. Golland, D. Gouhring, Y. Jia, D. Klein, P. Abbeel, and T. Darrell. Grounding spatial relations for human-robot interaction. In *IEEE/RSJ Int. Conf. on Intelligent Robots and System (IROS)*, pages 1640–1647, 2013. 1

[14] M. Guillaumin, J. Verbeek, and C. Schmid. Multimodal semi-supervised learning for image classification. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010. 2

[15] W. Guo and M. Diab. Combining orthogonal monolingual and multilingual sources of evidence for all words wsd. Annual Meeting of the Association for Computational Linguistics, pages 1542–1551. Association for Computational Linguistics, 2010. 2

[16] G. E. Hinton and S. Osindero. A fast learning algorithm for deep belief nets. *Neural Computation*, 18, 2006. 1

[17] H. Holzapfel, D. Neubig, and A. Waibel. A dialogue approach to learning object descriptions and semantic categories. *Robotics and Autonomous Systems*, 56(11):1004–1013, 2008. 1

[18] P. Iravani, P. Hall, D. Beale, C. Charron, and Y. Hicks. Visual object classification by robots, using on-line, self-supervised learning. In *IEEE Int. Conf. on Computer Vision Workshops (ICCV Workshops)*, pages 1092–1099, 2011. 5

[19] P. Kaiser, M. Lewis, R. P. A. Petrick, T. Asfour, and M. Steedman. Extracting common sense knowledge from text for robot planning. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3749 – 3756, 2014. 1

[20] T. Kulvicius, I. Markelic, M. Tamosiunaite, and F. Wörgötter. Semantic image search for robotic applications. In *Int. Workshop on Robotics in Alpe-Adria-Danube Region (RAAD)*, 2013. 2, 3

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60:91–110, Nov. 2004. 1, 3

[22] T. Miller, C. Biemann, T. Zesch, and I. Gurevych. Using distributional similarity for lexical expansion in knowledge-based word sense disambiguation. In *Proc. of COLING*, 2012. 2

[23] M. Muja, R. B. Rusu, G. Bradski, and D. G. Lowe. REIN-A fast, robust, scalable REcognition INfrastructure. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011. 1

[24] R. Navigli and S. P. Ponzetto. Joining forces pays off: Multilingual joint word sense disambiguation. In *Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1399–1410, 2012. 2

[25] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007. 1

[26] 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. 1

[27] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 1–8, 2007. 2

[28] Y. Sun, L. Bo, and D. Fox. Attribute based object identification. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013. 1

[29] M. Tenorth, U. Klank, D. Pangercic, and M. Beetz. Web-enabled robots. *Robotics Automation Magazine, IEEE*, 18(2):58–68, June 2011. 1

5.2 Concurrent Categorization and Pose Estimation

The *TransClean* algorithm is powerful in generating large amounts of relevant training data for OC pipelines. Using this algorithm, a robot can automatically retrieve new training images and train its classifier whenever a task with an unknown *category* requires it. Unluckily, it is insufficient for complicated robotic manipulations, which require the use of an object in a defined way. The paper in the previous section [2], for instance, showed the robot execution of the task "fill the cup". It "assumed" the cup in an upright position, which may not be necessarily the case in some scenarios. For such scenarios one needs to estimate pose for encountered objects before actions can be executed. This is commonly addressed by aligning the stored model to a new observation (of the same object). While possible in the context of IR, this is not applicable to OC, because we deal with *category* level classes and, therefore, do not have models for particular objects. This motivated us to design our combined Categorization-Pose Estimation framework (p. 57) using a DCNN architecture:

[4] Papon, J. and **Schoeler, M.**: "Semantic Pose using Deep Networks Trained on Synthetic RGB-D," *IEEE International Conference on Computer Vision (ICCV)*, 2015 (in press). See page 57.

Here we treated pose estimation and classification in unison using a common large network. We solved the need for large amounts of annotated training data for such a network by automatically assembling synthetic scenes with thousands of different models from various *categories*. To close the gap between training on synthetic scenes and testing on real scenes, we simulated the model and geometry of real RGB-D sensors and did subsequent transfer learning[1] on the 795 training-images from the *NYU Depth V2*[2] dataset. As a consequence, we demonstrated that networks trained on synthetic RGB-D scenes can be easily adapted to work on the most challenging real scenes available. This forged a system which not only detects objects, predicts the *category* and the pose, but does so on heavily cluttered scenes with a high degree of confidence.

[1] For transfer learning, we do not initialize weights randomly. Instead, learning on the target domain (in our case NYU) starts with weights trained on the first domain (in our case Synthetic scenes). This way the network converges faster and needs less training data from the target domain.

[2] http://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html

Semantic Pose using Deep Networks Trained on Synthetic RGB-D

Jeremie Papon and Markus Schoeler
Bernstein Center for Computational Neuroscience (BCCN)
III. Physikalisches Institut - Biophysik, Georg-August University of Göttingen
jpapon@gmail.com mschoeler@gwdg.de

Abstract

In this work we address the problem of indoor scene understanding from RGB-D images. Specifically, we propose to find instances of common furniture classes, their spatial extent, and their pose with respect to generalized class models. To accomplish this, we use a deep, wide, multi-output convolutional neural network (CNN) that predicts class, pose, and location of possible objects simultaneously. To overcome the lack of large annotated RGB-D training sets (especially those with pose), we use an on-the-fly rendering pipeline that generates realistic cluttered room scenes in parallel to training. We then perform transfer learning on the relatively small amount of publicly available annotated RGB-D data, and find that our model is able to successfully annotate even highly challenging real scenes. Importantly, our trained network is able to understand noisy and sparse observations of highly cluttered scenes with a remarkable degree of accuracy, inferring class and pose from a very limited set of cues. Additionally, our neural network is only moderately deep and computes class, pose and position in tandem, so the overall run-time is significantly faster than existing methods, estimating all output parameters simultaneously in parallel on a GPU in seconds.

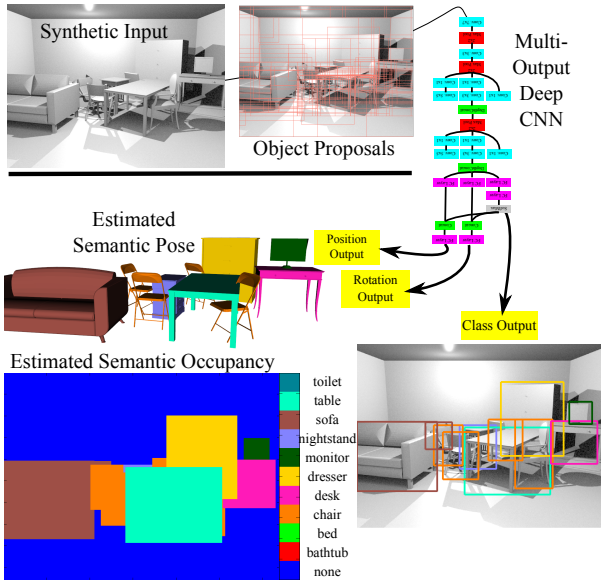


Figure 1. Overview of our approach. Normals for a scene are efficiently calculated using [7], proposals are generated using [8], and then fed through our synthetically trained CNN. Outputs are then consolidated using non-maximum suppression, leading to a scene class & pose heat map and a scene rendered with generalized models.

1. Introduction

In order for autonomous systems to move out of the controlled confines of labs, they must acquire the ability to understand the cluttered indoor environments they will inevitably encounter. While many researchers have addressed the problems of pose estimation, object detection, semantic segmentation, and object classification separately, comprehensive understanding of scenes remains an elusive goal. To this end, in this work we propose an architecture which is able to perform all of the above tasks in concert using a single artificial neural network.

Classification in cluttered indoor scenes can be extremely challenging, especially when trying to classify instances of objects which have never been observed before. Considering only 2D color information only compounds this problem, as clutter can easily cause vast changes in the visible signature of otherwise distinguishable items. 3D geometric features, on the other hand, tend to be less susceptible to clutter and have (especially for furniture) geometric features which generalize well across the class. As such, in this work we use 3D geometric features in addition to standard RGB channels.

Pose estimation in-the-wild is another difficult problem, as it requires estimating pose for object instances which have never been observed before. For example, consider the task of helping a human to sit down in a chair - to be of any help, one must be able to determine pose of the back-

[4]

Figure 2. Example of estimated pose output (overlaid as a generic orange model) for chairs from the test set. Pose here is shown using a generic chair model. None of these test models were observed in training.

rest, the seat area, and the supporting legs - even on types of chairs that one has never seen before. In this work we will show that just such a task is possible, to a surprising degree of accuracy, using a wide, deep, multi-stage CNN trained on synthetic models. In fact, it is possible to do so even with wholly unobserved types of chairs - for example, in Fig. 2, none of the chair models were seen in training. Moreover, we shall demonstrate that it is possible to estimate such poses even in complex cluttered scenes containing many classes of furniture (*e.g.* see Fig. 3).

Our approach, outlined in Fig. 1, uses a relatively complex CNN architecture to solve our three sub-tasks; class-, pose-, and position-estimation of objects, concurrently. One unusual aspect of our network is that it recombines class output back into the network layers which calculate pose and position, allowing the network to accurately determine pose for multiple classes within a single architecture. Furthermore, we are able to train this large network by using synthetic rendered RGB-D scenes consisting of randomly placed instances from a dataset of thousands of 3D object models. Our training scenes are generated on the fly on the CPU and a secondary GPU as we train on the primary GPU, allowing us to have a training set of virtually unlimited size at a completely hidden computational cost. Finally, we use a small number of transfer learning iterations using a small set of real annotated images to adapt our network to the modality of real indoor RGB-D scenes.

To demonstrate the effectiveness of our approach, we perform a variety of experiments on both synthetic and real scenes. Our pose estimation and classification results outperform existing methods on a difficult real dataset. We also present qualitative and quantitative results on both real and synthetic data which demonstrate the capability of our system to distill semantic understanding of scenes. Moreover, we do these tasks jointly in a single forward pass through our network, allowing us to produce results significantly faster than existing methods.

1.1. Related Work

As we propose to solve multiple problems in tandem in this work, there is a substantial body of work which could be considered related. We will restrict ourselves to those recent works which deal exclusively with RGB-D data and/or use CNNs to accomplish one or more of our sub-tasks.

As a first step in a pipeline to parse full scenes, the image is typically broken down into small "object proposals" to be considered by other methods. For example, in Silberman *et al*. [12] they perform an over-segmentation, and then iteratively merge regions using classifiers which predict whether regions belong to the same object instance. These are then classified using an ensemble of features with a logistic regression classifier.

Couprie *et al*. [1] take a different approach, instead using a multi-scale CNN to classify the full image, and then use superpixels to aggregate and smooth prediction outputs. While this allows them to extract a per-pixel semantic segmentation, they fail to achieve very high scores in important classes, such as table and chair. Hariharan *et al*. [6] also predict pixel-level class associations, but classify region proposals instead of the full image. They also use a CNN as a feature extractor on these regions, before classifying into categories with an SVM and aggregating onto a coarse mask. They then use a second classifier stage on this coarse mask projected on to superpixels to extract a detailed segmentation. While these results are interesting, we question the overall utility of such a fine grained segmentation, as it does not provide pose with respect to a class-level representation.

Song and Xiao [14] use renderings of 3D models from many viewpoints to obtain synthetic depth maps for training an ensemble of Exemplar-SVM classifiers. They use a 3D sliding window to obtain proposals during testing and perform non-maximum suppression to obtain bounding boxes. While this 3D sliding window approach is able to handle occlusions and cluttered scenes well, it is very expensive (tens of minutes per image), requiring testing of many windows on many separate detector classifiers.

Guo and Hoiem [3] predict support surfaces (such as tables and desks) in single view RGB-D images using a bottom up approach which aggregates low-level features (*e.g.* edges, voxel occupancy). These features are used to pro-

pose planar surfaces, which are then classified using a linear SVM. While they provide object-class pose annotations for the NYUv2 set which we use in this paper, they do not classify objects or their pose themselves.

Object detection in RGB-D is addressed directly by Gupta *et al*. [5] using a CNN which classifies bounding-box proposals in a room-centric embedding. As with other approaches, they use superpixels to aggregate their classifier results in order to get class instance segmentations. Lin *et al*. [10] use candidate cuboids, rather than bounding boxes, and classify them using a CRF approach. While they achieve good overall classification performance, they merge similar classes (such as table and desk), and while their cuboids give them spatial extent of objects, they do not give pose.

In contrast to the above methods, we do not need expensive and difficult to obtain annotated ground truth data for training. Instead, we use synthetic renderings of scenes containing 3D models pulled from the Internet. While these models need to be aligned to a common pose, this is a relatively inexpensive operation which has already been performed in the ModelNet10 database [16].

The only other work to address pose directly, that of Gupta *et al*. [4], suffers from using unrealistic training data - training instances are single objects rendered in empty space. In contrast, our synthetic data is cluttered and contains realistic noise, as we use a camera model which closely replicates Kinect-like sensors. Because of this, our trained networks are far more effective on real data - we test on the full NYU dataset, while they must leave out instances that have many (>50%) missing depth pixels. Additionally, since we work with full scenes rather than single object instances, our model is trained on and can thus handle inter-object occlusions, rather than only self-occlusions. Moreover, their network contains separate top-level layers for each object class, while we only need a single output network for pose for all classes. Their method is also computationally demanding, requiring about a minute per image per class, while ours runs in a few seconds for all classes.

2. Synthetic RGB-D Scenes

One of the main obstacles to using deep CNNs on RGB-D data is the lack of large annotated datasets. This is especially true for pose data, where annotation of a set of the size required for training a deep network is simply not feasible. Synthetic data, on the other hand, provides labeled segmentations and exact pose for free, but has yet to find widespread use, likely owning to the difficulty of rendering photo-realistic scenes. Fortunately, RGB-D data lends itself to the use of synthetic data due to the simplicity with which depth data can be rendered realistically. One only needs to simulate the active model of the sensor, and can largely ignore lighting, textures, and surface composition.

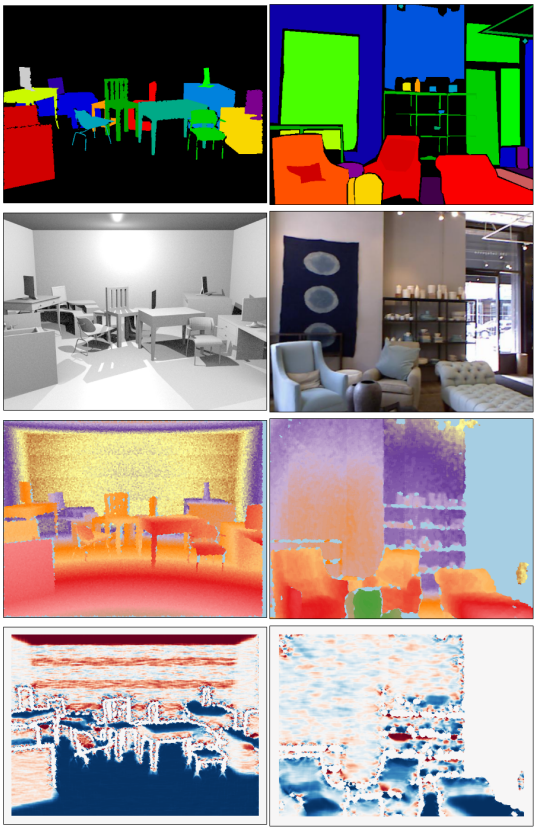


Figure 3. Example of a randomly generated synthetic scene using our rendering pipeline (left) and a scene from the NYUv2 dataset. The rows show A. Ground truth labels, B. RGB Channel, C. Depth Channel, D. Normals calculated using [7]. The left column shows our synthetic data, and the right an image from NYUv2 [12].

Our synthetic scenes are produced by sequentially placing objects models at random in a virtual room. As each object is placed, we ensure that its mesh does not intersect with other objects or the room surfaces. Additionally, we use context cues to increase the realism of our scenes - large furniture (*e.g*. sofas or beds) is biased to occur near walls, chairs are biased to occur near tables and desks, and monitors are always placed on top of desks. We also randomly place a light source on the ceiling in the room to simulate shadow effects in the rendered intensity images. An example random scene is shown in Fig. 3. We have published the dataset used in this work for use by the community, and have also included the code for easily generating more scenes on the fly at training time[1].

[4]

[1]–Website removed for blind review–

2.1. Rendering & Camera Model

We build upon the BlenSor sensor simulation toolbox [2] to generate realistic RGB-D renderings of our randomly generated scenes. The ray-tracing used allows us to reproduce the real geometry of the Kinect sensor, faithfully simulating the projection of an IR pattern onto the scene and observation of the returns. As Kinect-type sensors will generally fail when reflections are present, we can safely limit our ray-tracing to a single hop. Additionally, we simulate the 9x9 correlation window required by the Kinect to produce depth measurements [13] and add Perlin noise to the disparity measurements. We also use a standard Blender pipeline to render accompanying RGB images, though these are not photo-realistic due to a lack of textures on the object models and a simplified lighting model. As we only use the intensity channel, we found this simple RGB rendering to be sufficient, especially given that we use transfer learning to adapt to real sensor images.

2.2. Models

Our models must be aligned to a reference pose, preventing us from simply pulling CAD models from the Internet. Fortunately, the Princeton ModelNet10 dataset [16] provides a varied set of pose-aligned models for ten object categories: bathtub, bed, chair, desk, dresser, monitor, nightstand, sofa, table, and toilet. We use the standard training/testing split provided. As the models are not scale-normalized, we choose a reasonable range of values per class, and rescale models randomly to fall within these ranges. Models are inserted on the floor or a supporting surface of our synthetic rooms at random locations with random rotations around the axis perpendicular to the floor.

3. Network Architecture

We tested several different network configurations, all of which involved at least two Krizhevsky-style [9] (*i.e.* Conv-ReLU-Pooling) convolutional layers at the input. Our most successful model, shown in Fig. 5, then uses a succession of Network-in-Network (NiN) layers [11], in a configuration similar to the recent "Inception" architecture [15]. We then use separate multilayer perceptrons with two hidden layers to classify. Additionally, we connect our class output back into the second hidden layer of our pose and position classifiers.

3.1. Input Preprocessing

The input to our network is a 96x96 real-valued image consisting of five layers - an intensity layer, a depth layer, and three layers representing the surface normal vector (*e.g.* $(normal_x, normal_y, normal_z)$). Depth values are used directly (in meters) and intensity values are computed from RGB using CIE 1931 linear luminance coefficients. While

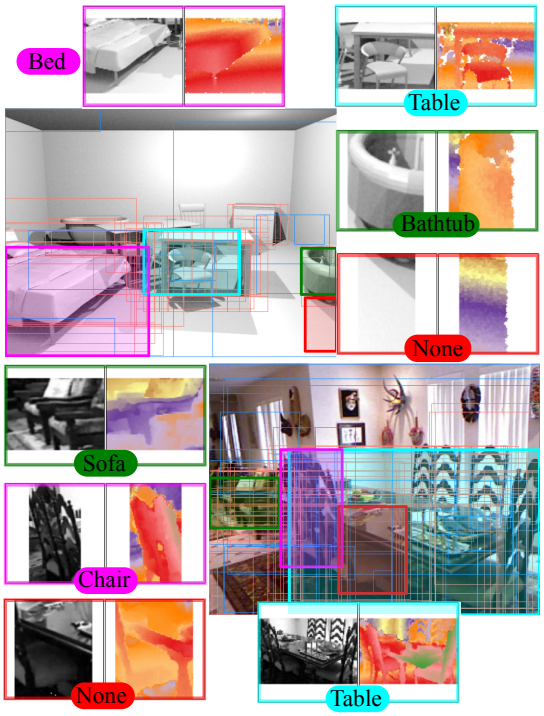


Figure 4. Example of bounding box proposals on synthetic data (top) and the NYUv2 Dataset[12] (bottom).

hue information is likely useful, our synthetic models are not colored, so we chose not to use it. We exploit the structured nature of RGB-D data to efficiently compute surface normals using the method of Holzer *et al.* [7]. All channels are zero centered using mean values computed on a random sample of proposed bounding boxes from our training set.

3.2. Proposal Generation

Bounding box proposals are generated using the Geodesic Object Proposals (GOP) of Krhenbhl and Koltun [8]. The method identifies level sets in geodesic distance transforms for seed points which are placed using classifiers optimized for object discovery. The method produces accurate and consistent bounding boxes at a low computational cost (approx. 1 second per image). Examples of proposed bounding boxes on our synthetic rendered images as well as on the NYUv2 images are shown in Fig. 4. We do not consider depth when generating our proposals, as we did not find it to be helpful in practice - a result supported by other researchers [12].

3.3. Network Layers

We tested four models in total: two "standard" Krizhevsky-style CNNs, and two larger networks with "inception"-style layers. The first, baseline, model is a stan-

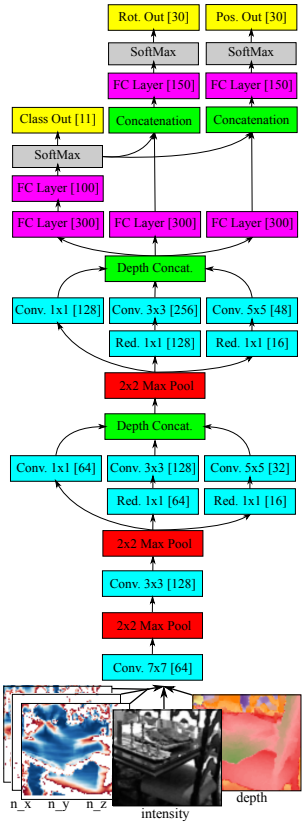


Figure 5. Network architecture of our most successful model. Numbers in brackets are either number of filters (conv. layers) or nodes (FC layers). The input consists of 96x96 5 channel images with normals, intensity, and depth.

dard CNN network closely resembling the successful model of Krizhevsky *et al*. [9] - it consisted of five Conv-ReLU-Pooling layers, followed by two fully-connected (FC) classification layers for each output layer. The second model takes the class output and reconnects it back into the fully connected layers for pose and depth estimation. The third model expands the network by replacing the top convolutional layers with two inception-style[15] network-in-network layers. Lastly, the largest model increases the number of nodes even further by adding another inception layer, as well as an additional FC multi-layer network branching off from the first inception layer and reconnecting as an additional input to the classification FC layers. Dropout was used on the convolutional layers as well as the fully connected (FC) layers of the perceptrons in all models to limit over-fitting. Our most successful model is shown in Fig.5.

4. Training

We train our networks to predict three outputs: a class label, a rotation around the floor normal axis, and a distance from the camera. Combined with a bounding box in the image plane, these allow us to generate a full description of the pose of furniture with respect to the set of standard reference models used by Guo and Hoeim [3]. We chose to predict binned rotation and depth values rather than perform a regression as we found that, in practice, the training was much more stable for classification, even with the relatively large number of bins ($n = 30$, *i.e.* 12 degrees per bin) used. We use a standard SoftMax cross-entropy loss for the class output, but adopt a soft-binning scheme for the pose and depth outputs. This takes a weighted (by γ) average of the local bins around the ground truth in the loss function, in order to help with poses which lie near bin boundaries:

$$L_i = -\log\left(\frac{\sum_{k=-1}^{1}\gamma_k e^{f_{y_{i+k}}}}{\sum_j e^{f_j}}\right) : \sum_{k=-1}^{1}\gamma_k = 1. \quad (1)$$

4.1. Synthetic Data

While we can generate unlimited data at training time, for comparison purposes we trained on a fixed set of 7000 randomly generated scenes, consisting of a total of 59784 instances from our set of 2842 pose-aligned models from the ModelNet10 dataset [16]. There is no constraint on the number of synthetic scenes possible - we only limited ourselves due to time constraints and in order to compare models. Our validation set was generated randomly during training. Additionally, we generated a test set of 1000 random scenes, using a separate set of 812 models from the same dataset. For training, we extracted bounding boxes using GOP and selected those that had 70% overlap with the ground truth, leading to a total of 300,000 training instances. We scale bounding box proposals to fit our input size by fitting the larger dimension to our window size and zero padding the other.

Additionally, we randomly selected an equal number of "none"-class instances for training from the set of proposals containing less than 30% of an object ground-truth box. To avoid biasing our networks, we assign uniformly distributed random poses to these, and assign depths as the centroid of points in the window. Over the course of training, proportion of "none" exemplars used was gradually reduced to help with pose and depth estimation performance for the other classes. Additionally, we experimented with training using a loss function specific to only one task (class, pose or depth) after training on the full combined task, but found no benefit to doing so - the specialized loss function (and gradients computed from it) did not allow the models to increase their performance in the selected task in a significant way.

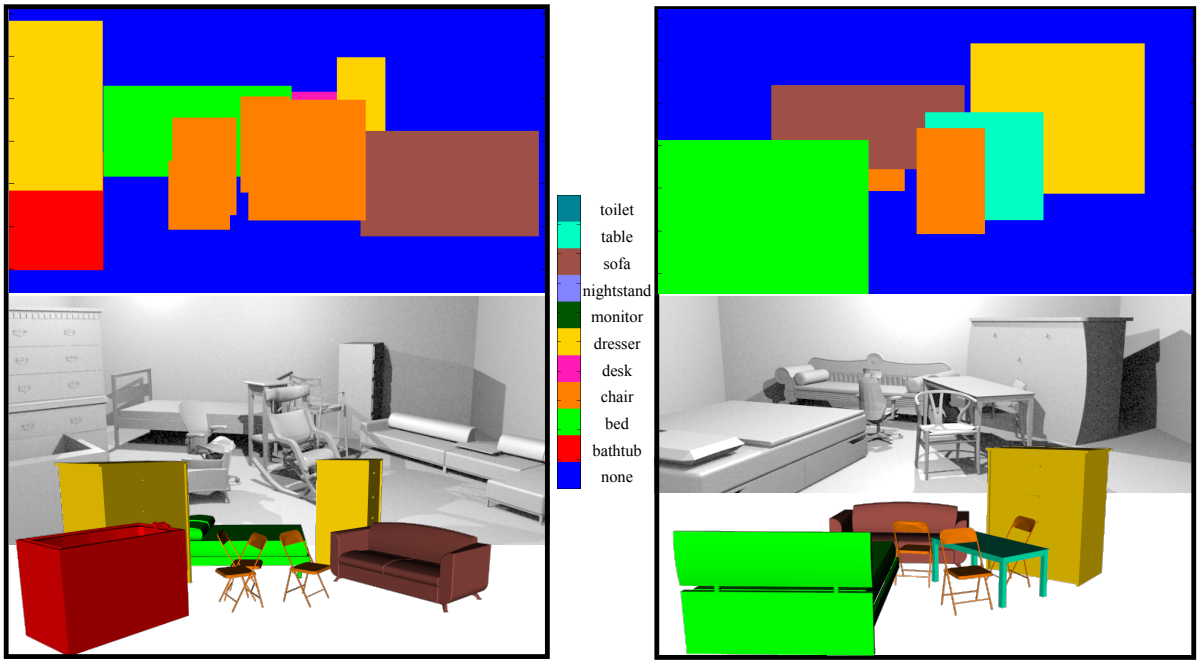


Figure 6. Qualitative pose and classification results from our synthetic test set. The top row shows our estimated semantic heatmap, while the bottom row shows pose and classification using generic models. The models in the test set are distinct from those used in training - this means that poses here are general class-based pose, rather than specific model-based.

Training times ranged from approximately 12 hours for the simpler models to up to 48 hours for the most complex model on a Titan X GPU. For the largest model we were constrained by memory (12Gb) to using a relatively small batch size - we would expect better performance with larger batches.

4.2. Transfer to Real Data

In order to improve performance on the NYUv2 dataset [12], we use transfer learning to adapt our synthetically trained networks to the new, more difficult, domain. We experimented with three strategies for adaptation: 1. Only allow the high-level layers in the network to adapt, keeping the two lowest-level layers fixed as a "feature-extractor" (we choose two layers based on [17]), 2. Allow all levels of the network to adapt, and 3. Alternate training iterations between full-adaptation iterations and iterations on synthetic data, with the proportion of synthetic data being reduced over the course of training. To avoid over-fitting as well as to allow adaptation of the none-class, we use bounding box proposals for training (in addition to the ground truth boxes).

5. Experimental Evaluation

In this section we evaluate the performance of our trained models on each sub-task independently. We show results for both our synthetic test set as well as on the NYUv2 dataset of Silberman *et al*. [12]. When possible, we compare to the state of the art, and show how our method both outperforms and is subject to fewer constraints - primarily because we train directly on cluttered noisy data. We also evaluate our different network architectures on our own synthetic test set. Finally, we present qualitative results which show the ability of our method to provide semantic understanding and pose for full scenes.

5.1. Architectures

We first compare results from our four different network architectures of increasing complexity. In Fig.7 we give per-category and averaged results for all four models on all three sub-tasks. As can be seen, the difference between the models is not very substantial - leading us to believe that we were actually limited by the size of our training set. This seems to be confirmed by the fact that the larger two networks began over-fitting towards the end of their training runs.

	Task	Model	**ACC**	bathtub	bed	chair	desk	dresser	monitor	nightstand	sofa	table
Synthetic	**Classification**	Standard CNN	76.07	30.49	68.90	88.55	56.29	71.87	80.30	57.24	69.94	80.47
		Standard CNN + Recomb.	77.63	23.63	71.69	90.67	57.80	73.96	85.30	50.46	71.35	82.43
		Incept. CNN + Recomb.	78.33	41.82	71.24	90.70	54.56	70.81	81.70	57.51	74.94	82.18
		Large Incept. CNN + Recomb.	76.42	45.02	69.82	87.63	53.58	72.48	78.95	52.22	75.44	81.30
			Mean AUC	bathtub	bed	chair	desk	dresser	monitor	nightstand	sofa	table
	Pose Estimation	Standard CNN	40.30	19.25	40.83	41.40	33.32	29.95	41.29	22.83	53.32	38.92
		Standard CNN + Recomb.	40.13	23.94	38.49	41.88	33.15	30.66	39.63	23.35	54.27	36.31
		Incept. CNN + Recomb.	39.68	19.65	43.29	40.52	32.47	31.90	42.75	24.55	53.68	35.06
		Large Incept. CNN + Recomb.	39.70	20.93	43.03	39.70	37.87	35.13	44.65	22.95	55.79	33.84

	Task	Model	**ACC**	bathtub	bed	chair	desk	dresser	monitor	nightstand	sofa	table
NYUv2	**Classification**	Incept. CNN + Recomb.	51.04	30.30	52.71	70.98	26.76	33.64	19.05	39.32	58.21	39.54
		Large Incept. CNN + Recomb.	51.44	14.29	55.05	70.93	18.05	34.59	27.45	51.13	55.88	40.68
			Mean AUC	bed	chair	desk	sofa	table				
	Pose Est.	Incept. CNN + Recomb.	29.66	31.14	27.58	23.30	39.77	29.68				
		Large Incept. CNN + Recomb.	31.87	35.05	28.20	28.73	47.28	32.21				

Figure 7. Performance of the four CNN architectures on synthetic (top) and NYUv2 (bottom) datasets for classification and pose estimation. Per class classification results show the F1-score, per class pose estimation results show the normalized AUC measure. See Secs. 5.2 and 5.3 for details. All numbers are in percent.

5.2. Classification

To evaluate classification performance, we classify the ground truth bounding boxes from our synthetic test set and the NYUv2 dataset. Fig. 7 gives per class accuracy on both datasets. As an overall measurement for the classification we use accuracy (ACC) being the fraction of correctly classified samples across all classes. Per class performance is measured using the F1-score as the harmonic mean of recall and precision. Unfortunately, while we would like to compare to other works, each recent work has reported classification accuracy slightly differently. Lin *et al*. [10] merge similar classes (*e.g*. table and desk), and only report an overall number. Couprie *et al*. [1] only report pixel-wise accuracy, which we do not compute, as we do not need such a fine-grained segmentation. Gupta *et al*. [4] do not evaluate their classification independently and instead give detector average precision (AP).

5.3. Pose

To measure absolute pose estimation performance, we evaluate the estimated pose per class against ground truth poses. As all objects are located on the floor plane (or in the case of monitors, a horizontal supporting surface), we need only estimate a rotation around the floor normal. For our synthetic set we compare against the ground truth poses used to render the data, while for the NYUv2 dataset we use the annotations of Guo and Hoeiem [3]. We only include results for the 5 of our trained classes for which Guo and Hoeiem provided annotation (bed, chair, desk, sofa, and table). For both synthetic and real datasets, we use the ground truth boxes as our input to isolate pose estimation performance.

To compute a real valued pose and depth we extract the value of the maximum bin and its two neighbors, and compute a weighted sum using the bin centers, *i.e*.

$$\theta = \frac{\sum_{i=-1}^{1} \theta_{hist}(\kappa+i) * \theta_{\kappa+i}}{\sum_{i=-1}^{1} \theta_{hist}(\kappa+i)} : \kappa = \operatorname*{argmax}_{k} \theta_{hist}(k), \tag{2}$$

where θ_κ is the angle at the center of bin κ, and $\theta_{hist}(\kappa)$ is the soft-maxed value of bin κ. We only consider the local distribution around the max bin so that our estimates are not corrupted by multi-peaked histogram distributions (which occur due to rotational symmetries). We should also note that 90 degree rotational symmetries are an unavoidable source of error for some of our classes, especially tables and night stands. To evaluate pose error, we adopt the measure of [4], which plots the accuracy vs increasing allowed angular error δ_θ. To retrieve a scalar performance measure for the pose estimation we use a normalized Area-Under-Curve (AUC) for threshold values up to 15 degrees. For overall performance we average the values weighted by number of instances per class. As seen in Fig. 8, we strongly outperform the state of the art [4] in two classes, with slightly poorer performance in the other.

[4]

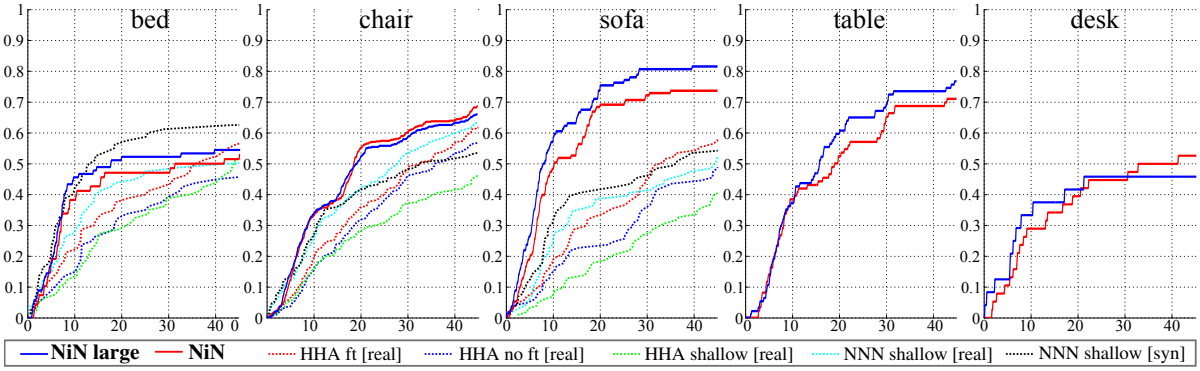


Figure 8. Pose estimation performance five classes in the NYUv2 [12] test set. We plot accuracy versus allowed angular error δ_θ. Our methods (NiN and NiN large - solid lines) outperform the state of the art results of Gupta *et al*. [4].

Figure 9. Qualitative pose and classification results on the NYUv2 dataset. Within each pair: Top: Original Scene; Bottom: Point-cloud with generic pose-aligned models inserted. Note that classification and pose-estimation are model independent. For visualization we used a random model per class.

5.4. Qualitative Results

To combine our classifier results, we first use non-maximum suppression (NMS) to disentangle and remove multiple detections with an allowed overlap of 20%. Then we combine all bounding box activations using a per-pixel max-pooling scheme. Figure 9 shows an example of the semantic heatmaps generated this way, which give a rough class-wise labeling of the scene. Additionally, we show placed generic 3D models for each detected object to show results of pose estimation.

6. Conclusions

We have presented a method for generating realistic synthetic RGB-D scenes for training vision algorithms to segment, classify, and estimate pose and position of common furniture classes. We then showed that these scenes can be used to train deep CNNs to recognize and estimate pose for objects of the classes trained on, even if the object models tested on were not part of the training set; that is, the networks can be used to solve class-based pose estimation, rather than specific model-based pose as has been the prevailing standard.

Furthermore, we have also demonstrated with several experiments that networks trained on synthetic RGB-D scenes can be adapted easily to work on the most challenging real data available, even if the amount of annotated real data available is relatively small. Moreover, we have accomplished all three tasks within a single network, allowing understanding of full scenes in a matter of seconds on a modern GPU. Furthermore, with our pipeline the amount of training data available is practically limitless, as we generate the next batch while the current scenes are trained on - the only limitation is the number and types of models. Fu-

ture work will expand the pose-aligned classes to include the full ModelNet40 dataset and should add further cues to generate even more realistic procedural scenes. We expect this to allow future researchers to extend even further the complexity and performance of machine learning techniques on RGB-D data.

References

[1] C. Couprie, C. Farabet, L. Najman, and Y. LeCun. Indoor semantic segmentation using depth information. In *First International Conference on Learning Representations (ICLR)*, 2013. 2, 7

[2] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree. BlenSor: Blender Sensor Simulation Toolbox Advances in Visual Computing. volume 6939 of *Lecture Notes in Computer Science*, chapter 20, pages 199–208. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2011. 4

[3] R. Guo and D. Hoiem. Support surface prediction in indoor scenes. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2144–2151. IEEE, 2013. 2, 5, 7

[4] S. Gupta, P. A. Arbeláez, R. B. Girshick, and J. Malik. Aligning 3d models to rgb-d images of cluttered scenes. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015. 3, 7, 8

[5] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision (ECCV)*, pages 345–360. Springer, 2014. 3

[6] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision (ECCV)*, pages 297–312. Springer, 2014. 2

[7] S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli, and N. Navab. Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2684–2689. IEEE, 2012. 1, 3, 4

[8] P. Krähenbühl and V. Koltun. Geodesic object proposals. In *European Conference on Computer Vision (ECCV)*, pages 725–739. Springer, 2014. 1, 4

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 4, 5

[10] D. Lin, S. Fidler, and R. Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1417–1424. IEEE, 2013. 3, 7

[11] M. Lin, Q. Chen, and S. Yan. Network in network. In *First International Conference on Learning Representations (ICLR)*, 2013. 4

[12] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision (ECCV)*, pages 746–760. Springer, 2012. 2, 3, 4, 6, 8

[13] J. Smisek, M. Jancosek, and T. Pajdla. 3d with kinect. In *Consumer Depth Cameras for Computer Vision*, pages 3–25. Springer, 2013. 4

[14] S. Song and J. Xiao. Sliding shapes for 3d object detection in rgb-d images. In *European Conference on Computer Vision (ECCV)*, volume 2, page 6, 2014. 2

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014. 4, 5

[16] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao. 3d shapenets for 2.5 d object recognition and next-best-view prediction. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015. 3, 4, 5

[17] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014. 6

[4]

Summary: Category Perception

In this chapter we treated object perception at the *category* level and contributed to solving two problems:

Problem 1: Modern classification methods, like Deep Convolutional Neural Networks(DCNNs), require large amounts of training data. This is usually addressed by downloading from on-line word-based image-databases with subsequent manual filtering of irrelevant images.

Our contribution: We proposed a method (*TransClean*) for automatically generating task-relevant training sets, when provided with the *category* name as well as a context. The algorithm is able to deal with ambiguous *category* names like *nut* being the food as well as the hardware.

Problem 2: Categorization and Pose Estimation at the category level (PEC)

Our contribution: We introduced a DCNN for concurrent categorization and pose-estimation. It is trained on fully annotated (*i.e.*, category label and pose), synthesized scenes.

The *TransClean* algorithm can be used in two ways: First, an agent can automatically extract context from a spoken or written command, retrieve task-relevant images from the Internet, train on them, and recognize the *object* in the scene. Second, human operators can provide *TransClean* with a descriptive context to generate a training set; instead of manually telling apart relevant from irrelevant images.

The DCNN architecture is able to provide agents with the pose and the *category* of novel objects. Therefore, the agent can apply known motor behavior (*e.g.*, trajectories for manipulations) to those objects.

Outlook: Function Perception

By going from *category* perception to *function* perception, we try to annotate objects not with a *category* but with potential *functionality*: For example, which object can I use best to transport water, which object could be used for drilling a hole, and so on. It turns out that perceiving objects as a combination of parts becomes important.

The whole is more than the sum of its parts.

Aristotle

6

Low-level Object Partitioning (OP) and High-level Function Assignment (OFA)

At the beginning of this thesis we advocated the idea to advance robot perception of objects in an even more general way than what is usually done with Object Categorization (OC). Object Function Assignment (OFA) allows us to analyze objects even more deeply than by merely *assimilating* them into a known *category*.

Having a certain task in mind, we begin by analyzing our surrounding for objects which could accomplish set goal. Interestingly, we are able to think of makeshift solutions by looking at the detailed geometry of objects. This process which probably facilitated usage of natural objects like branches and rocks for early hominids is quite a remarkable development. It does not only separate monkeys and early hominids from all other animals, but it also eventually led to modern tool usage nowadays driven by human ingenuity.

Observed from a machines perspective, resembling human ingenuity is a very hard task. Nevertheless, we show in this chapter how capabilities can arise in an agent if rigged with the necessary algorithms.

As the agent deals with unusual and novel *categories*, our developed algorithms for OC are not applicable anymore. The variety of possible improvised tools is just too large. One way of reducing this large intra-class variance is by deconstructing an object into its constituent parts and by treating objects not as a whole, but rather as an assembly of different parts connected in a specific way. We therefore decided to tackle this problem from two sides. First, by using a set

of generic unsupervised bottom-up algorithms, which can be applied to various objects for retrieving parts. Second, by employing a top-down supervised algorithm, which uses annotated and partitioned objects (*i.e.*, *function* defined for the full object and the parts) for the training and predicts *function* for new objects and their parts. To get rid of some of the intra-class variance of perceived objects, we decided to work solely in 3D without using color. While color is great for recognizing known objects, it is misleading in the context of functional properties and affected by changing illumination. Furthermore, 3D does not suffer from variance coming from perspective distortion, and finally, one cannot expect an agent (or even a human) to analyze a novel object for *functionality* based on only 2D - hence partial observations. We, as humans, would analyze a novel object by observing it from all sides.

6.1 Preprocessing Point Clouds

The first problem when working in 3D is the huge amount of data. To speed-up calculations we, consequently, need an algorithm which is able to compress the information stored in point clouds without loosing important information for the subsequent algorithms. Motivated by the success of superpixels [73–75], which are used for creating over-segmentations[1] for images in 2D (called superpixels), we developed our Voxel Cloud Connectivity Segmentation (VCCS) algorithm for creating over-segmentations for point clouds in 3D (called supervoxels):

[5] Papon, J. and Abramov, A. and **Schoeler, M.** and Wörgötter, F.: "Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. See page 71.

While these supervoxels do not represent meaningful entities as such, they have remarkable properties which make them applicable in various fields of 3D data processing: They do not cross object boundaries, they are efficient to calculate, they store adjacency information in an efficient way, and they estimate point normals[2]. While we only showed in the paper that this compression respects object-to-object boundaries, it turned out that even part-to-part boundaries are mostly preserved.

[1] Over-segmentation is a mid-level representation of a point cloud or an image. By grouping similar pixels or points in an image or point cloud, it is above the pixel level but below the level of meaningful entities like objects or parts - hence mid-level.

[2] Point normals describe the direction which points away from the local surface.

Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds

Jeremie Papon Alexey Abramov Markus Schoeler
Florentin Wörgötter
Bernstein Center for Computational Neuroscience (BCCN)
III Physikalisches Institut - Biophysik, Georg-August University of Göttingen
{jpapon,abramov,mschoeler,worgott}@physik3.gwdg.de

Abstract

Unsupervised over-segmentation of an image into regions of perceptually similar pixels, known as superpixels, is a widely used preprocessing step in segmentation algorithms. Superpixel methods reduce the number of regions that must be considered later by more computationally expensive algorithms, with a minimal loss of information. Nevertheless, as some information is inevitably lost, it is vital that superpixels not cross object boundaries, as such errors will propagate through later steps. Existing methods make use of projected color or depth information, but do not consider three dimensional geometric relationships between observed data points which can be used to prevent superpixels from crossing regions of empty space. We propose a novel over-segmentation algorithm which uses voxel relationships to produce over-segmentations which are fully consistent with the spatial geometry of the scene in three dimensional, rather than projective, space. Enforcing the constraint that segmented regions must have spatial connectivity prevents label flow across semantic object boundaries which might otherwise be violated. Additionally, as the algorithm works directly in 3D space, observations from several calibrated RGB+D cameras can be segmented jointly. Experiments on a large data set of human annotated RGB+D images demonstrate a significant reduction in occurrence of clusters crossing object boundaries, while maintaining speeds comparable to state-of-the-art 2D methods.

1. Introduction

Segmentation algorithms aim to group pixels in images into perceptually meaningful regions which conform to object boundaries. While they initially only considered low-level information from the image, recent semantic segmentation methods take advantage of high-level object knowledge to help disambiguate object borders. Graph-based approaches, such as Markov Random Field (MRF) and Conditional Random Field (CRF), have become popular, as they merge relational low-level context within the image with object level class knowledge. While the use of such techniques have met with significant success, they have the drawback that the computational cost of inference on these graphs generally rises sharply with increasing number of nodes. This means that solving graphs with a node for every pixel quickly becomes intractable, which has limited their use in applications which require real-time segmentation.

The cost of solving pixel-level graphs led to the development of mid-level inference schemes which do not use pixels directly, but rather use groupings of pixels, known as superpixels, as the base level for nodes [9]. Superpixels are formed by over-segmenting the image into small regions based on local low-level features, reducing the number of nodes which must be considered for inference. While this scheme has been successfully used in many state-of-the-art algorithms [4, 15], it suffers from one significant disadvantage; mistakes in the over-segmentation which creates the superpixels generally cannot be recovered from and will propagate to later steps in the vision pipeline.

Due to their strong impact on the quality of the eventual segmentation [5], it is important that superpixels have certain characteristics. Of these, avoiding violating object boundaries is the most vital, as failing to do so will decrease the accuracy of classifiers used later - since they will be forced to consider pixels which belong to more than one class. Additionally, even if the classifier does manage a correct output, the final pixel level segmentation will necessarily contain errors. Another useful quality is regular distribution over the area being segmented, as this will produce a simpler graph for later steps.

In this paper, we present a novel method, Voxel Cloud Connectivity Segmentation (VCCS), which takes advantage of 3D geometry provided by RGB+D cameras to generate superpixels which conform to object boundaries better than existing methods, and which are evenly distributed in the actual observed space, rather than the projected image plane. This is accomplished using a seeding method-

[5]

ology based in 3D space and a flow-constrained local iterative clustering which uses color and geometric features. In addition to providing superpixels which conform to real geometric relationships, the method also can be used directly on point clouds created by combining several calibrated RGB+D cameras, providing a full 3D supervoxel (the 3D analogue of superpixels) graph at speeds sufficient for robotic applications. Additionally, the method source code is freely distributed as part of the Point Cloud Library [11] (PCL) [1].

The organization of the paper is as follows: first, in Section 2 we give an overview of existing methods. In Section 3 we present the 3D supervoxel segmentation algorithm. In Section 4 we present a qualitative evaluation of the method segmenting 3D point clouds created by merging several cameras. In Section 5 we use standard quantitative measures on results from a large RGB+D semantic segmentation dataset to demonstrate that our algorithm conforms to real object boundaries better than other state-of-the-art methods. Additionally, we present run-time performance results to substantiate the claim that our method is able to offer performance equivalent to the fastest 2D methods. Finally, in Section 6 we discuss the results and conclude.

2. Related Work

There are many existing methods for over-segmenting images into superpixels. These can be generally classified into two subsets - graph-based and gradient ascent methods. In this section, we shall briefly review recent top-performing methods.

Graph-based superpixel methods, similar to graph-based full segmentation methods, consider each pixel as a node in a graph, with edges connecting to neighboring pixels. Edge weights are used to characterize similarity between pixels, and superpixel labels are solved for by minimizing a cost function over the graph. Moore *et al*. [8] produce superpixels which conform to a regular lattice structure by seeking optimal paths horizontally and vertically across a boundary image. This is done using either a graph cuts or dynamic programming method which seeks to minimize the cost of edges and nodes in the paths. While this method does have the advantage of producing superpixels in a regular grid, it sacrifices boundary adherence to so, and furthermore, is heavily dependent on the quality of the precomputed boundary image.

The Turbopixels [7] method of Levinshtein *et al*. uses a geometric flow-based algorithm based on level-set, and enforces a compactness constraint to ensure that superpixels have regular shape. Unfortunately, it is too slow for use in many applications; while the authors claim complexity linear in image size, in practice we experienced run times over 10 seconds for VGA-sized images. Veksler *et al*. [13], inspired by Turbopixels, use an energy minimization framework to stitch together image patches, using graph-cuts to optimize an explicit energy function. Their method (referred to here as GCb10) is considerably faster than Turbopixels, but still requires several seconds even for small images.

Recently, a significantly faster class of superpixel methods has emerged - Simple Linear Iterative Clustering[1] (SLIC). This is an iterative gradient ascent algorithm which uses a local k-means clustering approach to efficiently find superpixels, clustering pixels in the five dimensional space of color and pixel location. Depth-Adaptive Superpixels[14] recently extended this idea to use depth images, expanding the clustering space with the added dimensions of depth and point normal angles. While DASP is efficient and gives promising results, it does not take full advantage of RGB+D data, remaining in the class of 2.5D methods, as it does not explicitly consider 3D connectivity or geometric flow.

For the sake of clarity, we should emphasize that our method is not related to existing "supervoxel" methods [1, 8, 13], which are simple extensions of 2D algorithms to 3D volumes. In such methods, video frames are stacked to produce a structured, regular, and solid volume with time as the depth dimension. In contrast, our method is intended to segment actual volumes in space, and makes heavy use of the fact that such volumes are not regular or solid (most of the volume is empty space) to aid segmentation. Existing "supervoxel" methods cannot work in such a space, as they generally only function on a structured lattice.

3. Geometrically Constrained Supervoxels

In this Section we present Voxel Cloud Connectivity Segmentation (VCCS), a new method for generating superpixels and supervoxels from 3D point cloud data. The supervoxels produced by VCCS adhere to object boundaries better than state-of-the-art methods while the method remains efficient enough to use in online applications. VCCS uses a variant of k-means clustering for generating its labeling of points, with two important constraints:

1. The seeding of supervoxel clusters is done by partitioning 3D space, rather than the projected image plane. This ensures that supervoxels are evenly distributed according to the geometry of the scene.

2. The iterative clustering algorithm enforces strict spatial connectivity of occupied voxels when considering points for clusters. This means that supervoxels strictly cannot flow across boundaries which are disjoint in 3D space, even though they are connected in the projected plane.

First, in 3.1 we shall describe how neighbor voxels are calculated efficiently, then in 3.2 how seeds are generated and filtered, in 3.3 the features and distance measure used

[1] https://github.com/PointCloudLibrary/pcl/

for clustering, and finally in 3.4 how the iterative clustering algorithm enforces spatial connectivity. Unless otherwise noted, all processing is being performed in the 3D point-cloud space constructed from one or more RGB+D cameras (or any other source of point-cloud data). Furthermore, because we work exclusively in a voxel-cloud space (rather than the continuous point-cloud space), we shall adopt the following notation to refer to voxel at index i within voxel-cloud V of voxel resolution r:

$$V_r(i) = \mathbf{F}_{1..n}, \tag{1}$$

where $\mathbf{F}$ specifies a feature vector which contains n point features (*e.g.* color, location, normals).

3.1. Adjacency Graph

Adjacency is a key element of the proposed method, as it ensures that supervoxels do not flow across object boundaries which are disconnected in space. There are three definitions of adjacency in a voxelized 3D space; 6-,18-, or 26-adjacent. These share a face, faces or edges, and faces, edges, or vertices, respectively. In this work, whenever we refer to adjacent voxels, we are speaking of 26-adjacency.

As a preliminary step, we construct the adjacency graph for the voxel-cloud. This can be done efficiently by searching the voxel kd-tree, as for a given voxel, the centers of all 26-adjacent voxels are contained within $\sqrt{3} * R_{voxel}$. R_{voxel} specifies the voxel resolution which will be used for the segmentation (for clarity, we shall simply refer to discrete elements at this resolution as voxels). The adjacency graph thus constructed is used extensively throughout the rest of the algorithm.

3.2. Spatial Seeding

The algorithm begins by selecting a number of seed points which will be used to initialize the supervoxels. In order to do this, we first divide the space into a voxelized grid with a chosen resolution R_{seed}, which is significantly higher than R_{voxel}. The effect of increasing the seed resolution R_{seed} can be seen in Figure 2. Initial candidates for seeding are chosen by selecting the voxel in the cloud nearest to the center of each occupied seeding voxel.

Once we have candidates for seeding, we must filter out seeds caused by noise in the depth image. This means that we must remove seeds which are points isolated in space (which are likely due to noise), while leaving those which exist on surfaces. To do this, we establish a small search radius R_{search} around each seed, and delete seeds which do not have at least as many voxels as would be occupied by a planar surface intersecting with half of the search volume (this is shown by the green plane in Figure 1). Once filtered, we shift the remaining seeds to the connected voxel within the search volume which has the smallest gradient in the search volume. Gradient is computed as

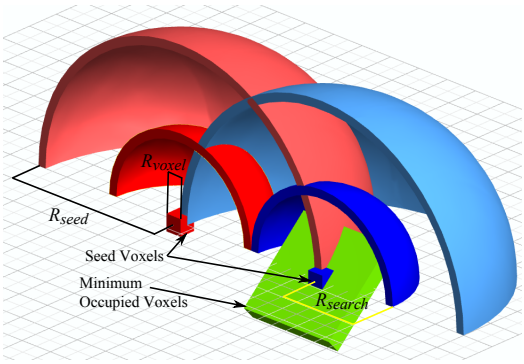


Figure 1. Seeding parameters and filtering criteria. R_{seed} determines the distance between supervoxels, while R_{voxel} determines the resolution to which the cloud is quantized. R_{search} is used to determine if there are a sufficient number of occupied voxels to necessitate a seed.

$$G(i) = \sum_{k \in V_{adj}} \frac{\parallel V(i) - V(k) \parallel_{CIELab}}{N_{adj}}; \tag{2}$$

we use sum of distances in CIELAB space from neighboring voxels, requiring us to normalize the gradient measure by number of connected adjacent voxels N_{adj}. Figure 1 gives an overview of the different distances and parameters involved in seeding.

Once the seed voxels have been selected, we initialize the supervoxel feature vector by finding the center (in feature space) of the seed voxel and connected neighbors within 2 voxels.

3.3. Features and Distance Measure

VCCS supervoxels are clusters in a 39 dimensional space, given as

$$\mathbf{F} = [x, y, z, L, a, b, \text{FPFH}_{1..33}], \tag{3}$$

where x, y, z are spatial coordinates, L, a, b are color in CIELab space, and $\text{FPFH}_{1..33}$ are the 33 elements of Fast Point Feature Histograms (FPFH), a local geometrical feature proposed by Rusu *et al.* [10]. FPFH are pose-invariant features which describe the local surface model properties of points using combinations of their k nearest neighbors. They are an extension of the older Point Feature Histograms optimized for speed, and have a computational complexity of $O(n \cdot k)$.

In order to calculate distances in this space, we must first normalize the spatial component, as distances, and thus their relative importance, will vary depending on the seed resolution R_{seed}. Similar to the work of Achanta *et al.*, [1] we have limited the search space for each cluster so that it

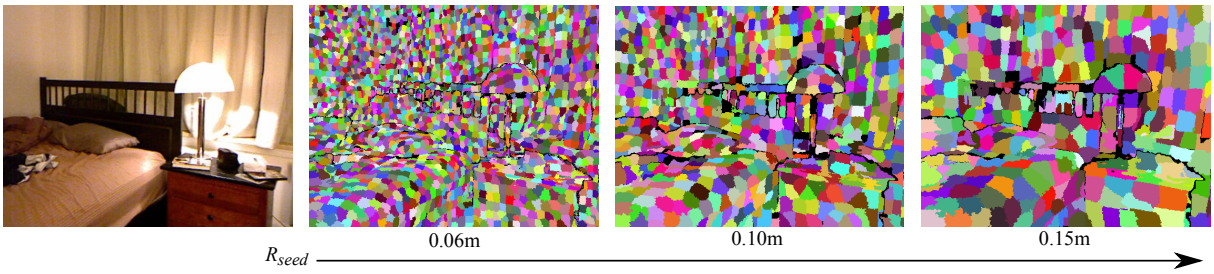


Figure 2. Image segmented using VCCS with seed resolutions of 0.1, 0.15 and 0.2 meters.

ends at the neighboring cluster centers. This means that we can normalize our spatial distance D_s using the maximally distant point considered for clustering, which will lie at a distance of $\sqrt{3}R_{seed}$. Color distance D_c, is the euclidean distance in CIELab space, normalized by a constant m. Distance in FPFH space, D_f, is calculated using the Histogram Intersection Kernel [2]. This leads us to a equation for normalized distance D:

$$D = \sqrt{\frac{\lambda D_c^2}{m^2} + \frac{\mu D_s^2}{3R_{seed}^2} + \epsilon D_{HiK}^2}, \quad (4)$$

where λ, μ, and ϵ control the influence of color, spatial distance, and geometric similarity, respectively, in the clustering. In practice we keep the spatial distance constant relative to the other two so that supervoxels occupy a relatively spherical space, but this is not strictly necessary. For the experiments in this paper we have color weighted equally with geometric similarity.

3.4. Flow Constrained Clustering

Assigning voxels to supervoxels is done iteratively, using a local k-means clustering related to [1, 14], with the significant difference that we consider connectivity and flow when assigning pixels to a cluster. The general process is as follows: beginning at the voxel nearest the cluster center, we flow outward to adjacent voxels and compute the distance from each of these to the supervoxel center using Equation 4. If the distance is the smallest this voxel has seen, its label is set, and using the adjacency graph, we add its neighbors which are further from the center to our search queue for this label. We then proceed to the next supervoxel, so that each level outwards from the center is considered at the same time for all supervoxels. We proceed iteratively outwards until we have reached the edge of the search volume for each supervoxel (or have no more neighbors to check).

This amounts to a breadth-first search of the adjacency graph, where we check the same level for all supervoxels before we proceed down the graphs in depth. Importantly, we avoid edges to adjacent voxels which we have already checked this iteration. The search concludes for a supervoxel when we have reached all the leaf nodes of its adjacency graph or none of the nodes searched in the current level were set to its label. This search procedure, illustrated in Figure 3, has two important advantages over existing methods:

1. Supervoxel labels cannot cross over object boundaries that are not actually touching in 3D space, since we only consider adjacent voxels, and

2. Supervoxel labels will tend to be continuous in 3D space, since labels flow outward from the center of each supervoxel, expanding in space at the same rate.

Once the search of all supervoxel adjacency graphs has concluded, we update the centers of each supervoxel cluster by taking the mean of all its constituents. This is done iteratively; either until the cluster centers stabilize, or for a fixed number of iterations. For this work we found that the supervoxels were stable within a few iterations, and so have simply used five iterations for all presented results.

4. Three Dimensional Voxel Segments

The proposed method works directly on voxelized point clouds, which has advantages over existing methods which operate in the projected image plane. The most important of these is the ability to segment clouds coming from many sensor observations - either using multiple cameras [3] or accumulated clouds from one [6]. Computationally, this is advantageous, as the speed of our method is dependent on the number of occupied voxels in the scene[2], and not the number of observed pixels. As observations will have significant overlap, this means that it is cheaper to segment the overall voxel cloud than the individual 2D observations. For instance, the scene in Figure 5 comes from 180 Kinect observations (640x480), and yet the final voxel cloud (with $R_{voxel} = 0.01m$) only contains 450k voxels.

Additionally, while VCCS will become more accurate as cloud information is filled in by additional observations, 2D methods must necessarily segment them independently and therefore cannot make use of the added information. Most

[2] We should note that while the initial voxelization of the cloud does take more time with a larger cloud, it remains insignificant overall

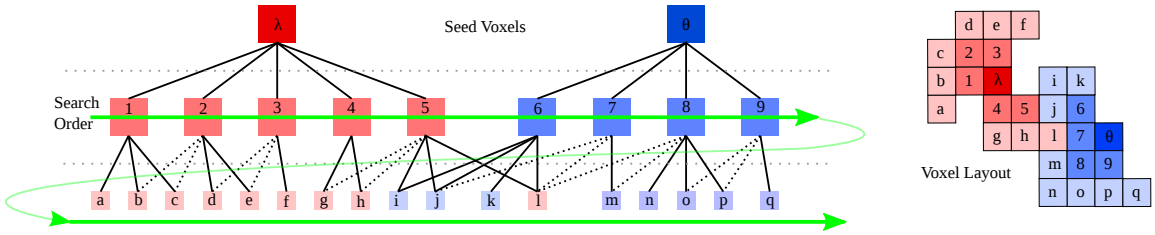


Figure 3. Search order for the flow constrained clustering algorithm (shown in 2D for clarity). Dotted edges in the adjacency graph are not searched, as the nodes have already been added to the search queue.

importantly, even with methods that use depth information, such as that of Weikersdorfer *et al*. [14], it is not clear how one would combine the multiple segmented 2d images, as superpixels from sequential observations will have no relation to each other and will have conflicting partitionings of space in the merged cloud.

5. Experimental Evaluation

In order to evaluate the quality of supervoxels generated by VCCS, we performed a quantitative comparison with three state-of-the-art superpixel methods using publicly available source code. We selected the two 2D techniques with the highest published performance from a recent review [1]: a graph based method, GCb10 [13][3], and a gradient ascent local clustering method, SLIC [1][4]. Additionally, we selected another method which uses depth images, DASP[14][5]. Examples of over-segmentations produced by the methods are given in Figure 6.

5.1. Dataset

For testing, we used the recently created NYU Depth Dataset V2 semantic segmentation dataset of Silberman *et al*. [12][6]. This contains 1449 pairs of aligned RGB and depth images, with human annotated densely labeled ground truth. The images were captured in diverse cluttered indoor scenes, and present many difficulties for segmentation algorithms such as varied illumination and many small similarly colored objects. Examples of typical scenes are shown in Figure 6.

5.2. Returning to the Projected Plane

RGB+D sensors produce what is known as an organized point cloud- a cloud where every point corresponds to a pixel in the original RGB and depth images. When such a cloud is voxelized, it necessarily loses this correspondence, and becomes an unstructured cloud which no longer has any direct relationship back to the 2D projected plane. As such, in order to compare results with existing 2D methods we were forced to devise a scheme to apply supervoxel labels to the original image.

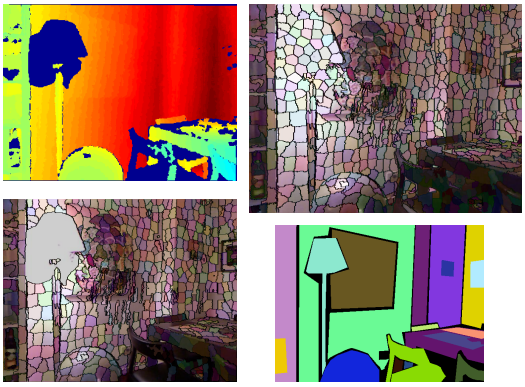


Figure 4. Example of hole-filling for images after returning from voxel-cloud to the projected image plane. Depth data, shown in the top left, has holes in it, shown as dark blue areas (here, due to the lamp interfering with the Kinect). The resulting supervoxels do not cover these holes as shown in the bottom left, since the cloud has no points in them. To generate a complete 2D segmentation, we fill these holes in using the SLIC algorithm, resulting in a complete segmentation, seen in the top right. The bottom right shows human annotated ground truth for the scene.

To do this, we take every point in the original organized cloud and search for the nearest voxel in the voxelized representation. Unfortunately, since there are blank areas in the original depth image due to such factors as reflective surfaces, noise, and limited sensor range, this leaves us with some blank areas in the output labeled images. To overcome this, we fill in any large unlabeled areas using the SLIC algorithm. This is not a significant drawback, as the purpose of the algorithm is to form supervoxels in 3D space, not superpixels in the projected plane, and this hole-filling is only needed for comparison purposes. Additionally, the hole fill-

[3] http://www.csd.uwo.ca/~olga/Projects/superpixels.html
[4] http://ivrg.epfl.ch/supplementary_material/RK_SLICSuperpixels/index.html
[5] https://github.com/Danvil/dasp
[6] http://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html

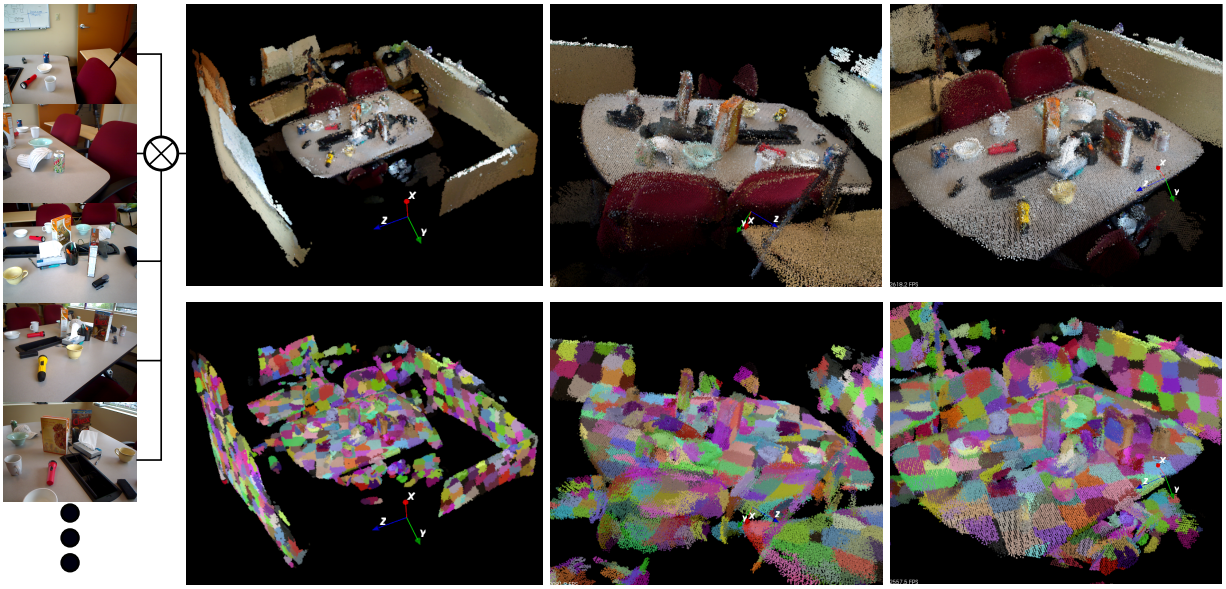


Figure 5. Over-segmentation of a cloud from the RGB-D scenes dataset[6]. The cloud is created by aligning 180 kinect frames, examples of which are seen on the left side. The resulting cloud has over 3 million points, which reduces to 450k points at $R_{voxel} = 0.01m$ and 100k points with $R_{voxel} = 0.02m$. Over-segmentation of these take 6 and 1.5 seconds, respectively (including voxelization).

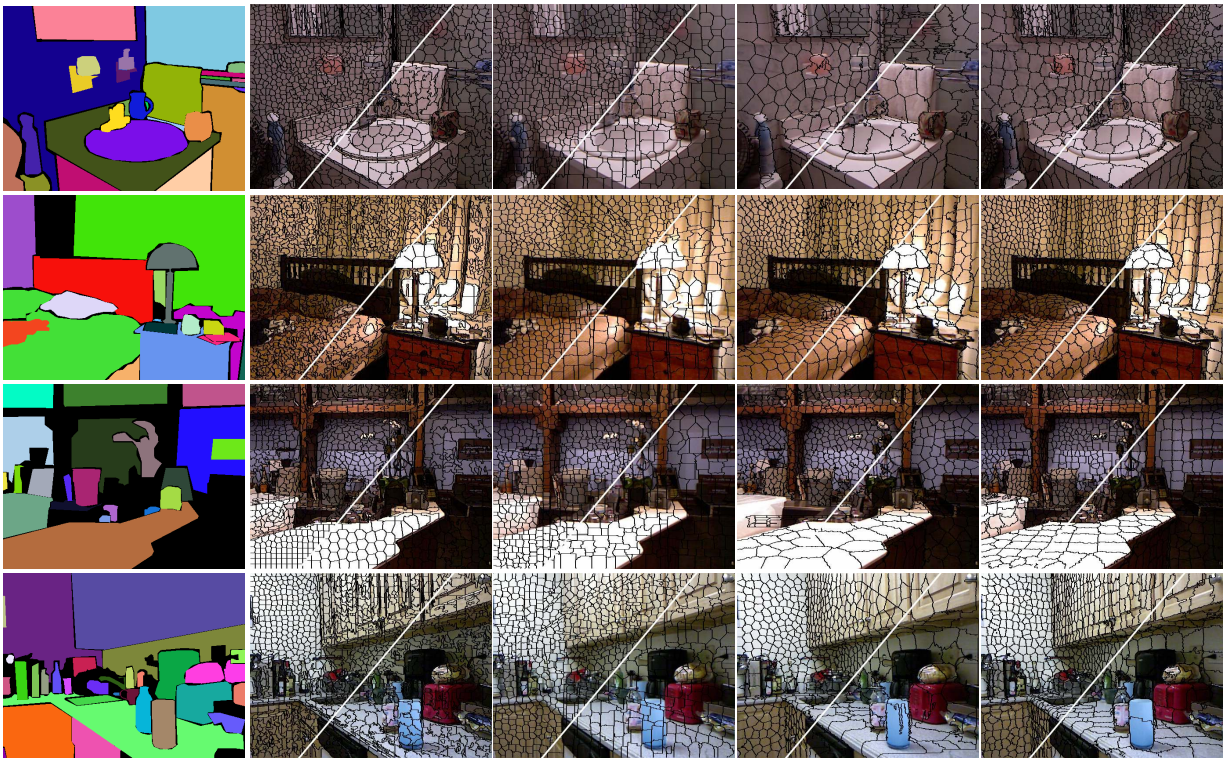


Figure 6. Examples of under-segmentation output. From left to right- ground truth annotation, SLIC, GCb10, DASP, and VCCS. Each is shown with two different superpixel densities.

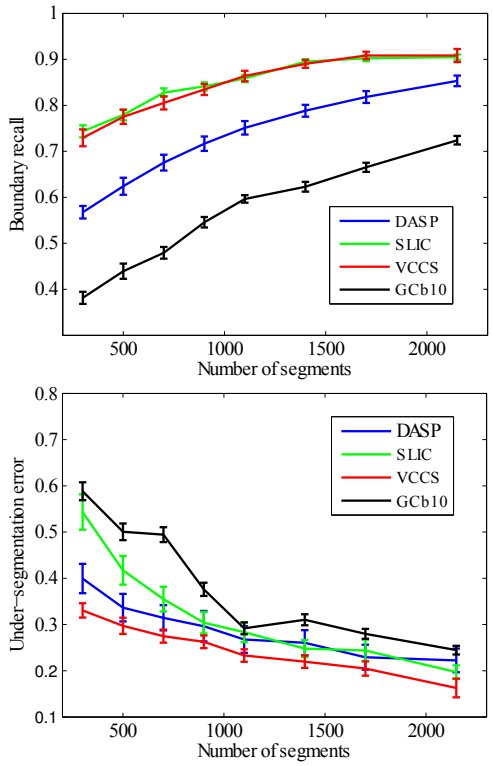


Figure 7. Boundary recall and under-segmentation error for SLIC, GCb10, DASP, and VCCS.

ing actually makes our results worse, since it does not consider depth, and therefore tends to bleed over some object boundaries that were correctly maintained in the supervoxel representation. An example of what the resulting segments look like before and after this procedure are shown in Figure 4.

5.3. Evaluation Metrics

The most important property for superpixels is the ability to adhere to, and not cross, object boundaries. To measure this quantitatively, we have used two standard metrics for boundary adherence- boundary recall and under-segmentation error[7, 13]. Boundary recall measures what fraction of the ground truth edges fall within at least two pixels of a superpixel boundary. High boundary recall indicates that the superpixels properly follow the edges of objects in the ground truth labeling. The results for boundary recall are given in Figure 7. As can be seen, VCCS and SLIC have the best boundary recall performance, giving similar results as the number of superpixels in the segmentation varies.

Under-segmentation error measures the amount of leakage across object boundaries. For a ground truth segmentation with regions $g_1, ..., g_M$, and the set of superpixels from an over-segmentation, $s_1, ...s_K$, under-segmentation error is defined as

$$E_{useg} = \frac{1}{N}\left[\sum_{i=1}^{M}\left(\sum_{s_j|s_j\cap g_i}|s_j|\right) - N\right], \qquad (5)$$

where $s_j \mid s_j \cap g_i$ is the set of superpixels required to cover a ground truth label g_i, and N is the number of labeled ground truth pixels. A lower value means that less superpixels violated ground truth borders by crossing over them. Figure 7 compares the four algorithms, giving under-segmentation error for increasing superpixel counts. VCCS outperforms existing methods for all superpixel densities.

5.4. Time Performance

As superpixels are used as a preprocessing step to reduce the complexity of segmentation, they should be computationally efficient so that they do not negatively impact overall performance. To quantify segmentation speed, we measured the time required for the methods on images of increasing size (for the 2D methods) and increasing number of voxels (for VCCS). All measurements were recorded on an Intel Core i7 3.2Ghz processor, and are shown in Figure 8. VCCS shows performance competitive with SLIC and DASP (the two fastest superpixel methods in the literature) for voxel clouds of sizes which are typical for Kinect data at $R_{voxel} = 0.008m$ (20-40k voxels). It should be noted that only VCCS takes advantage of multi-threading (for octree, kd-tree, and FPFH computation), as there are no publicly available multi-threaded implementations of the other algorithms.

6. Discussion and Conclusions

We have presented VCCS, a novel over-segmentation algorithm for point-clouds. In contrast to existing approaches, it works on a voxelized cloud, using spatial connectivity and geometric features to help superpixels conform better to object boundaries. Results demonstrated that VCCS produces over-segmentations which perform significantly better than the state-of-the-art in terms of under-segmentation error, and equal to the top performing method in boundary recall. This is fortunate, as we consider under-segmentation error to be the more important of the two measures, as boundary recall does not penalize for crossing ground truth boundaries- meaning that even with a high boundary recall score, superpixels might perform poorly in actual segmentation. We have also presented timing results which show that VCCS has run time comparable to the fastest existing methods, and is fast enough for use as a pre-processing step in online semantic segmentation applications such as robotics.

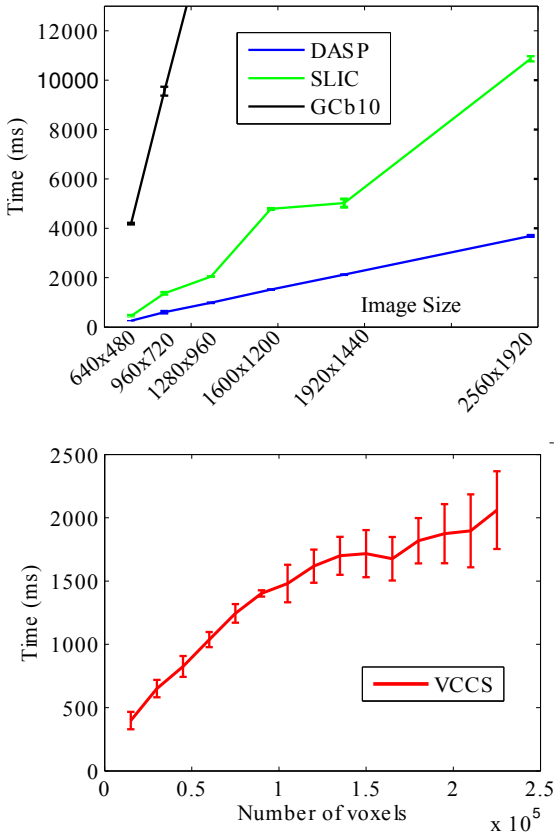


Figure 8. Speed of segmentation for increasing image size and number of voxels. Use of GCb10 rapidly becomes unfeasible for larger image sizes, and so we do not adjust the axes to show its run-time. The variation seen in VCCS run-time is due to dependence on other factors, such as R_{seed} and overall amount of connectivity in the adjacency graphs.

We have made the code publicly available as part of the popular Point Cloud Library, and intend for VCCS to become an important step in future graph-based 3D semantic segmentation methods.

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 and grant agreement no. 269959, Intellact.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Machine Intell.*, 34(11):2274 –2282, nov. 2012. 2, 3, 4, 5

[2] A. Barla, F. Odone, and A. Verri. Histogram intersection kernel for image classification. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, pages III – 513–16 vol.2, sept. 2003. 4

[3] D. A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim. Shake'n'sense: reducing interference for overlapping structured light depth cameras. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 1933–1936, New York, USA, 2012. 4

[4] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 670 –677, oct. 2009. 1

[5] A. Hanbury. How do superpixels affect image segmentation? In *Progress in Pattern Recognition, Image Analysis and Applications*, Lecture Notes in Computer Science, pages 178–186. 2008. 1

[6] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817 –1824, may 2011. 4, 6

[7] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Trans. Pattern Anal. Machine Intell.*, 31(12):2290 –2297, dec. 2009. 2, 7

[8] A. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *Computer Vision and Pattern Recognition, 2008 (CVPR). IEEE Conference on*, pages 1 –8, june 2008. 2

[9] X. Ren and J. Malik. Learning a classification model for segmentation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 10 –17 vol.1, oct. 2003. 1

[10] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009 (ICRA). IEEE International Conference on*, pages 3212 –3217, may 2009. 3

[11] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *Robotics and Automation, 2011 (ICRA). IEEE International Conference on*, Shanghai, China, may 2011. 2

[12] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision ECCV 2012*, volume 7576 of *Lecture Notes in Computer Science*, pages 746–760. 2012. 5

[13] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. In *Computer Vision ECCV 2010*, volume 6315, pages 211–224. 2010. 2, 5, 7

[14] D. Weikersdorfer, D. Gossow, and M. Beetz. Depth-adaptive superpixels. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2087–2090, 2012. 2, 4, 5

[15] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes. Layered object detection for multi-class segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3113 –3120, june 2010. 1

6.2 Getting Objects

Based on the supervoxel algorithm, we created the Locally Convex Connected Patches (LCCP) algorithm, which can be used to segment objects in a scene by analyzing the relations of supervoxels to neighbors:

[6] Stein, S. and Wörgötter, F. and **Schoeler, M.** and Papon, J. and Kulvicius, T.: "Convexity based object partitioning for robot applications", *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. See page 81.

[7] Stein, S. and **Schoeler, M.** and Papon, J. and Wörgötter, F.: "Object Partitioning using Local Convexity", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. See page 89.

The key idea is that human perception of unknown objects and their parts is much facilitated by the existence of concavities. While this theory has been much researched in psychology [64–66, 76], we transfered it to segmenting point clouds: First, we classify edges between adjacent supervoxels as either being concave, convex, or invalid. Invalid connections exist where surfaces touch at a singular point (see [7] Fig. 1). Second, we perform region growing along convex-connected patches, which results in a segmentation which isolates objects even in cluttered scenes. In [7] we additionally propose the so-called Depth Dependent Voxel Grid (DDVG) which treats decreasing point-density at far distances in single view point clouds by scaling the voxel-size with the distance to the camera. This greatly increased applicability to challenging problems, for example, scene-segmentation in the *NYU Depth V2* dataset, for which we achieve results comparable to state-of-the-art supervised methods.

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 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 scstein@physik3.gwdg.de

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

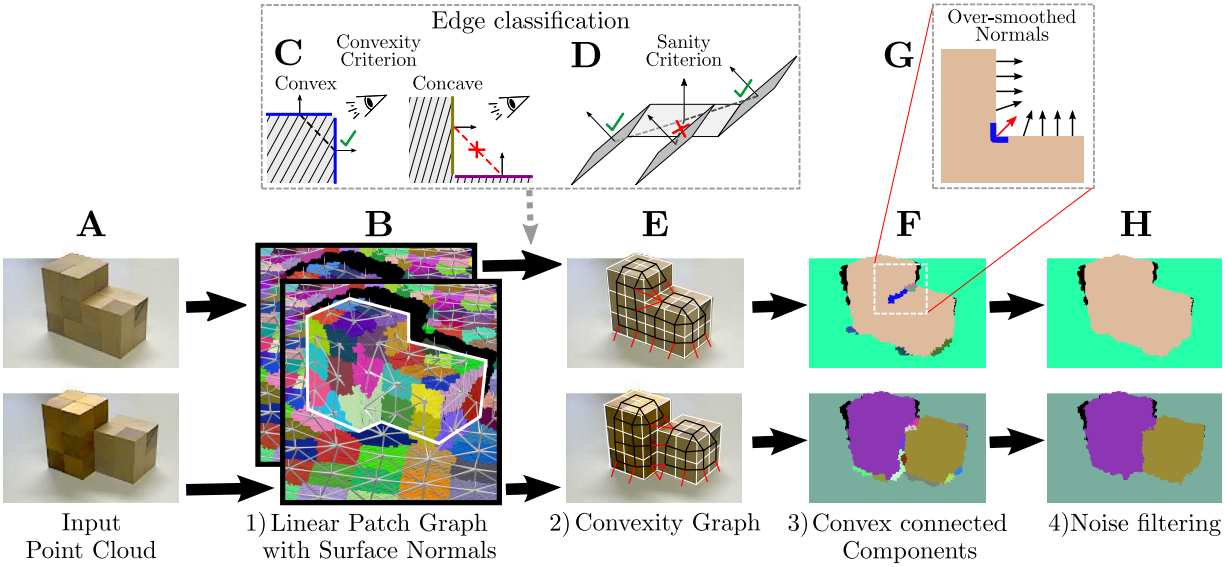


Fig. 1. Flow diagram of the segmentation algorithm. Two example cases are shown: single-object case (upper panels) and two-objects case (lower panels). Illustration of **A)** RGB images corresponding to the point clouds (not shown) of the scenes, which serve as an input to the segmentation algorithm. **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 D).

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 normals are estimated using a local neighborhood. Because of this, a group of normals may seem to have a concave

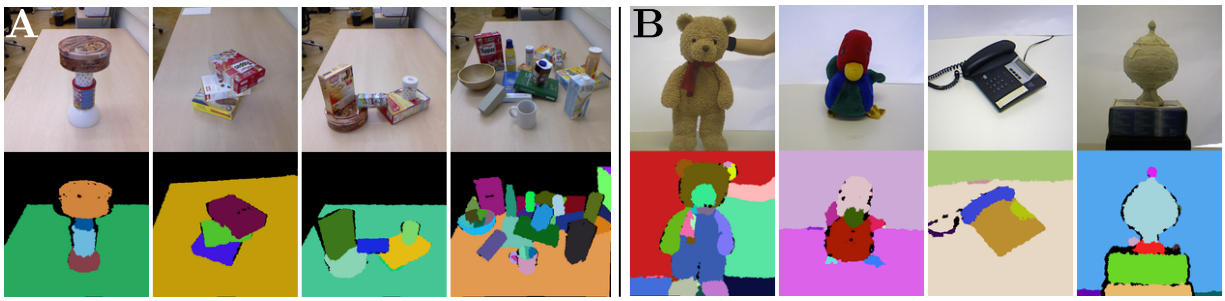


Fig. 2. Examples of segmentation: **A)** images from OSD dataset and **B)** images from our data set. Top and bottom panels show original and segmented images, respectively. Points beyond a distance of 1.3 m were cropped for visualization.

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.

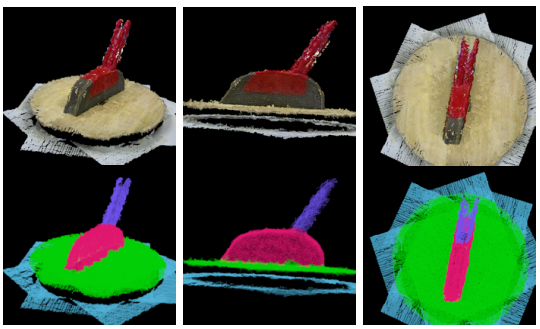


Fig. 3. Example segmentation of a point cloud combined from multiple views showing a foam hand broom. The input point cloud and segmentation result from different view points are shown in the top and bottom images respectively (also see supplementary video).

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].

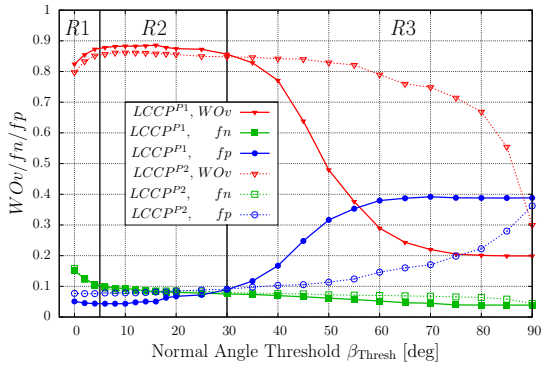


Fig. 4. Statistics obtained from segmentation of scenes from the OSD dataset using our method. Average results are shown. $LCCP^{P1}$ and $LCCP^{P2}$ stand for the different parameter sets with small (solid lines) and large (dashed lines) supervoxels respectively.

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 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%	-	-

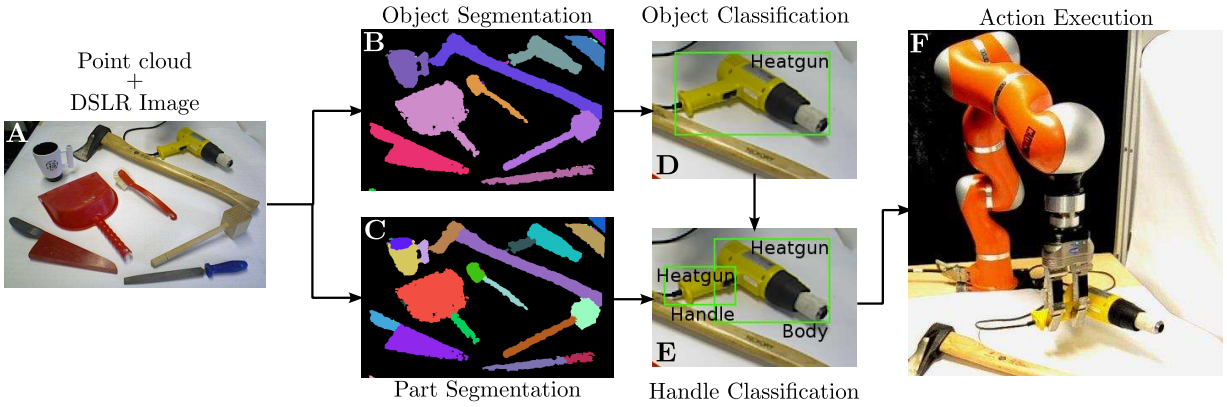


Fig. 5. Flow diagram of robotic application: **A)** Original image of the scene, **B, C)** object/part segmentation, **D, E)** object/part classification and **F)** action execution – robot grasping a heatgun by its handle. LCCP segmentation was applied after an initial ground plane subtraction. While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

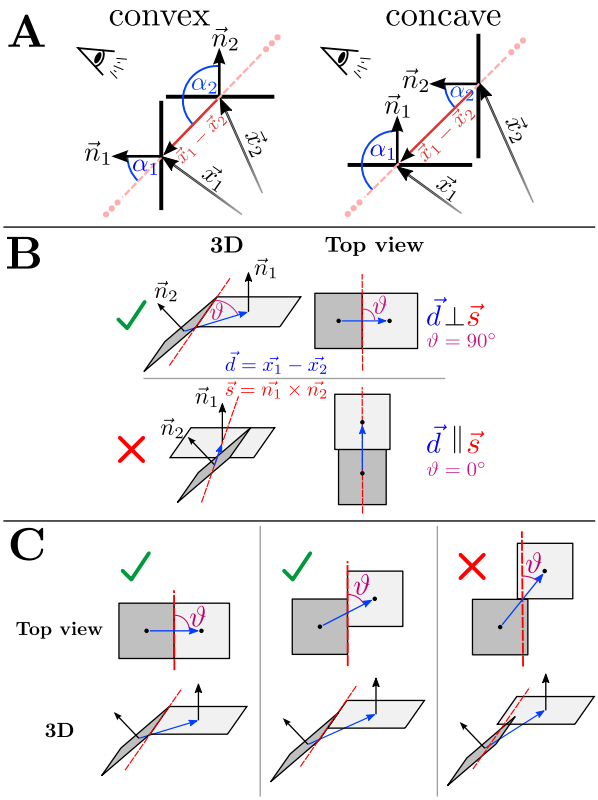


Fig. 6. **A)** Illustration of convex and concave connections between two linear patches. **B,C)** Illustration of the sanity criterion: **B)** A singular connection can be obtained by measuring the angle ϑ between the line of intersection $\vec{s}$ of the two planes represented by the patches and the 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:

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

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

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

$conv(\vec{p}_i, \vec{p}_j)$ 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.

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

[6]

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]:

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

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]:

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

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.

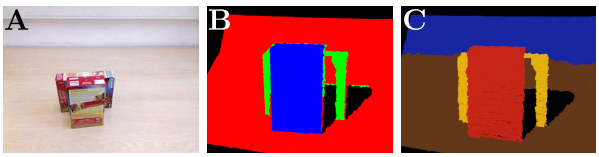


Fig. 7. **A)** Original image, **B)** ground-truth from point cloud perspective from the OSD dataset and **C)** segmentation result of our method.

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, p. submitted.

[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.

Object Partitioning using Local Convexity

Simon Christoph Stein, Markus Schoeler, Jeremie Papon and Florentin Wörgötter
Bernstein Center for Computational Neuroscience (BCCN)
III Physikalisches Institut - Biophysik, Georg-August University of Göttingen
{scstein,mschoeler,jpapon,worgott}@physik3.gwdg.de

Abstract

The problem of how to arrive at an appropriate 3D-segmentation of a scene remains difficult. While current state-of-the-art methods continue to gradually improve in benchmark performance, they also grow more and more complex, for example by incorporating chains of classifiers, which require training on large manually annotated data-sets. As an alternative to this, we present a new, efficient learning- and model-free approach for the segmentation of 3D point clouds into object parts. The algorithm begins by decomposing the scene into an adjacency-graph of surface patches based on a voxel grid. Edges in the graph are then classified as either convex or concave using a novel combination of simple criteria which operate on the local geometry of these patches. This way the graph is divided into locally convex connected subgraphs, which – with high accuracy – represent object parts. Additionally, we propose a novel depth dependent voxel grid to deal with the decreasing point-density at far distances in the point clouds. This improves segmentation, allowing the use of fixed parameters for vastly different scenes. The algorithm is straightforward to implement and requires no training data, while nevertheless producing results that are comparable to state-of-the-art methods which incorporate high-level concepts involving classification, learning and model fitting.

1. Introduction and State-of-the-Art

Segmentation of scenes into objects remains one of the most challenging topics of computer vision despite decades of research. To address this, recent methods often use hierarchies which create a rank order that build bottom-up from small localized superpixels to large-scale regions [19, 1, 3]. As an alternative, researchers have also pursued strictly top-down approaches. These began with coarse segmentations using multiscale sliding window detectors [26], later progressing to finer grained segmentations and detections based on object parts [8, 6]. These two avenues of research led naturally to methods which *combine* bottom-up hierarchy building with top-down object- and part-detectors [2, 22, 9]. While these approaches have yielded quite good results even on complex, varied data sets, they have lost much of the generality of learning-free approaches. In general the most powerful methods to-date use trained classifiers for segmentation [22, 9]. This means they cannot be applied to arbitrary unknown scenes without being re-trained, requiring the acquisition of a new data-set tailored to each test environment.

In this work we investigate model- and learning-free bottom-up segmentation of 3D point clouds captured with RGB-D cameras. In particular, we focus on the partitioning of cluttered scenes into basic *object parts* without the need for training data. As inspiration for a general rule for breaking scenes into elemental parts, we look to psychophysical studies, mostly performed on 2D images, which suggest that the transition between convex and concave image parts might be indicative of the separation between objects and/or their parts [13, 25, 21, 15, 7, 4]. While this feature has been used in machine vision to some degree [10, 17, 20, 24, 11] success has remained limited and more recent studies were forced to combine this feature with additional, often very complex feature constellations to achieve good scene partitioning [20, 24, 11]. It, thus, appears that direct transfer from 2D to 3D of the conventional, geometrically-defined convex-concave transition criterion shown in Fig. 2 A is not possible for achieving good 3D-segmentation. This puzzling observation can best be understood by ways of an example.

Fig. 1 A, left shows two cases, one of which humans will commonly consider as a single object (the left-most structure), while they report the jagged staircase on the right as consisting of three parts. The simple convex-concave criterion will usually fail in such situations because it can produce two substantial errors: On the one hand, it may bind patches across *surface singularities*, one of which is depicted in Fig. 1 A (blue box). On the other hand it may separate objects along *isolated concavities* (Fig. 1 A, red bars). There is a natural trade-off between both errors, making an appropriate segmentation of both cases generally not possi-

[7]

ble. Thus, in this study we design a novel 3D-compatible convex-concave separation criterion, which addresses these two problems in a straight-forward way and combine it with a depth dependent transform which helps to reduce the effect of noisy and sparse depth measurements from RGB-D sensors. Together this leads to a remarkably accurate partitioning of real scenes, which can compete with far more complex state-of-the-art methods as shown here by several standard benchmarks. The algorithm is simple and easy to implement and we demonstrate that the resulting segmentations can be understood in human terms as "objects" or "object-parts".

This paper is organized as follow: First, in Section 2 we present our segmentation algorithm and visualize its individual steps. In Section 3 we evaluate our method, benchmark it against other approaches and discuss the results. Finally, Section 4 will summarize our findings. The method source code is freely distributed as part of the Point Cloud Library (PCL)[1].

2. Methods

Our principal goal is to deconstruct a scene into "nameable" object parts without the need for training or classification. As psycho-physical studies suggest that the lowest-level decomposition of objects into parts is closely intertwined with 3D concave/convex relationships, we propose an algorithm, an overview of which is given in Fig. 1, that can reliably identify regions of local convexity in point cloud data.

2.1. Building the Surface-Patch Adjacency Graph

To build a surface patch adjacency graph, we use Voxel Cloud Connectivity Segmentation (VCCS) [18], a recent method which over-segments 3D point cloud data into patches, known as supervoxels (a 3D analog of superpixels). VCCS uses a local region growing variant of k-means clustering to generate individual supervoxels $\vec{p}_i = (\vec{x}_i, \vec{n}_i, N_i)$, with centroid $\vec{x}_i$, normal vector $\vec{n}_i$, and edges to adjacent supervoxels $e \in N_i$.

Supervoxels maintain adjacency relations in voxelized 3D space (specifically, 26-adjacency). The adjacency graph of supervoxels (and the underlying voxels) is maintained efficiently within the octree by searching for neighboring leaves in the voxel grid, where R_{voxel} specifies the octree leaf resolution. This adjacency graph is used for both the region growing to generate the supervoxels as well as determining adjacency of the resulting supervoxels themselves. Supervoxels are grown from a set of seed points distributed evenly in space on a grid with resolution R_{seed}. Expansion from the seed points is governed by a distance measure calculated in a feature space consisting of spatial extent, color, and normals. In this work, as we are interested only in geometric features, we only use the spatial weight $w_s = 1$ and the normal direction weight $w_n = 4$. Additionally, we have extended the VCCS algorithm by modifying how the surface normals are calculated. There are two changes: First, rather than using a simple radius-search, we use the adjacency graph to decide which nearby points are included in the normal calculation (specifically, we use the neighbors, and the neighbors-neighbors). Secondly, we use the proba-

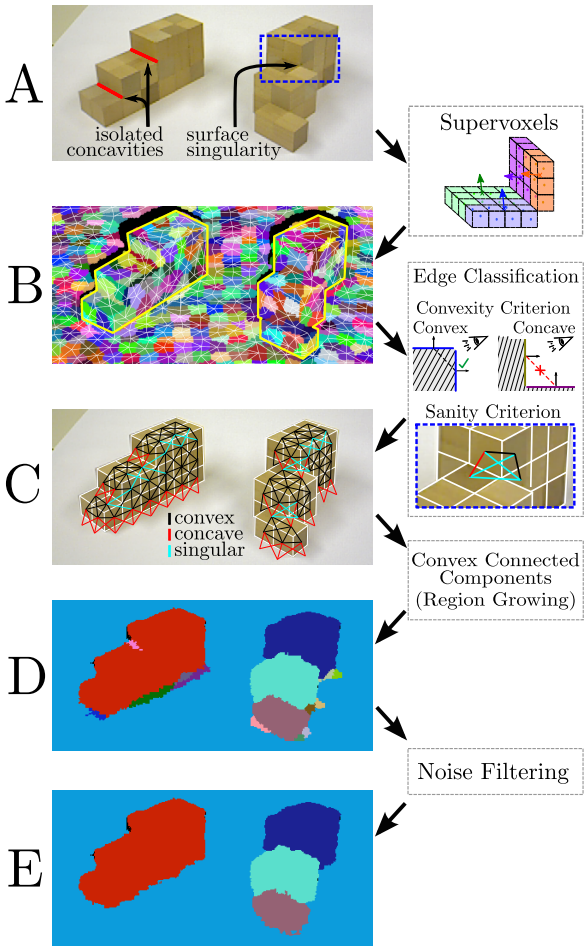


Figure 1. Flow diagram of the segmentation algorithm. **A)** RGB images corresponding to the point clouds of the scene. The red lines show two isolated concavities. The blue box shows an area with a surface singularity. **B)** Supervoxel adjacency graph. **C)** Model depicting the classified graph. Black lines denote convex connections, red lines concave ones and turquois lines singular connections (those, where two patches are connected only in a single point). **D)** Segmentation result; object labels are shown by different colors. **E)** Final result after noise filtering. The right column illustrates the supervoxel patches and the convexity and sanity criteria used for edge classification.

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

bilistic normal method of Boulch and Marlet [5]. The combination of these two changes makes normals significantly sharper, resulting in supervoxels which conform better to sharp edges.

2.2. Locally Convex Connected Patches (LCCP)

Next we segment the supervoxel adjacency graph by classifying whether the connection $e = (\vec{p}_i, \vec{p}_j)$ between two supervoxels is convex (=valid) or concave (=invalid).

Extended Convexity Criterion (CC) Consider two adjacent supervoxels with centroids at the positions $\vec{x}_1, \vec{x}_2$ and normals $\vec{n}_1, \vec{n}_2$. Whether the connection between these is convex or concave can be inferred from the relation of the surface normals to the vector joining their centroids.

The angle of the normals to the vector $\vec{d} = \vec{x}_1 - \vec{x}_2$ joining the centroids can be calculated using the identity for the dot product $\vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}| \cdot \cos(\alpha)$ with $\alpha = \measuredangle(\vec{a}, \vec{b})$. For *convex* connections, α_1 is smaller than α_2 (see Fig. 2 A). 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 these operations are commutative, thus the choice of which patch is $\vec{x}_1$, does not change the result. Also the criterion is still valid if the $\vec{x}_i$ are displaced, as long as they stay within the surface.

To compensate for 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 *concavity tolerance threshold* β_{Thresh}, concave surfaces with low curvature are seen as convex and thus merged in the segmentation. This behavior may be desired to ignore small concavities. We set:

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

where the variable CC_b defines the *basic convexity criterion*. However, local errors in the feature estimation caused by noise in the data can propagate very easily, potentially leading to errors in the resulting segmentation. This also makes the recognition of small concavities harder, as subtle features are more sensitive to noise. To improve on this we – finally – also include neighborhood information in the classification of edges: For a convex edge $e = (\vec{p}_i, \vec{p}_j)$, we require that there exists a common neighbor $\vec{p}_c$ of $\vec{p}_i$ and $\vec{p}_j$ that has a convex connection to both.

Thus we define *extended convexity* CC_e:

$$\begin{aligned} \text{CC}_e(\vec{p}_i, \vec{p}_j) = \text{CC}_b(\vec{p}_i, \vec{p}_j) \wedge \text{CC}_b(\vec{p}_i, \vec{p}_c) \\ \wedge \text{CC}_b(\vec{p}_j, \vec{p}_c) \end{aligned} \tag{2}$$

With extended convexity, more evidence is necessary for a connection to be labeled as convex. Our implementation is based on the idea proposed by Moosmann [16]. In some cases results are already satisfactory even without using this type of neighborhood information. Thus, in the results section we will always denote whether this is used or not.

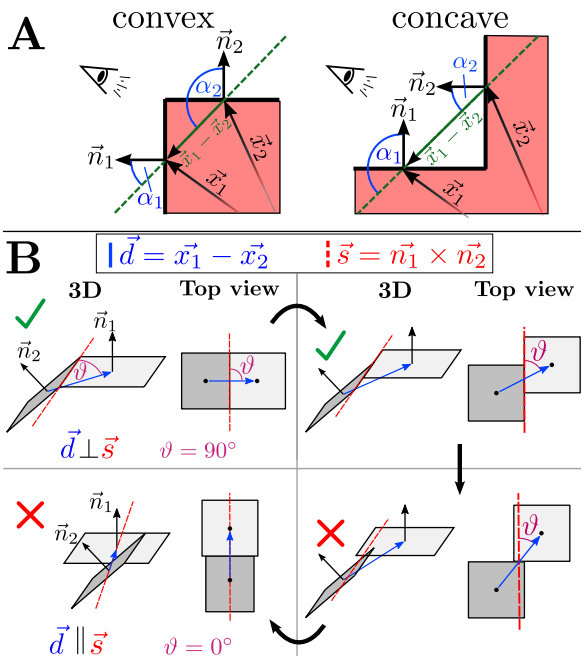


Figure 2. **A)** Illustration of measures used in the basic convexity criterion. **B)** Illustration of the sanity criterion for decreasing values of ϑ. Singular connections are characterized by small values of ϑ.

Sanity criterion (SC): As pointed out in the Introduction when discussing the problem of *surface-singularities*, in such cases, the classification of the connection of two supervoxels into convex or concave does not make sense. Hence, 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. To do this, we use the vector $\vec{d}$ which connects the centroids and the direction of intersection $\vec{s} = \vec{n}_1 \times \vec{n}_2$ of the planes approximating the supervoxel surface. As illustrated in Fig. 2 B, singular configurations can be identified by looking at the angle ϑ between $\vec{d}$ and

[7]

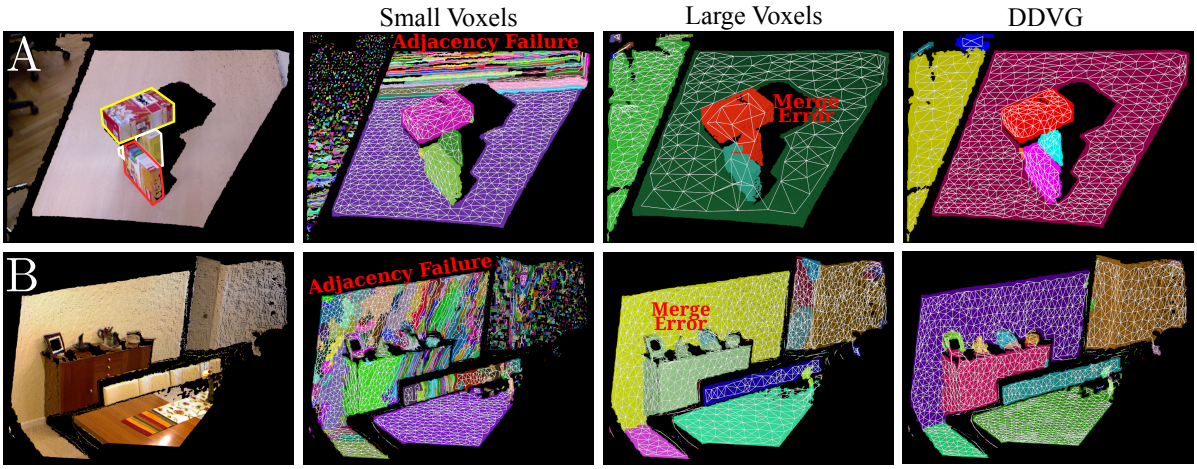


Figure 3. Two example point clouds **(A,B, left)** showing the need for the Depth Dependent Voxel Grid. For better visibility outlines have been drawn around the boxes in A. Using *Small Voxels* objects close to the camera can be segmented, but adjacency breaks down as the depth increases and the point density decreases. Using *Large Voxels* corrects the adjacency graph in the background, but leads to objects being merged in the foreground due to the coarse resolution. Using *DDVG*, the scale of the voxels gradually increases with distance from the camera – adapting to the increased noise level and lower point density – consequently adjacency is maintained and the segmentation of scenes with large depth variance is possible using fixed parameters. Note, the flat rug on the table in B does not differ enough from the table's surface and cannot be segmented by any purely depth dependent method.

$\vec{s}$. For two supervoxels sharing one side of their boundary, the two directions are orthogonal ($\vartheta = 90°$). If the directions are parallel ($\vartheta = 0°$), the patches are only connected in a single point and the situation is singular. Because the orientation of $\vec{s}$ is arbitrary, we define ϑ to be the minimum angle between the two directions, that is:

$$\begin{aligned}\vartheta(\vec{p}_1, \vec{p}_2) &= \min(\measuredangle(\vec{d}, \vec{s}), \measuredangle(\vec{d}, -\vec{s})) \\ &= \min(\measuredangle(\vec{d}, \vec{s}), 180° - \measuredangle(\vec{d}, \vec{s}))\end{aligned} \tag{3}$$

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 supervoxels 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 (a softened step function) of the angle between normals:

$$\vartheta_{\text{Thresh}}(\beta) = \vartheta^{\text{max}}_{\text{Thresh}} \cdot \left(1 + \exp\left[-a \cdot (\beta - \beta_{\text{off}})\right]\right)^{-1}, \tag{4}$$

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

The sanity criterion SC is then defined as

$$\text{SC}(\vec{p}_i, \vec{p}_j) := \begin{cases} \text{true} & \vartheta(\vec{p}_i, \vec{p}_j) > \vartheta_{\text{Thresh}}(\beta(\vec{n}_1, \vec{n}_2)) \\ \text{false} & \text{otherwise} \end{cases} \tag{5}$$

We shall denote convex edges, that is, those which satisfy both 1 and 5, as satisfying

$$\text{conv}(\vec{p}_i, \vec{p}_j) = \text{CC}_{b,e}(\vec{p}_i, \vec{p}_j) \wedge \text{SC}(\vec{p}_i, \vec{p}_j)\ . \tag{6}$$

Region Growing: The second problem discussed in the Introduction, the danger of splitting objects along *isolated concavities*, can be addressed in the next step. For this we need to find all *clusters* of supervoxels, which belong to the same subgraph of valid convex connected edges. This is accomplished using a region growing process: First, an arbitrary seed supervoxel is chosen and labeled. This label is then 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 choose a new seed supervoxel that has not been labeled and propagate the new label as before, repeating the process until all supervoxels have been labeled. Note that all our criteria are commutative, so the output of the region growing does not depend on the choice of the seeds.

Noise Filtering: Noisy surface normals (predominantly present at boundaries) can lead the convexity classification to fail and wrongfully split connected surfaces. Because these noisy patches are usually small, they can be filtered out in a post-processing step. For every segment of the segmentation, we check if it contains at least n_{filter} supervoxels. If a segment's size is smaller or equal to the filter size, we merge it with the neighboring segment with the greatest

size. The filtering continues until no segments (that have neighbors) smaller than n_{filter} are present in the image. We use $n_{\text{filter}} = 3$ for all results presented in this work.

2.3. Depth Dependent Voxel Grid (DDVG)

So far we have described the main algorithm for segmentation. Next we will introduce a generally applicable depth transform, which improves this specific analysis but can be used for all types of image analyses using algorithms with a fixed scale of observation on RGB-D data from a single RGB-D camera. In our case, we address shortcomings of the voxel grid VCCS is based on.

It is evident that observations from a single RGB-D camera have a significant drawback - the point density, and thus available detail of the scene geometry, falls rapidly with increasing distance from the camera. In addition, the levels of both quantization and noise grow quadratically [23, 12], leading to a further degradation in the quality of geometric features. This change in point density with depth creates a tradeoff between capturing small-scale detail in the foreground (using small voxels) and avoiding noise in the background (using large voxels). This is a general problem which occurs in all algorithms working with a fixed scale (for example a radius search) on point clouds created from a single view.

We propose to compensate for the loss of point density and quantization with increasing depth z by transforming the points into a skewed space using the transformation T : $(x, y, z) \rightarrow (x', y', z')$ with

$$x' = x/z, \quad y' = y/z, \quad z' = \log(z) \tag{7}$$

The division of the x and y coordinates by z reverses the perspective transformation, equalizing the point density in the x-y-plane. Transforming the z coordinate helps to deal with the effects of depth quantization by compressing points as depth increases. It is easy to show that the transformation has the following property:

$$\frac{\partial x'}{\partial x} = \frac{\partial y'}{\partial y} = \frac{\partial z'}{\partial z} = \frac{1}{z} \tag{8}$$

Because the derivatives are equal, the local coordinate frame is stretched equally along all axes by the transformations. The important thing about this property is, that small cubic voxels are still cubic after the transformation. This leaves the geometry of space basically untouched in the foreground (if the voxel size is chosen sufficiently small), while voxels in the background are skewed and grow, to compensate for reduced amount of detail available in the data.

Rather than transforming the clouds back and forth, we instead transform the bins of the octree itself, creating an octree where bin volume (and thus, voxel size) effectively increases with distance from the camera viewpoint. Doing this directly within the octree allows us to determine adjacency as before (neighboring bins), even though distance between neighboring voxels increases with distance from the camera. Fig. 3 illustrates this advantageous effect of this transformation on the segmentation.

3. Evaluation

In the following sections we present qualitative results on some sample images which demonstrate how we segment into natural (nameable) parts. In addition to these qualitative results, we also provide quantitative results which compare to state-of-the-art methods on the *NYU Indoor Dataset*[22] and *Object Segmentation Database*[20]. We compare segments against ground truth using three standard measures: *Weighted Overlap* (WOv), which is a summary measure proposed by Silberman *et al.* [22], as well as *false negative* (fn) and *false positive* (fp) scores from [24] and *over-* (F_{os}) and *under-segmentation* (F_{us}) from [20].

3.1. Object Segmentation Database (OSD)

The *Object Segmentation Database* (OSD-v0.2) was proposed by Richtsfeld *et al.*[20] in 2012. It consists of 111 cluttered scenes of objects on a table, taken with close proximity to the pictured objects. The scenes contain multiple objects, which have mostly box-like or cylindrical shape, with partial and full occlusions and heavy clutter in 2D as well as 3D. Importantly, most objects in the data set are *simple*, that is, consist of only a single part. This makes the ground-truth data relatively non-ambiguous.

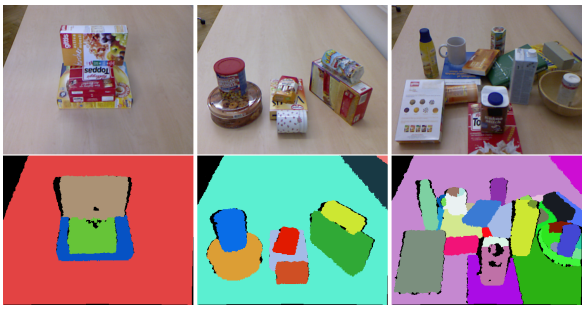


Figure 4. Example results for the OSD dataset. Points beyond a distance of 2m were cropped for visualization. Parameters: $R_{voxel} = 0.005$, $R_{seed} = 0.02$, $\beta_{\text{Thresh}} = 10°$.

The qualitative examples (Fig. 4) show that our algorithm performs very well in the segmentation of these cluttered scenes. The object separation can be intuitively understood: all objects present in the scenes are separated by concave boundaries, i.e. a line connecting neighboring surfaces of two different objects always travels through “air”. This is also true for the boundary between an object and the supporting surface. As a consequence, objects that have

Method	Learned Features	WOv	fp		fn		F_{os}	F_{us}
		Mean	Mean	SD	Mean	SD	Mean	Mean
LCCP	NO LEARNING	88.7%	4.8%	2.6%	8.3%	8.7%	7.4%	4.7%
Richtsfeld *et al.* [20]	RGB-D + Texture + Geometry	-	-	-	-	-	4.5%	7.9%
Ückermann *et al.* [24]	NO LEARNING	-	1.9%	3.3%	7.8%	7.3%	-	-

Table 1. Comparison of different segmentation methods on the OSD dataset using weighted overlap *WOv* (the higher, the better), false positives f_p, false negatives f_n, as well as over- and under-segmentation F_{os} and F_{us} (the lower, the better). LCCP results were produced with voxel resolution $R_{voxel} = 0.005$, seed resolution $R_{seed} = 0.02$ and concavity tolerance angle $\beta_{\text{Tresh}} = 10^{\circ}$.

a convex shape are correctly captured as one segment and separated from the other objects. Hollow objects (bowls, cups etc.) can be observed to show multiple segments inside, because the orientation of surface normals changes strongly on these concave surfaces. Despite most objects being simple, some objects have meaningful parts, such as handles or bottle caps, which are segmented by our algorithm.

The quantitative results (Table 1) demonstrate that our approach is able to compete with state-of-the-art methods in the task of segmenting cluttered scenes with 'single-part' objects. Compared to the learning-based method from [20] we achieve better object separation (F_{us}), but higher over-segmentation error (F_{os}). The latter is because we sometimes detect object parts (*e.g.* handles) and we do not utilize model fitting, which helps against noise. Comparing to the learning-free method of Ückermann *et al.* [24] we obtain results within a standard deviation. Note that our actual goal is to partition complex objects into parts, which is dissimilar from the other two methods.

3.2. NYU Indoor Dataset (NYU)

The *NYU Indoor Dataset* (NYUv2) from Silberman *et al.*[22] is a large and complex dataset, consisting of 1449 scenes with realistic cluttered conditions. The images are divided into a training and testing set of 795 and 654 images, respectively. In order to compare against other methods we only evaluated on the testing images, even though our method does not require a training set. The distance of objects from the camera is generally quite large in the dataset (considering the operational range of the Kinect camera), resulting in large depth quantization artifacts and few data for many objects. Furthermore, depth is often missing for significant portions of an image, due to various limitations of the Kinect sensor (e.g. reflective, transparent surfaces). To correct for this, the creators provide depth images enhanced by a hole filling algorithm (smoothdepth), which tries to estimate depth for missing areas based on the scheme from Levin *et al.*[14].

Example scenes in Fig. 5 show that the general object separation is still very good, which is expected from the results presented in the previous section. In contrast to the simple objects from the OSD dataset, however, in the NYU dataset the objects of interest are complex objects from various aspects of everyday life. As a consequence, these are mostly composed of multiple parts and thus our algorithm reveals interesting partitions. To give a few examples: In scene (A) the cupboard is partitioned into the top plate and its various drawers and the toilet is segmented into the flushing tank, the seat and the base. Scene (B) presents how a human is partitioned into hand, arm-bed, upper/lower part of the body and legs. For the sofas in scene (D) we get the two arm-rests, the back-rest and the seating. Note that in these cases the segments represent "nameable regions", which is also the case for many other segments (we discuss this further in Section 3.3). It is evident that the ground truth data will then disagree with our labeling, which results in unjustified errors. For example, in scene (A) the ground truth considers the sink to be an important part, while the drawers are not labeled individually. Despite this disagreement, we do not consider either of the labelings wrong in general, but see them as different views on the data which are in many cases equally justifiable.

Because we designed our algorithm to detect object parts defined only by geometry, very thin objects like the posters on the background wall in scene (C) are not recognizable. Also the challenging data quality sometimes leads the part partitioning to fail, as seen for the human in scene (E). To show how well the general idea of part partitioning using concave boundaries works, we present results for a complex object with high data quality in the next section.

The quantitative results (Table 2)[2] show that our algorithm is able to produce good results on the challenging dataset. Despite being much simpler and without requiring learning on human annotated ground-truth, we compete with the approach from [22] when only depth information is used. Additionally, we still achieve 93% of their score when comparing against the more complex feature spaces used in conjunction with learning-based algorithms. We should emphasize that our competitors do not aim for object parts but rather for "whole object" detections, specifically, those whole objects learned from this particular annotated ground truth. Conversely, our method establishes a general rule for object-ness that does not depend on this particular dataset, nor on the whims of a particular human annotator.

[2]Updated results for [22] are available at http://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html.

Figure 5. Example results for scenes from the NYU dataset using unsmoothed depth. Black areas indicate missing depth. Top row: rgb images. Mid. row: segmentation result. Bottom row: ground truth. Parameters A-C: $R_{voxel} = 0.0075$, $R_{seed} = 0.03$ and $\beta_{\text{Thresh}} = 8°$. Parameters D-E: $R_{voxel} = 0.01$, $R_{seed} = 0.04$ and $\beta_{\text{Thresh}} = 10°$ (identical to quantitative results, see Tab. 2).

Method	Learned Features	Depth Data	WOv
LCCP	NO LEARNING	depth	53.6%
	NO LEARNING	smoothdepth	53.8%
LCCP + ext. convexity	NO LEARNING	smoothdepth	57.6%
Silberman *et al.*[22]	RGB	-	50.3%
	Depth	both	53.7%
	RGB-D	both	60.1%
	RGB-D + Support + Structure classes	both	61.1%
Gupta *et al.*[9]	gPb-ucm Gradients (from [3])	-	55.0%
	gPb-ucm + Depth + Concavity Gradients	both	62.0%

Table 2. Comparison of different segmentation methods on the NYU dataset using weighted overlap *WOv*. LCCP results were produced with voxel size $R_{voxel} = 0.01$, seed size $R_{seed} = 0.04$ and concavity tolerance angle $\beta_{\text{Tresh}} = 10°$.

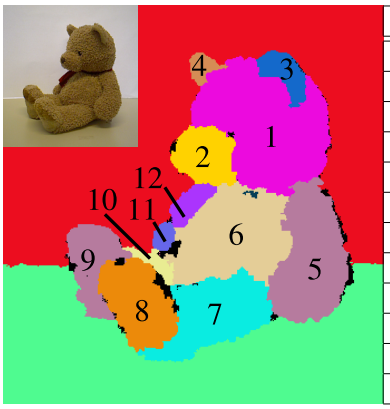
The average runtime was 470 ms per scene using an Intel Core i7 3.2 GHz processor with 95% spent computing the supervoxels. Note that supervoxels can be parallelized using GPUs, which should lead to a 10-fold speed-up.

3.3. Partitioning of Higher Quality Data

Since the NYU dataset and the OSD set do not provide close-ups on complex objects, we present a qualitative example in Fig. 6 which shows results on higher quality data. It is easy to see that our algorithm segments the image into meaningful parts, although we have not learned what constitutes a part. In particular, it is apparent that our algorithm partitions the object into parts which are easily "human-nameable". We should emphasize that the partitions in Fig. 5 A-C and in Fig. 6 were created using the proposed depth dependent voxel grid with the same, fixed parameters. This shows the applicability of our algorithm to disparate content without parameter tuning.

4. Conclusion

In this work we presented and evaluated a novel model- and learning-free bottom-up segmentation algorithm operating on 3D point clouds. Although our algorithm aims to partition parts in contrast to objects, we achieved state-of-the-art results on two well-known benchmarks, the *Object Segmentation Database* and the *NYU Indoor Dataset*. The latter being especially interesting for two reasons: First, all published results require learning (necessitating annotated ground-truth) as well as color information. Our approach, on the contrary, is the first published algorithm which requires neither. Secondly, ground-truth arbitrarily partitions the scene into "full" objects, thus segmenting objects into their parts will decrease overall scores. In spite of all this, we obtain results which are comparable to the state-of-the-art. Finally, we should emphasize that being a learning-free method has many advantages, the most important of



ID	name
1	head
2	snout
3	left ear
4	right ear
5	left arm
6	belly
7	left leg
8	left foot
9	right foot
10	right leg
11	right arm
12	tie

Figure 6. One example scene showing a teddy-bear and it's partitioning. Segments correspond to meaningful and nameable body parts.

which is that there is no need to create new training data and accompanying annotated ground truth images. Not only does this mean that the method is directly applicable as the first step in an automated bootstrapping process, it also enables use on arbitrary unknown scenes (made possible by our depth dependent grid) or scenes acquired by new devices (e.g. Kinect 2.0, or laser scanners).

Acknowledgments

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. 600578, ACAT.

References

[1] N. Ahuja and S. Todorovic. Connected segmentation tree; a joint representation of region layout and hierarchy. In *CVPR*, pages 1–8, 2008. 1

[2] P. Arbelaez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik. Semantic segmentation using regions and parts. In *CVPR*, pages 3378–3385, 2012. 1

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. PAMI*, 33(5):898–916, 2011. 1, 7

[4] M. Bertamini and J. Wagemans. Processing convexity and concavity along a 2-D contour: figure-ground, structural shape, and attention. *Psychonomic Bulletin & Review*, 20(2):191–207, 2013. 1

[5] A. Boulch and R. Marlet. Fast and robust normal estimation for point clouds with sharp features. *Computer Graphics Forum*, 31(5):1765–1774, 2012. 3

[6] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *ICCV*, pages 1365–1372, 2009. 1

[7] A. D. Cate and M. Behrmann. Perceiving parts and shapes from concave surfaces. *Attention, Perception, & Psychophysics*, 72(1):153–167, 2010. 1

[8] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. PAMI*, 32(9):1627–1645, 2010. 1

[9] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *CVPR*, pages 564–571, 2013. 1, 7

[10] R. Hoffman and A. K. Jain. Segmentation and classification of range images. *IEEE Trans. PAMI*, 9(5):608–620, 1987. 1

[11] A. Karpathy, S. Miller, and L. Fei-Fei. Object discovery in 3D scenes via shape analysis. In *ICRA*, pages 2088–2095, 2013. 1

[12] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012. 5

[13] J. J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13:321–330, 1984. 1

[14] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Trans. Graph.*, 23(3):689–694, 2004. 6

[15] T. Matsuno and M. Tomonaga. An advantage for concavities in shape perception by chimpanzees (pan troglodytes). *Behavioural Processes*, 75(3):253–258, 2007. 1

[16] F. Moosmann. *Interlacing Self-Localization, Moving Object Tracking and Mapping for 3D Range Sensors*. PhD thesis, Karlsruher Institut für Technologie (KIT), 2013. 3

[17] F. Moosmann, O. Pink, and C. Stiller. Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion. In *Intelligent Vehicles Symposium*, pages 215–220, 2009. 1

[18] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter. Voxel cloud connectivity segmentation - supervoxels for point clouds. In *CVPR*, pages 2027–2034, 2013. 2

[19] X. Ren and J. Malik. Learning a classification model for segmentation. In *ICCV*, pages 10–17 vol.1, 2003. 1

[20] A. Richtsfeld, T. Morwald, J. Prankl, M. Zillich, and M. Vincze. Segmentation of unknown objects in indoor environments. In *IROS*, pages 4791–4796, 2012. 1, 5, 6

[21] P. L. Rosin. Shape partitioning by convexity. *IEEE Trans. SMC, Part A: Systems and Humans*, 30:202–210, 2000. 1

[22] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, pages 746–760, 2012. 1, 5, 6, 7

[23] J. Smisek, M. Jancosek, and T. Pajdla. 3D with kinect. In *ICCV Workshops*, pages 1154–1160, 2011. 5

[24] A. Ückermann, R. Haschke, and H. Ritter. Real-time 3D segmentation of cluttered scenes for robot grasping. In *Humanoids*, 2012. 1, 5, 6

[25] L. M. Vaina and S. D. Zlateva. The largest convex patches: A boundary-based method for obtaining object parts. *Biol. Cybern.*, 62(3):225–236, 1990. 1

[26] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004. 1

6.3 Getting Parts

Figure 6.1: LCCP for part-segmentation showing the segmentation and the adjacency graph (top). Problem: Parts of objects are often not completely separated by concavities, which makes LCCP fail. Nevertheless some concavities do exist and hint at part-to-part boundaries. This evidence is used to induce cuts by ways of the CPC algorithm (bottom).

While object-to-object boundaries are most of the time fully concave, this assumption does not hold for part-to-part boundaries within an object (see Fig. 6.1). Still the existing fewer concavities hint at the existence of parts in an object. We use such evidence for our CPC algorithm:

[8] **Schoeler, M.** and Papon, J. and Wörgötter, F.: "Constrained Planar Cuts - Object Partitioning for Point Clouds", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. See page 99.

This is a bottom-up method for segmenting 3D point clouds into functional parts which does not require supervision and still achieves equally good results. We show that CPC is efficient to compute and generalizes well across different objects. The algorithm employs a novel locally constrained directional weighted Random Sample Consensus (RANSAC) scheme which proposes greedy cuts through the supervoxel adjacency graph. We evaluated the algorithm on recordings from RGB-D sensors as well as the Princeton Segmentation Benchmark[3], using a fixed set of parameters across all object classes. Our approach outperforms all existing bottom-up methods (reducing the gap to human performance by up to 50 %) and achieves scores similar to supervised methods.

[3] http://segeval.cs.princeton.edu/

Constrained Planar Cuts - Object Partitioning for Point Clouds

Markus Schoeler, Jeremie Papon and Florentin Wörgötter
Bernstein Center for Computational Neuroscience (BCCN)
III Physikalisches Institut - Biophysik, Georg-August University of Göttingen
{mschoeler,jpapon,worgott}@gwdg.de

Abstract

While humans can easily separate unknown objects into meaningful parts, recent segmentation methods can only achieve similar partitionings by training on human-annotated ground-truth data. Here we introduce a bottom-up method for segmenting 3D point clouds into functional parts which does not require supervision and achieves equally good results. Our method uses local concavities as an indicator for inter-part boundaries. We show that this criterion is efficient to compute and generalizes well across different object classes. The algorithm employs a novel locally constrained geometrical boundary model which proposes greedy cuts through a local concavity graph. Only planar cuts are considered and evaluated using a cost function, which rewards cuts orthogonal to concave edges. Additionally, a local clustering constraint is applied to ensure the partitioning only affects relevant locally concave regions. We evaluate our algorithm on recordings from an RGB-D camera as well as the Princeton Segmentation Benchmark, using a fixed set of parameters across all object classes. This stands in stark contrast to most reported results which require either knowing the number of parts or annotated ground-truth for learning. Our approach outperforms all existing bottom-up methods (reducing the gap to human performance by up to 50 %) and achieves scores similar to top-down data-driven approaches.

1. Introduction and State-of-the-Art

Segmentation of 3D objects into functional parts - forming a visual hierarchy - is a fundamental task in computer vision. Visual hierarchies are essential for many higher level tasks such as activity recognition [6, 12], semantic segmentation [1, 17], object detection [7], and human pose recognition [3, 16]. Nevertheless, part segmentation, particularly of 3D point clouds, remains an open area of research - as demonstrated by the inability of state-of-the-art methods to match human performance on existing benchmarks without excessive fitting to particular ground-truth training examples [5, 9, 15, 18].

In this work, we aim to partition objects from the bottom-up using a purely geometric approach that generalizes to most object types. This is in stark contrast to recent learning-based methods, which achieve good performance by training separate classifiers for each object class [9, 15]. While such methods do perform well on benchmarks, they are severely restricted in that one must know the object class a-priori, and they do not generalize to new objects at all. With unsupervised methods, such as the one presented in this work, there is no need to create new training data and annotated ground truth, allowing them to be employed as an off-the-shelf first step in object partitioning.

While many bottom-up approaches [8, 10, 13] have been tested on the Princeton Segmentation Benchmark [5], none of them are able to achieve results comparable to human segmentations. The recent learning-free approach of Zheng *et al*. [18] manages results closer to the human baseline, but only by making strong assumptions about the underlying skeleton of objects. This means that the method does not work for objects where skeletonization is uninformative, and thus does not generalize well to all object classes in the benchmark.

Psycho-physical studies [2, 4] suggest that the decomposition of objects into parts is closely intertwined with local 3D concave/convex relationships. It is readily observable that objects and object-parts tend to be isolated by concave boundaries. Stein *et al*. [14] used this idea in a bottom-up segmentation algorithm *LCCP*, which showed state-of-the-art performance in several popular object segmentation benchmarks. In that work, they make a strong assumption about local concavities, namely, that they completely isolate objects. While effective for object segmentation, this is problematic for more subtle part-segmentation where inter-part connections may not be strongly (and/or completely) concave. For instance, in Fig.1, the shoulder only has concave connections on the underside, so a strict partitioning criterion which only cuts concave edges will not separate the arm from the torso.

While it is clear that a strict partitioning will often fail

[8]

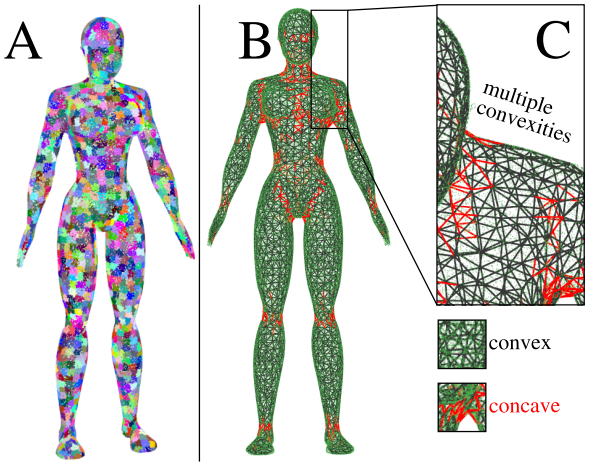


Figure 1. In complex objects parts are often only partially separated by concavities. **A)** Input object together with extracted Supervoxels. **B)** Supervoxel adjacency graph with convex/concave edge classification. **C)** Magnification of the shoulder showing how parts are not always strictly isolated by concave edges. While the underside of the shoulder is highly concave (suggesting a part boundary), the top of the shoulder is convex, so the arm cannot be separated from the torso by only cutting concave edges.

to separate parts, concave connections are nevertheless indicative of inter-part boundaries. In this work we use a relaxed cutting criterion which permits cuts of convex edges when nearby concave edges indicate a part boundary. To do this, we use local concavity information to find euclidean planar cuts which match a semi-global hierarchical concave boundary model. To find cuts which fit this model we propose a directionally weighted, locally constrained sample consensus scheme which, while being robust to noise, uses weights and penalties in a local model evaluation phase, leading to remarkably accurate partitionings of objects. We will show the first reported quantitative part-segmentation results on point-cloud data, results which outperform current state-of-the-art mesh-segmentation methods on the Princeton Object Segmentation benchmark and approach human ground truth segmentations.

This paper is organized as follow: First, in Section 2 we propose a constrained planar cutting criterion, and describe our algorithm for finding optimal cuts. In Section 3 we evaluate our method, benchmark it against other approaches and discuss the results. Finally, Section 4 will summarize our findings. The method's source code will be freely distributed as part of the Point Cloud Library (PCL)[1].

2. Methods

Our goal is to partition point clouds into their constituent objects and object parts without the need for top-down semantic knowledge (*e.g.* training or classification). As discussed earlier, local concavity is a powerful, arguably the most powerful, local feature indicative of part boundaries. In this Section we present our segmentation algorithm, which identifies regions of local concavity for a semi-global partitioning.

2.1. Local concavity evidence extraction

As a first step, we must find evidence of local concavities which hint at the existence of parts. We begin by creating a surface patch adjacency graph using Voxel Cloud Connectivity Segmentation (VCCS) [11], which over-segments a 3D point cloud into an adjacency graph of supervoxels (a 3D analog of superpixels). VCCS uses a local region growing variant of k-means clustering to generate individual supervoxels $\vec{p}_i = (\vec{x}_i, \vec{n}_i, N_i)$, with centroid $\vec{x}_i$, normal vector $\vec{n}_i$, and edges to adjacent supervoxels $e \in N_i$. Seed points for the clustering are initialized using a regular grid which samples the occupied space uniformly using an adjacency-octree structure. Clusters are expanded from the seed points, governed by a similarity measure calculated in a feature space consisting of spatial extent, color, and normal difference. In this work we ignore color, using only spatial distance ($w_s = 1$) and normal difference ($w_n = 4$) for clustering.

Once we have the supervoxel adjacency graph, we use the classification proposed for the LCCP-algorithm [14] to label edges in the graph as either convex or concave. Considering two adjacent supervoxels with centroids at $\vec{x}_1, \vec{x}_2$ and normals $\vec{n}_1, \vec{n}_2$ we treat their connection as convex if

$$\vec{n_1} \cdot \vec{d} - \vec{n_2} \cdot \vec{d} \geq 0, \tag{1}$$

with

$$\vec{d} = \frac{\vec{x_1} - \vec{x_2}}{||\vec{x_1} - \vec{x_2}||_2}. \tag{2}$$

Likewise, a connection is concave if

$$\vec{n_1} \cdot \vec{d} - \vec{n_2} \cdot \vec{d} < 0. \tag{3}$$

We use a concavity tolerance angle $\beta_{\text{thresh}} = 10°$, to ignore weak concavities and those coming from noise in the point-clouds.

2.2. Semi-global partitioning

To make use of the concavity information we will now introduce a recursive algorithm for partitioning parts which can cut convex edges as well. Beginning with the concave/convex-labeled supervoxel adjacency graph, we search for euclidean splits which maximize a scoring function. In this work we use a planar model, but other boundary models, such as constrained paraboloids are possible as well. In each level we do one cut per segment from the former level (see Fig. 2). All segments are cut independently,

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

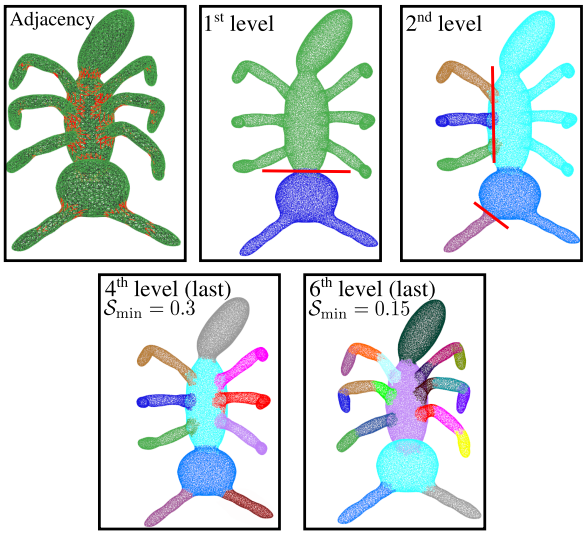


Figure 2. Recursive cutting of an object. **Top:** In each level we independently cut all segments from the former level. *Red lines*: Cuts performed in the level. **Bottom:** By changing the minumum cut score $\mathcal{S}_{\text{min}}$ we can select the desired level of granularity.

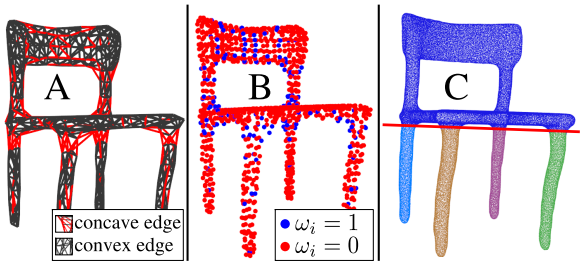


Figure 3. A chair from the Princeton Benchmark. **A:** Adjacency graph. **B:** Euclidean edge cloud extracted from the adjacency graph together with color-coded point weights ω_i. **C:** The first euclidean planar cut splits off all 4 legs, with concavities from each leg refining the cut's model.

that is, other segments are ignored. Cuts do not necessarily bi-section segments (as most graph cut methods), but as we cut in euclidean space, can split into multiple new segments with a single cut. This also allows us to use evidence from multiple scattered local concavities from different parts to induce and refine a globally optimal combined cut as shown in Fig. 3 C.

2.2.1 Euclidean edge cloud

An object shall be cut at edges connecting supervoxels. Consequently, we start by converting the adjacency graph into a *Euclidean Edge Cloud* (EEC) (see Fig. 3 B), where each point represents an edge in the adjacency graph. The point-coordinate is set to the average of the supervoxels it connects $(\vec{x}_1, \vec{x}_2)$. Additionally, the points maintain the direction of the edge $\vec{d}$ (see Eq. (2)) together with the angle α between the normals of both supervoxels $(\vec{n}_1, \vec{n}_2)$:

$$|\alpha| = \cos^{-1}(\vec{n}_2 \cdot \vec{n}_1). \tag{4}$$

We will use $\alpha < 0$ to describe convex edges and $\alpha > 0$ to denote concavities using Eqs. (1) and (3). The EEC has the advantage of efficiently storing the edge information and bridging the gap between the abstract adjacency graph representation and the euclidean boundary model.

2.2.2 Geometrically constrained partitioning

Next, we use the EEC to search for possible cuts using a geometrically-constrained partitioning model. To find the planes for cutting we introduce a locally constrained, directionally weighted sample consensus algorithm and apply it on the edge cloud as follows.

While canonical RANSAC treats points equally, here we extend it with *Weighted RANSAC*, allowing each point to have a weight. Points with high positive weights encourage RANSAC to include them in the model, whereas points with low or negative weights will penalize a model containing them. All points are used for model scoring, while only points with weights $\omega_i > 0$ are used for model estimation. We normalize the score by the number of inliers in the support region, leading to a scale-invariant scoring. With $\mathcal{P}_m$ being the set of points which lie within the support region (*i.e.* within a distance below a predefined threshold τ of the model m) and $|x|$ denoting the cardinality of set x, the score can thus be calculated using the equation:

$$\mathcal{S}_m = \frac{1}{|\mathcal{P}_m|} \sum_{i \in \mathcal{P}_m} \omega_i. \tag{5}$$

Using high weights for concave points and low or negative weights for convex points consequently leads to the models including as many concave and as few convex points as possible. In this work we use a heaviside step function $\mathcal{H}$ to tranform angles into weights:

$$\omega(\alpha) = \mathcal{H}(\alpha - \beta_{\text{thresh}}) \tag{6}$$

Please note that this will assign all convex edges a weight of zero. Still, this penalizes them in the model due to the normalization $\mathcal{P}_m$ of Eq. (5). The score for a cutting plane will therefore range between 0 (only convex points) and 1 (only concave points) in the support region.

Simply weighting the points by their concavity is not sufficient; weighted RANSAC will favor the split along as many concave boundaries as possible. Figure 4 A shows a minimalistic object with two principal concavities, which the algorithm will connect into a single cutting plane, leading to an incorrect segmentation (Fig. 4 B). To deal with

such cases, we introduce *Directional Weighted RANSAC* as follows. Let $\vec{s}_m$ denote the vector perpendicular to the surface of model m and $\vec{d}_i$ the i^{th} edge direction calculated from Eq. (2). To favor cutting edges with a plane that is orthogonal to the edge, we add a term to the scoring of concavities:

$$\mathcal{S}_m = \frac{1}{|\mathcal{P}_m|} \sum_{i \in \mathcal{P}_m} \omega_i t_i \tag{7}$$

$$t_i = \begin{cases} |\vec{d}_i \cdot \vec{s}_m| & i \text{ is concave} \\ 1 & i \text{ is convex.} \end{cases} \tag{8}$$

The notation $\cdot$ refers to the dot-product and $|x|$ to cardinality or absolute value. The idea behind Eq. (8) is that convexities should always penalize regardless of orientation, whereas concavities hint at a direction for the cutting. The effect on the partitioning is shown in Fig. 4 C. Due to perpendicular vectors $|\vec{s}_1 \cdot \vec{d}_1|$ and $|\vec{s}_1 \cdot \vec{d}_2|$ the directional concavity weights for the cut in B are almost decreased to zero.

2.2.3 Locally constrained cutting

The last step of the algorithm introduces *locally constrained cutting*. While our algorithm can use concavities separating several parts as shown in Fig. 3 C, this sometimes leads to cases where regions with strong concavities induce a global cut which will split off a convex part of the object (an example is shown in Fig. 5 B). To prevent this kind of over-segmentation we constrain our cuts to regions around local concavities as follows. Given the set of edge-points $\mathcal{P}_m$ located within the support region of a model, we start with a euclidean clustering of all edge-points using a cluster threshold equal to the seed-size of the supervoxels. Using $\mathcal{P}_m^n \subset \mathcal{P}_m$ to denote the set of points in the n^{th} cluster, we modify Eq. (7) to operate on the local clusters instead of $\mathcal{P}_m$:

$$\mathcal{S}_m^n = \frac{1}{|\mathcal{P}_m^n|} \sum_{i \in \mathcal{P}_m^n} \omega_i t_i. \tag{9}$$

As this operation is too expensive to be employed at each model evaluation step of the RANSAC algorithm, we only apply it to the highest scoring model. Only edges with a cluster-score $\mathcal{S}_m^n \geq \mathcal{S}_{\text{min}}$ will be cut.

This whole cutting procedure is repeated recursively on the newly generated segments and terminates if no cuts can be found which exceed the minimum score $\mathcal{S}_{\text{min}}$ or if the segment consists of less than N_{min} supervoxels.

3. Evaluation

In this section we will describe the experimental evaluation and analysis of our proposed method.

3.1. Data sets

We evaluate our algorithm quantitatively on the Princeton Object Segmentation Benchmark [5], and qualitatively on the benchmark as well as on Kinect for Windows V2 recordings. The benchmark consists of 380 objects in 19 categories together with multiple face-based ground-truth segmentations (*i.e.* each face in the object has a ground-truth label). In order to use a mesh annotated ground-truth to benchmark, we first create point clouds using an equi-density random point sampling on the faces of each object, and then calculate normals using the first three vertices of each face. To evaluate our segmentations, we determine the dominant segment label in the point ensemble for each face and map that label back to the face of the polygonal model.

3.2. Quantitative results

We compare to the mesh-segmentation results reported in [5, 9, 18] as well as to results from LCCP[14] (with extended convexity and the sanity criteria) using the standard four measures: *Cut Discrepancy*, *Hamming Distance*, *Rand Index* and *Consistency Error*.

Cut Discrepancy, being a boundary-based method, sums the distance from points along the cuts in the computed segmentation to the closest cuts in the ground truth segmentation, and vice-versa.

Hamming Distance ($\mathcal{H}$) measures the overall region-based difference between two segmentations A and B by finding the best corresponding segment in A for each segment in B and summing up the differences. Depending on if B or A is the ground-truth segmentation this yields the missing rate $\mathcal{H}_m$ or false alarm rate $\mathcal{H}_f$, respectively. $\mathcal{H}$ is defined as the average of the two rates.

Rand Index measures the likelihood that a pair of faces have either the same label in two segmentations or different labels in both segmentations. To be consistent with the other dissimilarity-based metrics and other reported results we will use $1 -$ Rand Index.

The fourth metric, *Consistency Error*, tries to account for different hierarchical granularities in the segmentation both globally (Global Consistency Error GCE) as well as locally (LCE). For further information on these metrics we refer the reader to [5].

Unlike most methods benchmarked on the Princeton Dataset our method does not need the number of expected segments as an input, allowing us to run the complete benchmark with a fixed set of parameters: $\mathcal{S}_{\text{min}} = 0.16$, $N_{\text{min}} = 500$ (see Fig. 6). For the supervoxels we use a seed resolution of $R_{\text{seed}} = 0.03$ and a voxel resolution $R_{\text{voxel}} = 0.0075$. To remove small noisy segments, we also merge segments to their largest neighbor if they are smaller than 40 supervoxels. The same settings were used for LCCP, too. We denoted the degree of supervision required for the algorithms using color codes (green: unsupervised

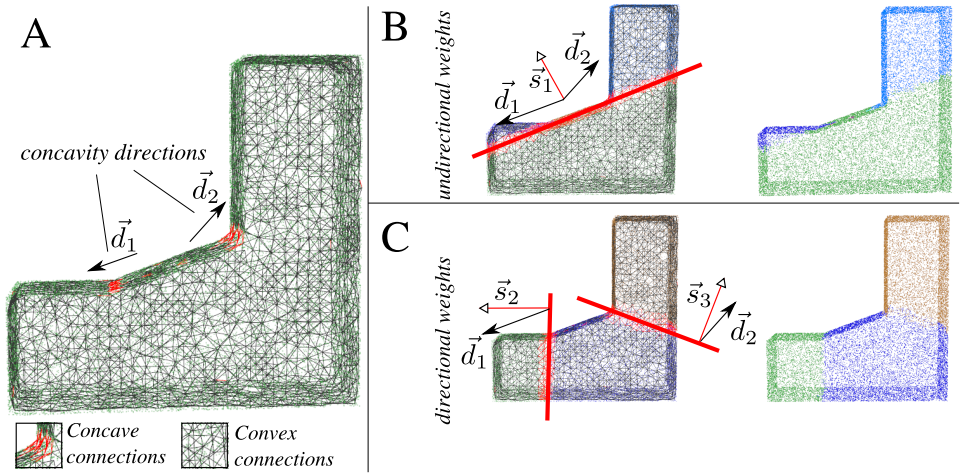


Figure 4. The highest scoring splits for undirectional and directional weights. **A)** Input object and adjacency graph. **B)** Using undirectional weights the best cut matches all concavities. However, this cut gets a lower score with directional weights due to the factors $|\vec{s}_1 \cdot \vec{d}_1|$ and $|\vec{s}_1 \cdot \vec{d}_2|$. **C)** The partition when using directional weights.

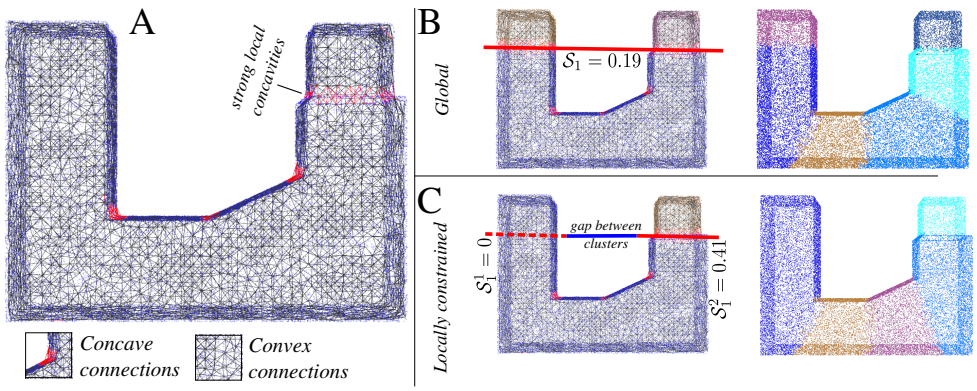


Figure 5. Comparison between locally constrained and global cuts. **A)** Input object and adjacency graph. **B)** Due to the strong local concavities on the right the algorithm will cut trough a perfectly convex part of the object (left). **C)** Using locally constrained cuts will find two clusters (along the dashed and solid red lines). Evaluating both clusters separately will only cut the right side. All cuts used directional weights.

orange: weakly supervised and red: supervised/learning). Unsupervised methods (such as ours) do not take model specific parameters into account and use fixed parameters for the full benchmark. Weakly supervised methods need to know the number of segments. Supervised algorithms need objects from the ground-truth of each category for training, using a different classifier for every class. Despite the fact that we need to convert the mesh information to point clouds and vice-versa, our method achieves better than state-of-the-art results on all measures. For Consistency Error and Rand Index we are able to reduce the gap for unsupervised and weakly-supervised methods to human performance by 50 %. Comparing the speed of our method to other methods, Table 1 shows that our method is competitive in terms of complexity, too. Please note that we measured time on a single 3.2 GHz core whereas the other methods have been benchmarked by [5] on a 2 GHz CPU. Still, this allows us to estimate that our method is faster than Randomized Cuts and Normalized Cuts and about as complex as Core Extraction and Shape Diameters, while being superior in performance to all.

3.3. Qualitative results

Example segmentations from the Princeton benchmark as well as Kinect for Windows V2 recordings from http://www.kscan3d.com are depicted in Figs. 7 and 8. We should emphasize that our algorithm does not require full scans of objects, that is, it can be applied to single views

[8]

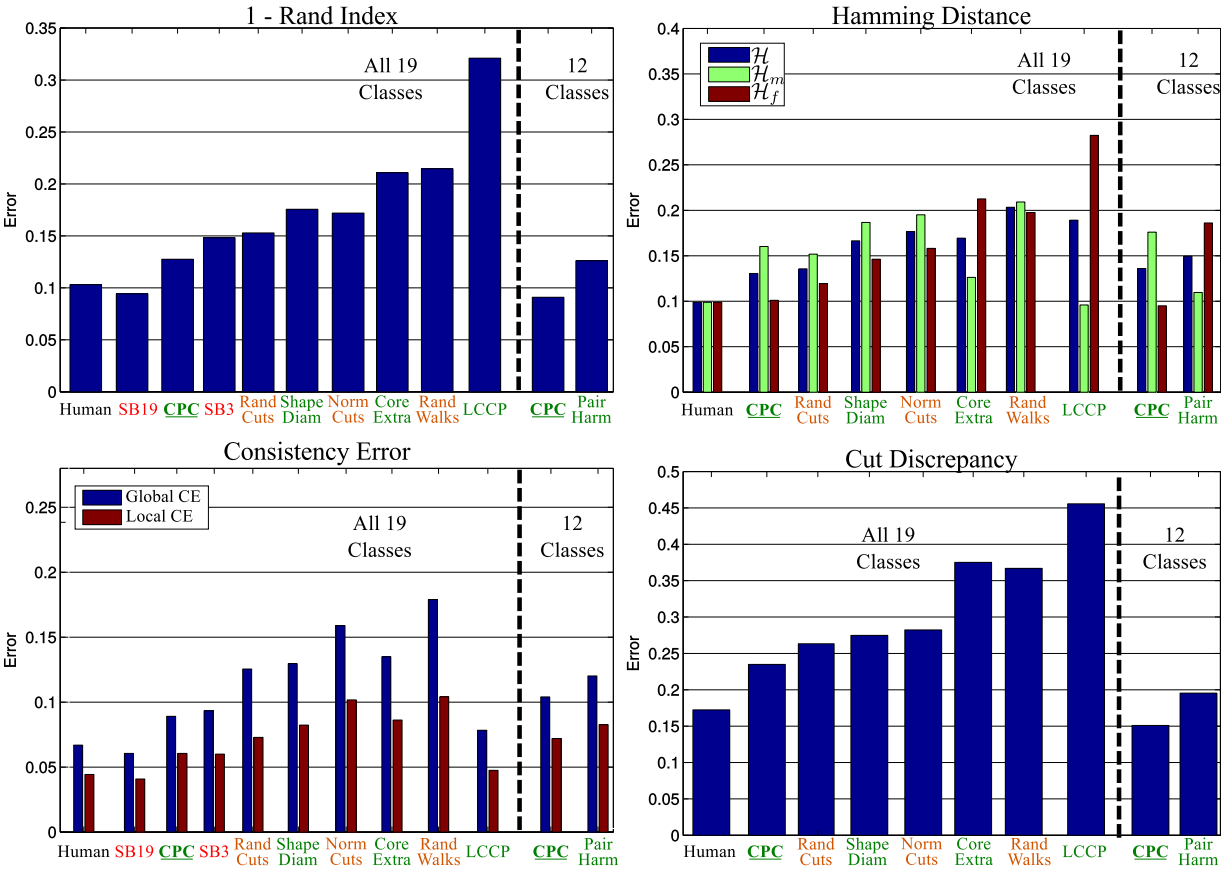


Figure 6. Comparison of proposed *CPC* algorithm to results published on the Princeton benchmark. Green algorithms are unsupervised, orange algorithms are weakly-supervised and red denotes supervised (*i.e.* training). For SB3 and SB19 [9] results on some error measures had not been published. As Zheng *et al.* [18] (PairHarm) did not report results on the full benchmark, we show results on their subset to the right of the dashed line. All objects have been segmented with local constrains and directional weights using fixed parameters.

as shown in Fig. 8 E. Additionally, we show the robustness of the proposed algorithm against shape variations as well as added noise in Fig. 9 using the "plier" class from the Princeton benchmark.

Method	Avg. Comp. Time	Rand Index
Human	-	0.103
CPC	13.9	0.128
Randomized Cuts	83.8	0.152
Normalized Cuts	49.4	0.172
Shape Diameter	8.9	0.175
Core Extraction	19.5	0.210
Random Walks	1.4	0.214
LCCP	0.47	0.321

Table 1. Comparison of averaged computational time in seconds per object for the different learning-free algorithms.

4. Conclusion

In this work we introduced and evaluated a novel model- and learning-free bottom-up part segmentation algorithm operating on 3D point clouds. Compared to most existing cutting methods it uses geometrically induced cuts rather than graph cuts, which allows us to generalize from local concavities to geometrical part-to-part boundaries. For this we introduced a novel RANSAC algorithm named *Locally Constrained Directionally Weighted RANSAC* and applied it on the edge cloud extracted from the Supervoxel Adjacency Graph. We were able to achieve better than state-of-the-art results compared to all published results from unsupervised or weakly-supervised methods and even compete with some data-driven supervised methods (note, SB19 needs 95% of the objects for training). For Consistency Error and Rand Index we are able to reduce the gap to human performance by 50 %. We also introduced a protocol to adapt mesh-segmentation benchmarks to point clouds us-

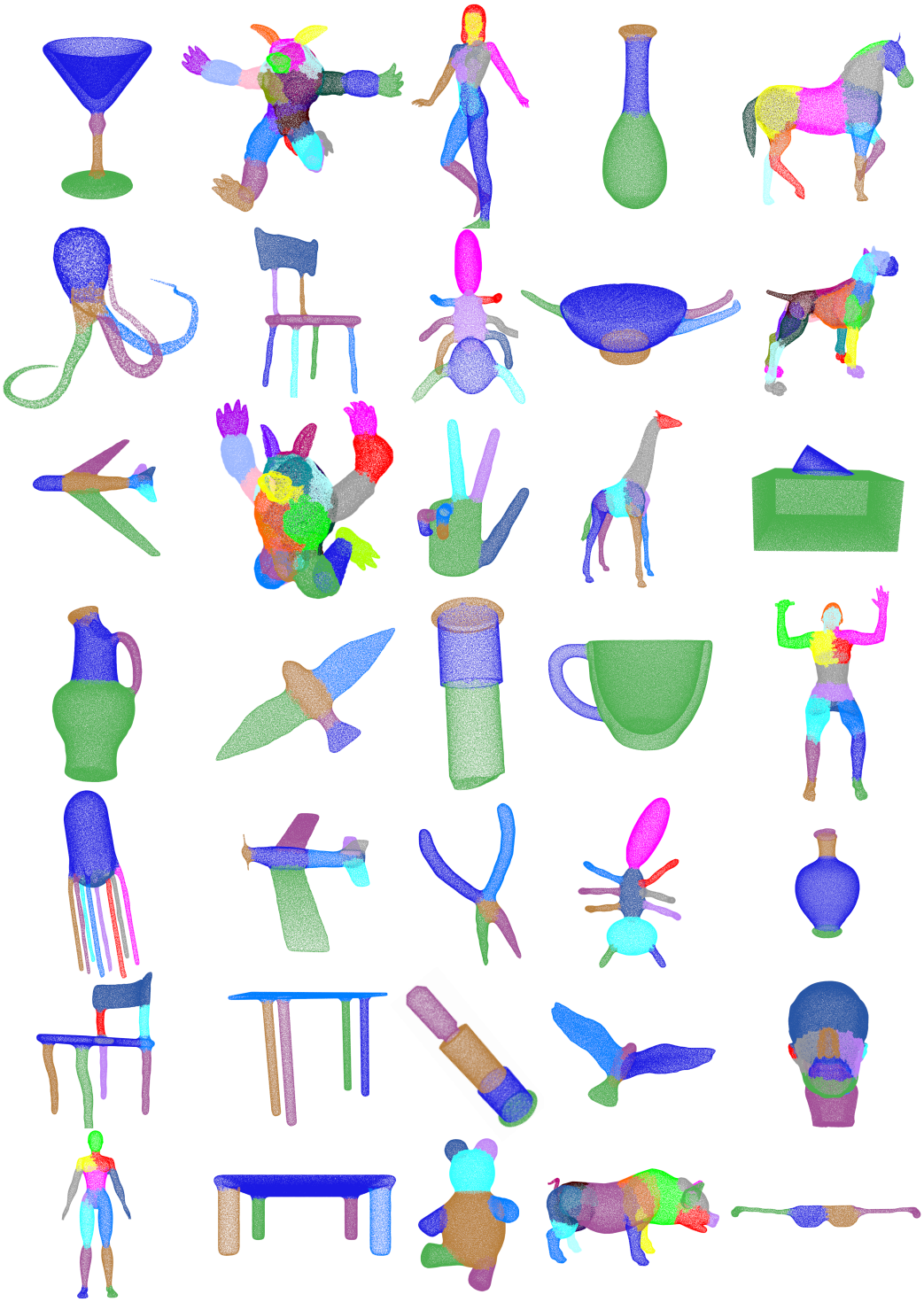


Figure 7. Qualitative results on the Princeton benchmark. All objects have been segmented with proposed algorithm using a single set of parameters ($\mathcal{S}_{\text{min}} = 0.16$, $N_{\text{min}} = 500$).

[8]

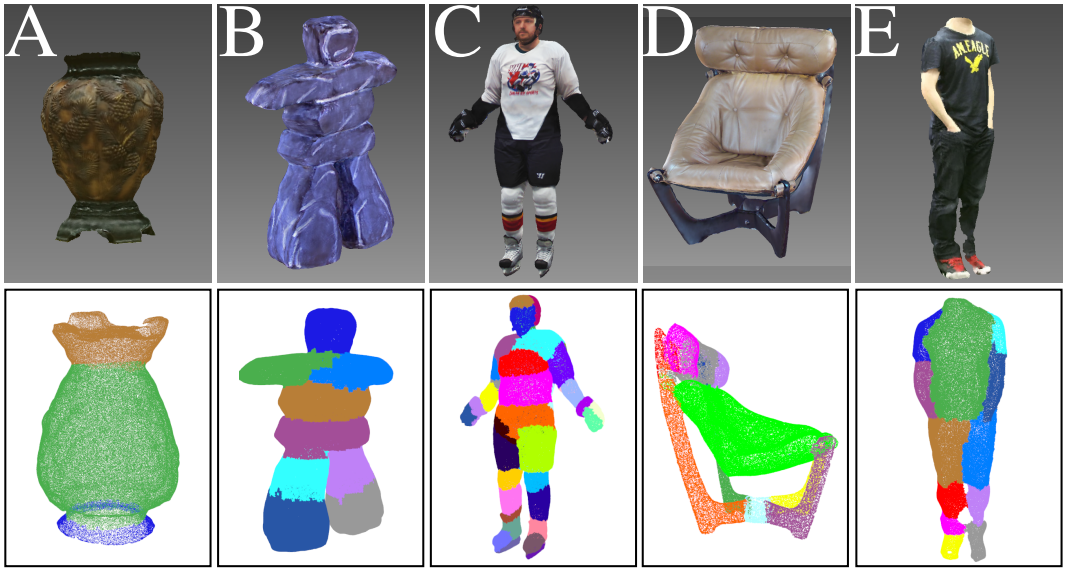


Figure 8. Qualitative results for the Kinect for Windows V2 gallery recordings from http://www.kscan3d.com using proposed algorithm. **A-D:** Full recordings. **E:** Single view recording.

ing an equi-density randomized point sampling, and a back-propagation of found labels to the mesh. This allowed us to report the first quantitative results on part-segmentation for point clouds.

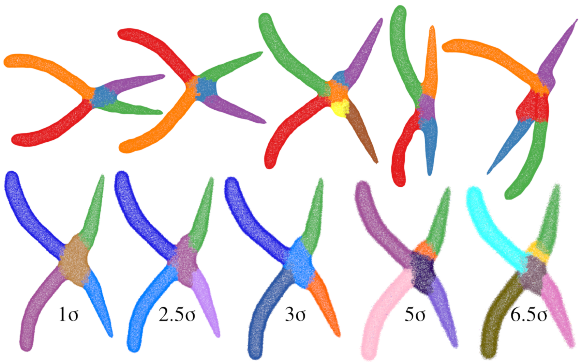


Figure 9. Robustness of the segmentation against shape variations (top) and increasing noise level (bottom).

Finally, we should emphasize that our method is learning-free, which has many advantages. Most importantly, there is no need to create new training data and annotated ground truth for new objects. Additionally, learning-based methods need to know the class of an object before they can be used for segmentation, since they must select which partitioning model to use. Our method, on the other hand, can be used directly on new data without any such limitations. Despite being a purely data-driven method our algorithm can cope with shape variations and added Gaussian noise resulting in consistent segmentations. This means that the method is directly applicable as the first step in an automated bootstrapping process and can segment arbitrary unknown objects.

Acknowledgments

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. 600578, ACAT.

References

[1] P. Arbelaez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik. Semantic segmentation using regions and parts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3378–3385, 2012. 1

[2] M. Bertamini and J. Wagemans. Processing convexity and concavity along a 2-D contour: figure-ground, structural shape, and attention. *Psychonomic Bulletin & Review*, 20(2):191–207, 2013. 1

[3] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *ICCV*, pages 1365–1372, 2009. 1

[4] A. D. Cate and M. Behrmann. Perceiving parts and shapes from concave surfaces. *Attention, Perception, & Psychophysics*, 72(1):153–167, 2010. 1

[5] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3d mesh segmentation. In *ACM Transactions on Graphics (TOG)*, volume 28, page 73. ACM, 2009. 1, 4, 5

[6] C. Desai, D. Ramanan, and C. Fowlkes. Discriminative models for static human-object interactions. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 9–16, 2010. 1

[7] A. Farhadi and M. A. Sadeghi. Phrasal recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2854–2865, Dec. 2013. 1

[8] A. Golovinskiy and T. Funkhouser. Randomized cuts for 3D mesh analysis. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA)*, 27(5), Dec. 2008. 1

[9] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3d mesh segmentation and labeling. *ACM Transactions on Graphics*, 29(4):1, July 2010. 1, 4, 6

[10] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin. Fast mesh segmentation using random walks. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 183–191. ACM, 2008. 1

[11] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter. Voxel cloud connectivity segmentation - supervoxels for point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2027–2034, 2013. 2

[12] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2847–2854. IEEE, 2012. 1

[13] L. Shapira, A. Shamir, and D. Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249–259, Apr. 2008. 1

[14] S. Stein, M. Schoeler, J. Papon, and F. Wrgtter. Object partitioning using local convexity. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 1, 2, 4

[15] W. Xu, Z. Shi, M. Xu, K. Zhou, J. Wang, B. Zhou, J. Wang, and Z. Yuan. Transductive 3d shape segmentation using sparse reconstruction. *Computer Graphics Forum*, 33(5):107–115, Aug. 2014. 1

[16] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1385–1392, 2011. 1

[17] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17–24, 2010. 1

[18] Y. Zheng, C.-L. Tai, E. Zhang, and P. Xu. Pairwise harmonics for shape analysis. *IEEE Transactions on Visualization and Computer Graphics*, 19(7):1172–1184, July 2013. 1, 4, 6

[8]

6.4 Getting Functionality

At this point we are able to segment a scene into objects (using LCCP, Section 6.2) and those objects into parts (using CPC, Section 6.3). This allows us, now, to introduce an algorithm which aims at assigning *function* to novel objects and their parts. There are two main advantages emanating from this part-perspective: First, we can describe/recognize parts of objects which is one prerequisite for robotic manipulations (*e.g.*, which part is the tool-end and which is the handle). Second, it reduces the intra-class variance, because we do not deal with the visual variation of full objects in a class (which is huge), but instead with the much smaller visual variation of simpler parts.

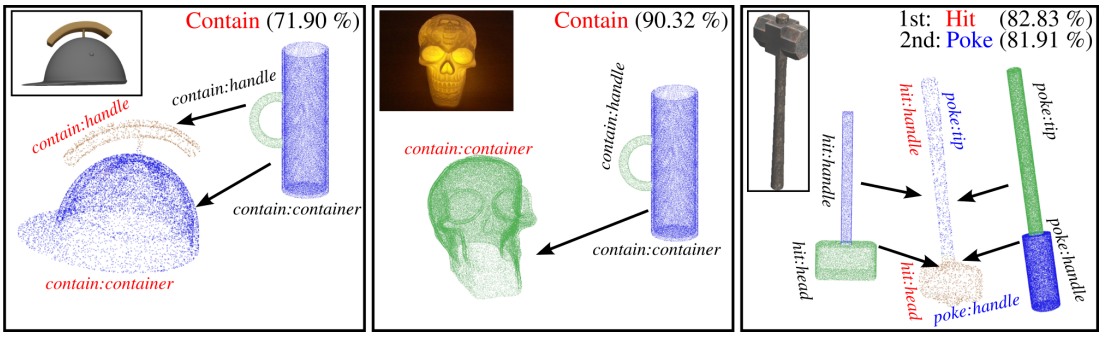


Figure 6.2: Showing our OFA system, proposed in [9], assigning *function* to three objects (helmet, skull, and hammer). The red labels show the recognized *functions* for the full-object (*e.g.*, Cotain, Hit, Poke, ...) and the part-function (*e.g.*, contain:handle). The objects with black part-labels have been found to be the most similar objects from the training set. The hammer on the right shows the case of two *functions* scoring equally high (red and blue labels). According to our algorithm, the hammer could be used to hit something, or it could be used to poke with the handle. Please note how the part-function "handle" gets assigned to a different part when changing the object *function* from "hit" to "poke".

A system for OFA should, therefore, have the following properties:

1. It should be able to assign objects and parts *functionality*.
2. It should generalize between objects with different number of parts. For example, a cup with 3 handles and a barrel are both "fillable".
3. It should be able to assign multiple *functions* to a single object. Two examples: Multi-tools are specifically designed to possess multiple *functions* and objects can often be used as a makeshift replacement. This again leads to parts being labeled differently depending on the overall *functionality* of the object.

We propose such a system in the first part of our paper:

[9] **Schoeler, M.** and Wörgötter, F.: "Bootstrapping the Semantics of Tools: Affordance analysis of real world objects on a per-part basis," *IEEE Transactions on Autonomous Mental Development (TAMD)*, 2015 (in press). See page 111.

In principal the algorithm compares parts of a new object to parts of known objects in the training set. We do this by describing objects as graphs; treating parts as annotated nodes and the way parts are attached to their neighbors as annotated edges. To be able to compare objects, we check all possible association between the parts from one object to parts in another object. We specifically allow multiple parts of one object to be assigned to a single part (*e.g.*, six handles of a cup can be associated to one handle of another mug). This allows for dealing with objects with a variable number of parts (see property 2).

To make multiple *function* assignments to one object possible (see property 3), the algorithm calculates compatibility scores for each function for an object. Together with the compatibility score we also associate the object-parts with functions. Motivated by the idea that first primitive tools in human development were deduced from human hand shapes, we consider the 6 primitive *functionalities*: *Contain*, *Cut*, *Hit*, *Hook*, *Poke*, and *Sieve* (see [9] Fig. 1). We showed that our system is able to assign the correct *functionality* to a wide range of objects. For example, the hollow skull shown in the middle of Fig. 6.2 is completely novel to our system, still it is recognized as a "fillable" object (the most similar object is a beer glass).

Additionally, we introduce an ontology, in the second part of the paper, which - based on hand-shapes - regards the geometry of the interacting tool-end and the relative motion of the tool in order to make predictions about *functionality*. More specifically, we showed that examining the geometry of the tool-end (*i.e.*, convex vs. concave and its aspect ratio) in addition to the relative pose towards the surface and the tool-motion (see [9] Fig. 13) leads to about 30 primitive tool-functionalities (see [9] Fig. 14), which could easily be stored and remembered by artificial agents.

Bootstrapping the Semantics of Tools: Affordance analysis of real world objects on a per-part basis

Markus Schoeler, Florentin Wörgötter

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

***Index Terms*—Tool bootstrapping, Object recognition, Function analysis, Tool ontology**

I. Introduction

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

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

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

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

The current study, though, adopts this hypothesis and we now introduce a computer vision based system that makes predictions about the possible functions of simple tools. This system uses "labeled data" for training. One central claim, related to our above speculations, is that there are only very few basic tool functions existing which are related to the shape and function of the human hand. Hence, understanding your hand-function provides you with all the labels for training the system "free of additional charge" (Fig. 1). This argument is much strengthened in Section IX where we show how to

M. Schoeler and F. Wörgötter are with the Third Physical Institute, University of Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany, e-mail: {mschoeler, worgott}@gwdg.de.

[1]E.g., a fist can be used as a hammer, a single finger as a borer, a flat hand as a paddle.

[2]By this the hand can lock better to the wrist, allowing for larger pressure to be applied to wrist and hand while grasping.

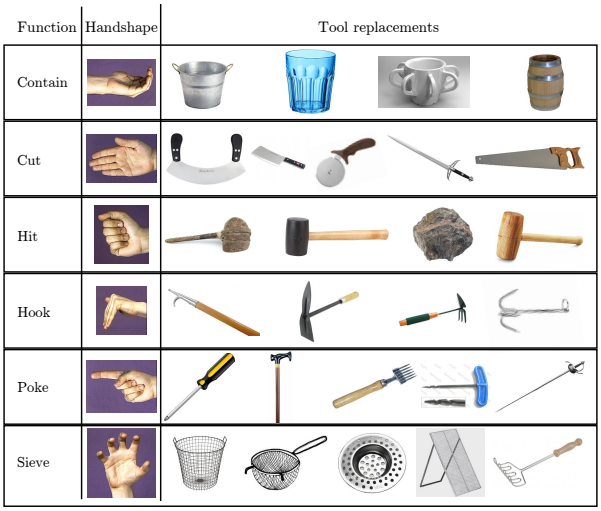


Fig. 1. Primitive functions, their associated hand shapes and tools meant to substitute and improve efficiency.

structure such a system ontologically, thereby demonstrating its rather low complexity, which can well facilitate the here-suggested bootstrapping of tools by hands.

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

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

II. Overview

The core idea of this work is the assumption that functional objects should be described by their parts and part relations. This holds for tools, which are mostly considered in this study, but also for most, if not all, other objects. For example an object for cutting may consist of a blade and a handle, an object for hitting has a handle and a head most of the time. But objects for the same function can have different number of parts. Moreover the way parts are attached to each other plays an important role for an object's functionality. The objects in the lower part of Fig. 7 consist of the same parts, but can be used differently or not at all because of their part-to-part relations. This leads to huge intra-class variance in case of functional categories, which, as we show in this paper, can be faced by analyzing objects on the part level. Thus, our algorithm uses features of the different parts, but also their geometrical configurations relative to each other ("relative pose") to define an object.

Figure 2 shows all algorithmic components. It looks complex, but there are only 3 blocks existing: (A) Preprocessing (yellow), (B) Object Signature Extraction (red) and (C) Object Similarity Calculation (green). In the middle we show the Object Signatures being the output of block B and the input to block C. Small section numbers refer to the sections where the different algorithmic components are described in this article.

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

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

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

III. Related Work

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

A. Biological Background

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

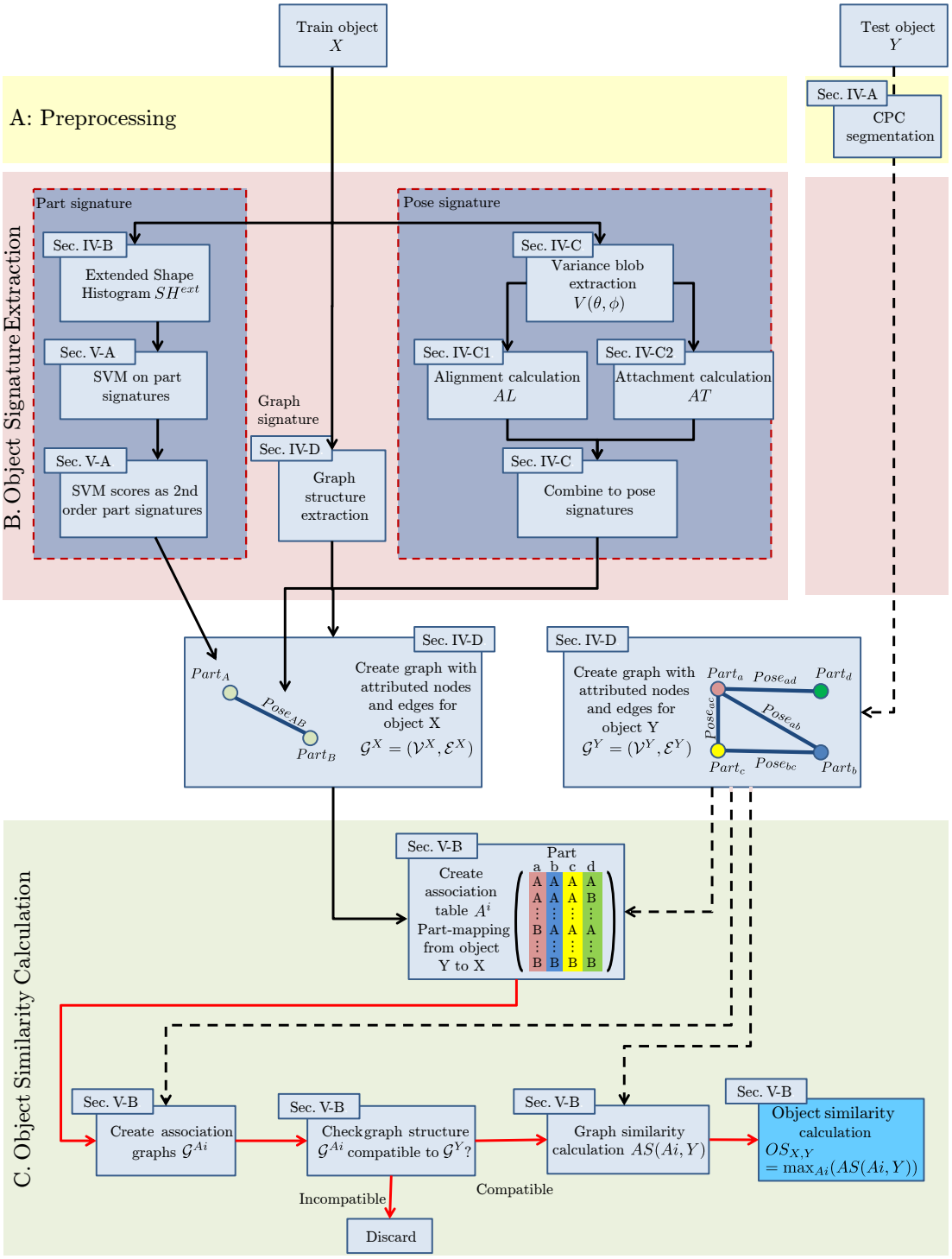


Fig. 2. Algorithm overview of the function analyzer.

which does not hold for geons (they can be named but these names carry no functional meaning).

The importance of parts for the visual system is additionally supported by the experiments of Biederman [7] which show that objects can be identified as long a single parts of an object can be recognized. As drivers for the part segmentation

[9]

in a human brain Richards and Hoffman [8], [9] indicated concavities (cusps) between parts, which is the mechanism that we also use in the CPC algorithm to obtain the parts for this study.

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

B. Computer vision

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

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

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

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

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

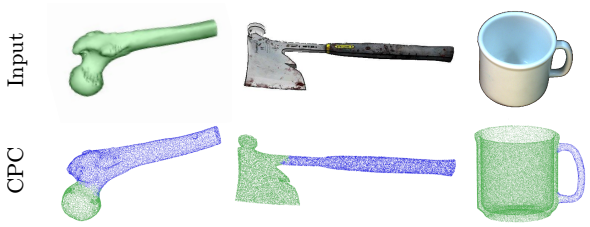


Fig. 3. The *CPC* algorithm applied to three different functional objects. Left: Femur model from shapes.aimatshape.net; Middle: Hatchet model from http://tf3dm.com; Right: WhiteCup scan from the KIT ObjectModels Web Database.

of attachment and relative pose). We capture this information in our pose-signature and show that it is important to recognize function of tools (see Table. I).

In *Attribute based approaches* [33] classes are expressed as a mixture of human-specified high-level descriptions, allowing the classifier to be applied to new classes with known attributes. Our approach, too, can deal with novel object classes like a saw when other objects for cutting are known. In contrast to [33], we do not need to provide attributes, but summarize all objects allowing for certain functions into super-categories. Combined with the fact that we can assign multiple functions to objects, we also pave the way to makeshift tool replacements. This is an interesting concept as it allows robots for instance to bootstrap alternative solutions to problems.

IV. Object Description at the Part Level

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

A. Part Segmentation using the CPC algorithm

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

B. Part Signatures

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

[3]https://github.com/mschoeler/pcl/archive/shape_generator.zip

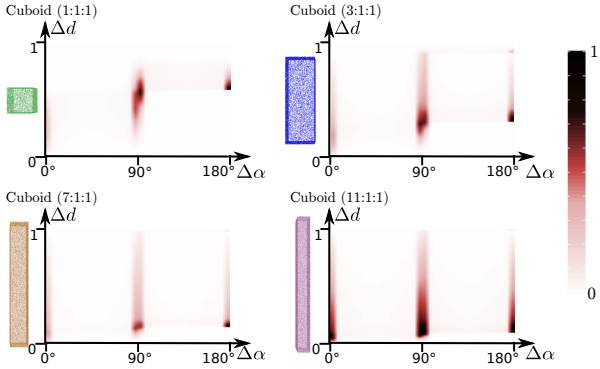


Fig. 4. Aspect Ratio dependency of the basic shape histogram SH for a cuboid with $N_d = N_\alpha = 80$.

For signature generation, in this work we investigate *Signature of Histograms of OrienTations* (SHOT) features [35], *Ensemble of Shape Functions* (ESF) features [36] as well as an extension of the two dimensional shape histogram proposed by Mustafa et al. [37]. Both SHOT and ESF features are geometry based and have shown state-of-the-art performance in object classification tasks [38]. All features use 3D information as their input.

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

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

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

[9]

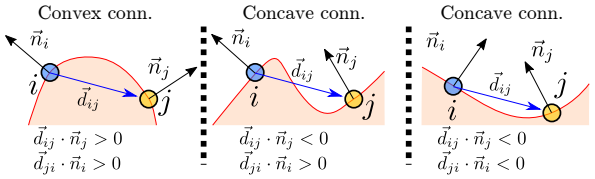


Fig. 5. Showing an example of convex vs. concave connectable point pairs introduced in Eq. (1). The red lines denote a potential surface connecting both points. $\vec{d}_{ji} = -\vec{d}_{ij}$.

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

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

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

C. Pose Signature

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

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

Whenever we write "pose signature" it means both properties.

1) Alignment AL: We define the alignment number between two parts A and B, AL_{AB}, as a scalar in the range of 0 to 1. It should change as soon as a part is rotated in respect to the other one. Still, to preserve overall rotational invariance, AL is not allowed to change, if identical transformations are applied to both parts. The more the parts are in-line, the bigger AL should be. One obvious solution to this would be to calculate each part's axis of elongation (for example by applying Principal Component Analysis (PCA) on the part's point cloud and using the first principal component axis). The angle between these axes could than be used to calculate

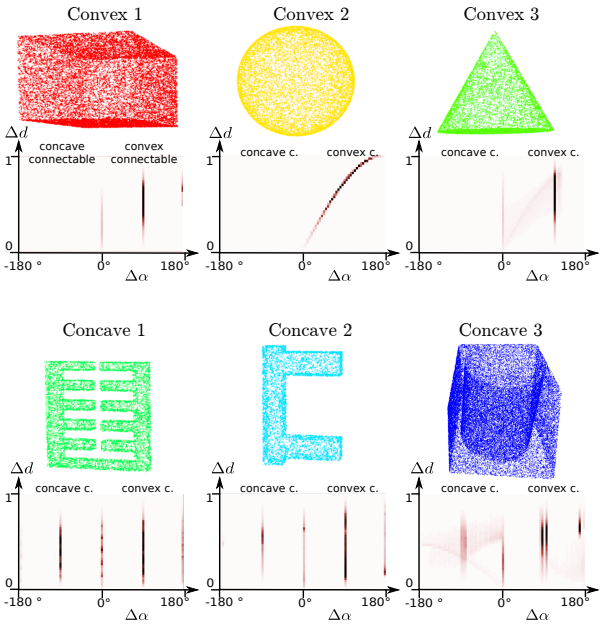


Fig. 6. Here we show three convex (top) and three concave shapes (bottom) together with their extended shape histogram SH^{Ext}. Each pair of points in the convex shapes are convex connectable (positive angles). This is why we do not have entries for concave connectable pairs (negative angles) in contrast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

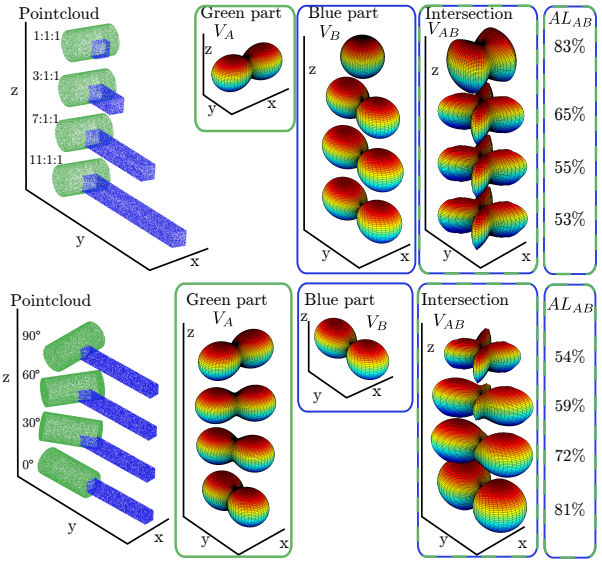


Fig. 7. Visualization of the influence of changing part shape (top) and relative orientation (bottom) on the variance blobs, intersection blobs, and the alignment number. V_A, V_B: Variance blobs for the green parts (A) and the blue parts (B). The bins of both variance blobs sum up to 1, i.e., $BS(V_A)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

AL. This however only works for parts which are elongated enough, which many times is not the case. An example is shown in Fig. 7 top: The more the blue part is scaled down, the harder it is to define a proper axis of elongation. This would lead to a discontinuous jump in the alignment number close to the 1:1:1 case.

In the following, we show how to circumvent this problem and how we define AL: First, we represent each part with a so-called *variance blob* V. The *variance blob* represents the variance of the point distribution of a part in all possible directions. To denote a direction $\vec{r}$ in 3D we use spherical coordinates, i.e., $\vec{r}(\theta, \phi)$. θ denotes the polar angle (measured from the pole), and ϕ represents the azimutal angle. $\vec{r}$ is L2-normalized, such that $||\vec{r}|| = 1$. The variance of the point cloud along a direction $\vec{r}(\theta, \phi)$ is calculated using:

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

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

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

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

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

A *variance blob* represents the variance in the Euclidean space of the point cloud. This is why the aspect ratio / elongation of a part is directly visible in the elongation of its *variance blob*. Exact shape detail (like round, cylindric, etc.) is, in contrast, ignored.

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

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

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

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

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

$$V^i_{AB} = \min(V^i_A, V^i_B). \quad (4)$$

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

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

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

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

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

To calculate AT_{BA} we use the value on B's variance blob instead. This is why $AT_{AB} \neq AT_{BA}$. The non-commutative property of the attachment is best explained looking at Fig. 8. Since the blue part is highly rotational symmetric it does not matter if we attach another part to the top or any of the sides. On the contrary, the green part is anisotropic, hence it makes a big difference for the functionality/usability in which direction/locations we attach other parts.

D. Object Representation

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

Again using the kd-trees, we calculate the closest distance for K random points from part A to the points in part B and vice versa. If any of these distances is smaller than τ, we

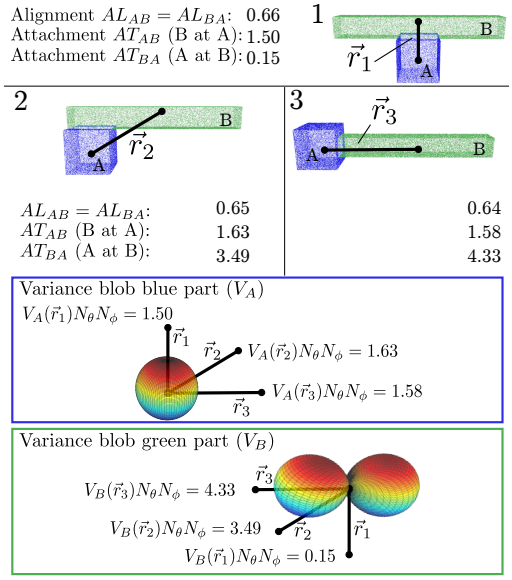


Fig. 8. Three objects consisting of identical parts. The blue part A (cube) is shifted from the long end of the green part B (elongated cuboid) to the middle. We did not draw the directions for the vectors $\vec{r}$ since values on opposite sites of the variance blobs are equal (the variance calculation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

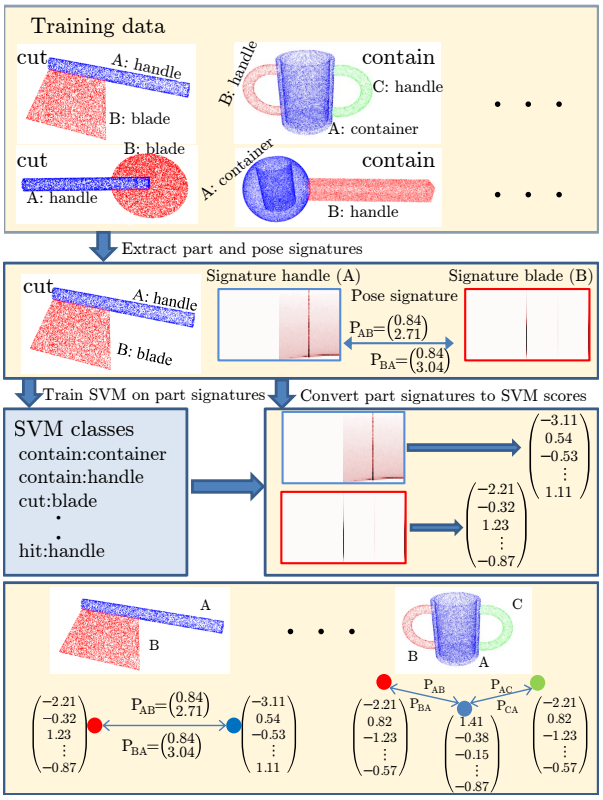


Fig. 9. Training procedure for the function analyzer.

consider the parts as touching. For the experiments we use $M = 30$ and $K = 1000$. Often parts are separated by a large margin, which makes this part of the algorithm robust.

V. Function Analyzer

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

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

A. Training

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

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

In the next step we calculate part and pose signatures (see Section IV-B and IV-C) for the training set. Inspired by the idea of Torresani et al. [40], we use the output of a Support-Vector-Machine (SVM) to convert the raw part signatures to second order signatures. Using a support-vector-machine at this level allows the function analyzer to better generalize across parts with the same function. At the same time it emphasizes properties of the raw signatures, which are important for the discrimination of different functional parts. We train a one-vs-rest SVM with a Chi-square kernel on the part signatures using each unique combination of function and part-name as a label (cut:blade, hit:head, contain:container, ...). For K functions with N parts each, this would result in $K \cdot N$ classes, which accordingly results in a $K \cdot N$ dimensional second order part signature. These signatures, the pose signatures, the graph structure, and the labels (for full objects and parts) are then stored for the later testing. Please note that we neither require all objects to have the same number of parts, nor that they should be composed of all possible parts. For example, a fillable object does not need a handle or can have multiple handles and still retains its functionality.

B. Testing

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

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

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

$$U(A_i) = \min(N_X, N_Y). \tag{7}$$

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

This pruning is not only important for eliminating cases where required parts for a functionality are not being assigned, but it is mandatory when comparing two objects with many parts, as the number of total associations grows exponentially. For example comparing the 6 handle cup ($N_Y = 7$) to any of the forks with 4 tips ($N_X = 5$) shown in Fig. 10 starts with 16807 possible associations ($N_Y^{N_X}$), reduces to 2520 associations after the surjective mapping test and again reduces to 360 after checking the graph structure.

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

$$AS(Ai, Y) = \omega^{Pa} AS^{Pa}\big(Ai, Y\big) + \omega^{Po} AS^{Po}\big(Ai, Y\big), \tag{8}$$

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

To calculate AS^{Pa} and AS^{Po} we L2-normalize all signatures and calculate the L2-distance between all associated part and pose pairs from the graph $\mathcal{G}^Y = (\mathcal{V}^Y, \mathcal{E}^Y)$ and $\mathcal{G}^{Ai} = (\mathcal{V}^{Ai}, \mathcal{E}^{Ai})$. The L2-distance between two signatures ranges between 0 and 2. This follows from the triangle inequality for two L2-normalized vectors x, and y: $||x - y|| \leq ||x|| + ||y|| = 2$.

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

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

and

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

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

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

$$OS_{X,Y} = \max_{Ai} \Big(AS(Ai, Y)\Big). \tag{11}$$

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

VI. Experimental Evaluation

A. M1 and M2 Benchmarks

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

For this experiment we let humans assign the most probable primitive functionality (see Fig. 1) to all objects. From all the human annotations we use the function (and the part assignments) with most votes as ground-truth. For the first benchmark (M1) we limit the objects to two parts. The second benchmark (M2) drops this restriction by allowing objects to have any number and type of parts, which increases intra-class variance. Both benchmarks consist of 72 models. Some example objects from M2 are shown in Fig. 10.

[9]

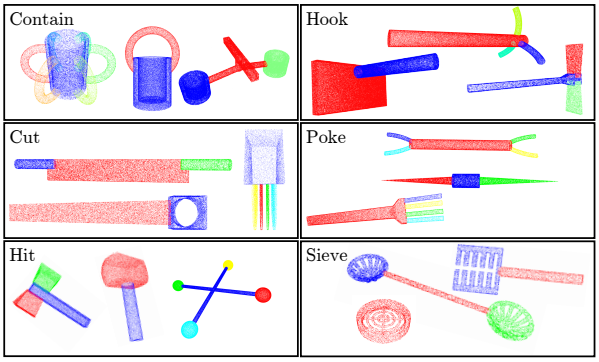


Fig. 10. Eighteen example objects from the M2 benchmark.

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

We use the following convention to name the classification pipelines:

1) Baseline centroid SHOT classifier trained on the full object (SHOT-C).
2) Baseline averaged SHOT classifier trained on the full object (SHOT-A).
3) Baseline ESF classifier trained on the full object (ESF).
4) Classifiers trained on parts ignoring pose. Superscript P (e.g. ESF^{P}).
5) Classifiers trained on parts together with the pose signatures. Superscript PP (e.g. ESF^{PP}).
6) Basic Shape Histogram using X distance and Y angle bins (SH$_{X,Y}$).
7) Extended Shape Histogram using X distance and Y angle bins (SH$^{Ext}_{X,Y}$).

B. Models and Scans

Furthermore, we investigate how well we can generalize across domains to polygonal models and scans from the databases 3Dcadbrowser, KIT Object Model Web Database (OMWD), tf3dm, and thingiverse. We sample equi-density random points on the faces of each model. Normals are calculated using the first three vertices of each face. For all objects we generate segmentations and ground truths using the *CPC*-algorithm. We use the SH$^{Ext,PP}_{4,4}$ classification pipeline together with all training images from the set M1 to determine the functionality of the objects.

VII. Results

A. M1 and M2 Benchmarks

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

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

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

B. Models and Scans

Figure 12 shows the segmentation and classification of several models and scans. Shown is the highest scoring function as well as the second highest if the difference is less than 2%. We empirically determined this number. Allowing a confidences difference of 5 % leads to wrong secondary function assignments in the case of difficult objects (like the double bladed cut object, which has a low confidence for the first function). This can be alleviated using a bigger training set.

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

The hammer in the bottom right corner shows interesting results. The function classifier retrieves two high scoring functions. It can be used like an ordinary hammer for hitting. Additionally, it was labeled with the function poke, which

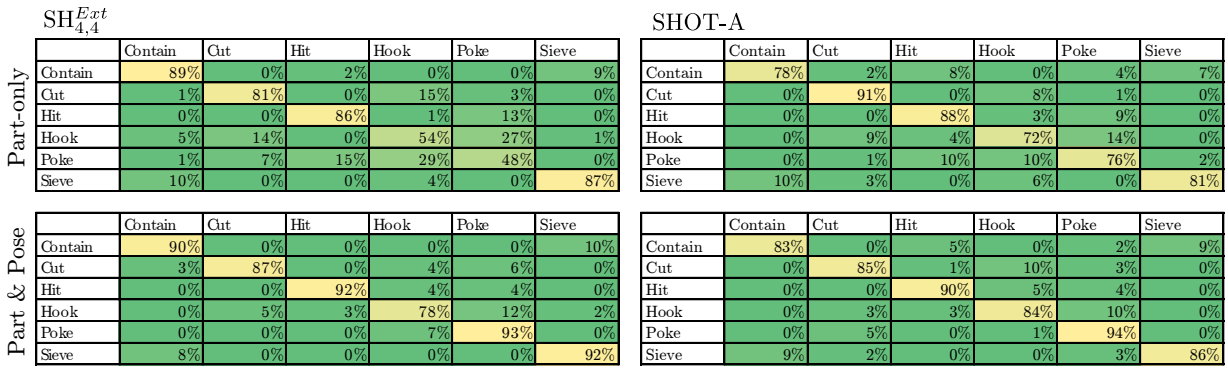


Fig. 11. Confusion matrices showing the importance of Pose information to the classification pipelines. It is especially important for telling hook from poke objects, as they mostly differ in the way parts are attached and aligned.

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

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

enables an agent to use the hammer as a improvised tool to drill holes into soil for instance.

VIII. DISCUSSION

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

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

However, some limitations and potential future work exists: First, we largely ignore size of objects, both for the part-signatures as well as for the pose-signatures. At this point we are interested in the general compatibility to tool-functions without looking at the target properties (e.g., target's size, material, and so on). When executing an action, the target properties will become important. For this, one would likely need to add a second layer of reasoning about tool-target compatibility. For example: If the target is soft (e.g., soil), the tool can be made out of wood. If the target is hard (e.g., wood), the tool needs to be made of metal or stone. If target is 5 liters of liquid, the container must have at least this volume. Still, for assessing general function-compatibility (not target compatibility), size can be ignored in our opinion. For

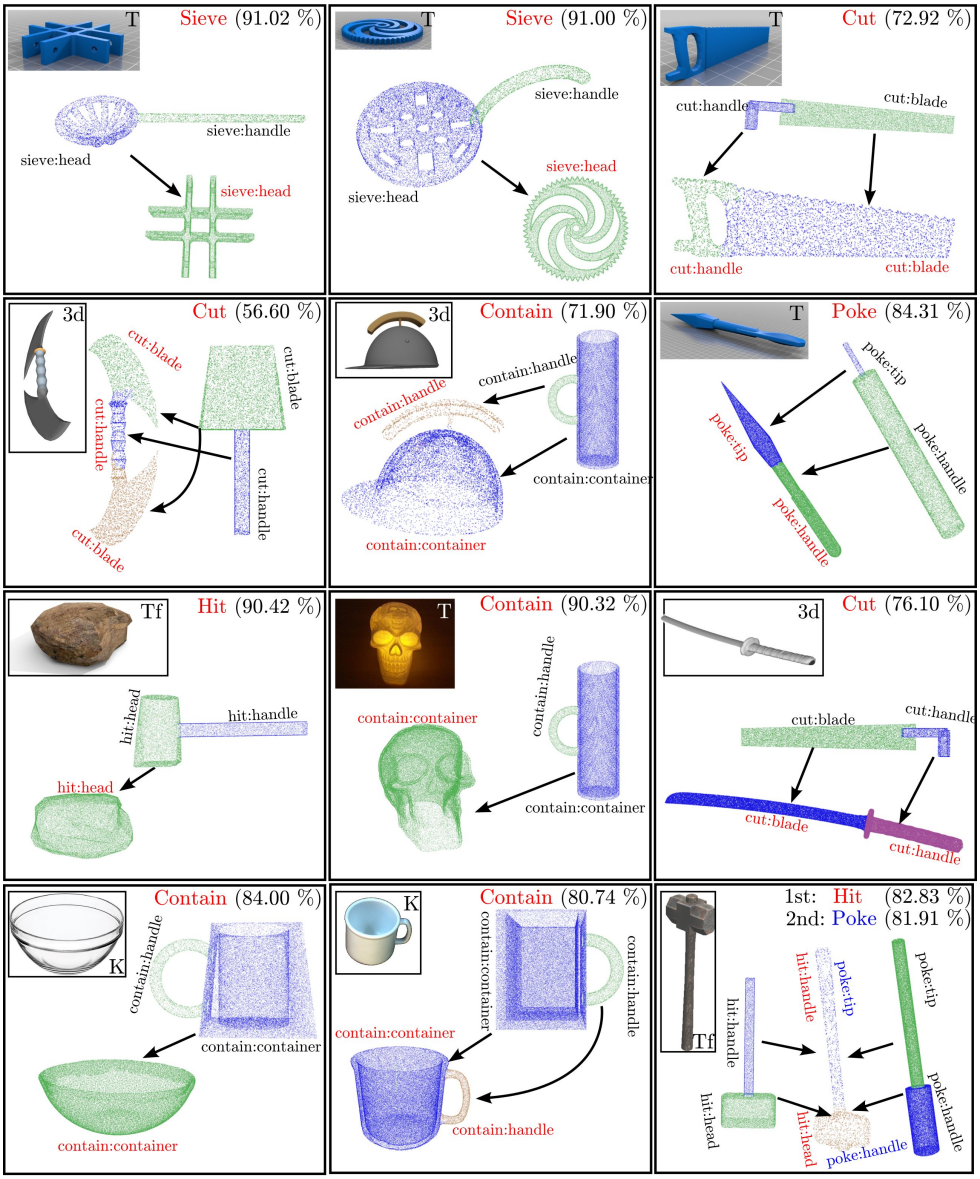


Fig. 12. Analysis of models and scans from Thingiverse (T), tf3dm (Tf), KIT OMWD (K), and 3Dcadbrowser (3d). All objects have been segmented using the *CPC* algorithm. The object segmentation as well as the most similar object are shown as colored point clouds. Additionally, the highest scoring function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

example, take a barrel and a cup. Both objects largely differ in their size, still they are both used to contain a liquid. Therefore, ignoring size largely increases generalization performance of the algorithm.

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

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

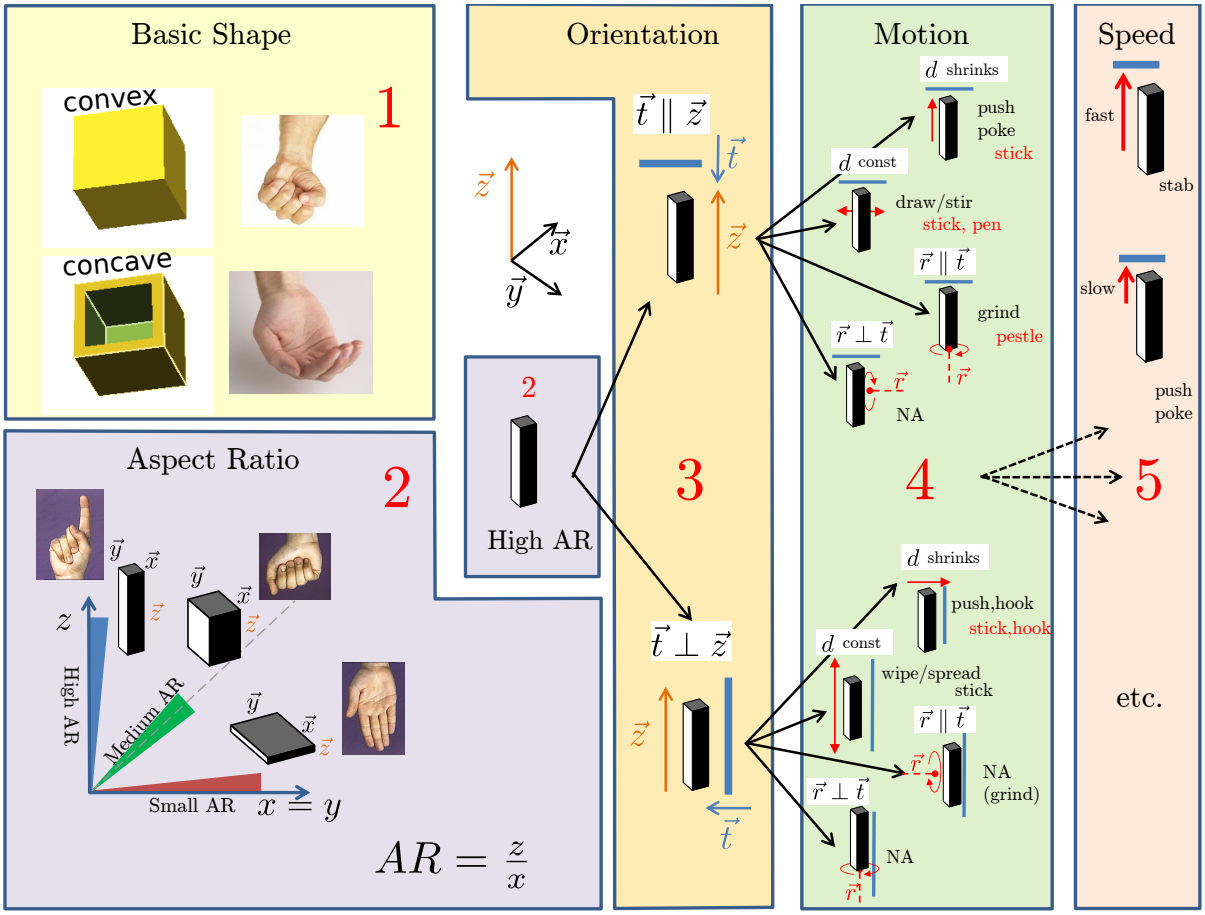


Fig. 13. Ontology of simple tools (reduced to the manipulator) derived from hand shapes and functions. Knowing the target surface (blue line) with normal $\vec{t}$ as well as the way the manipulator acts upon it, allows us to infer the function of the manipulator in this action. Red labels in 4 denote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

IX. Bootstrapping Tool Use - An Extension

In the current study we have shown our results on classifying objects (tools) by considering their parts and part relations. We have motivated our approach by the use of the human hand, which we consider as a possible structure for bootstrapping the understanding of tool function. That first part of the study was much motivated by the speculation that cognitive ability to use ones hand in different ways allows learning to transfer this knowledge to unknown objects, which are then recognized as tools. To support the conjecture of a possible cognitive hand-to-tool transfer we are, here, analyzing primary tools by also incorporating tool-dynamics. For this we design a tool-ontology and argue that its complexity remains quite limited.

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

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

We distinguish five levels (red numbers). The first three levels are strictly geometrical, where level 1 and 2 directly refer to the hand shape and level 3 to the arrangement of the hand (or manipulator) relative to the target object. Levels 4 and

[9]

5 take the movement patterns into account, too. Hence here we move from a pure object-guided (hand shape guided) ontology to the final one which uses manipulator shape, arrangement and the actual action pattern for tool classification. All this remains very reductionistic and we think *less is not possible*. We show in the end that such an ontology contains only 32 entries for different possible simple tools and how concept of wheels naturally emerges.

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

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

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

Level 3 addresses the relative orientation between hand shape (or tool) and target object (target surface $\vec{t}$) indicated by blue. Using our elongated tool with the tip pointing against the target surface, hence in a parallel way to the normal ($\vec{t} \parallel \vec{z}$, Level 3, top) could mean it is a poker. Using it in a perpendicular arrangement ($\vec{t} \perp \vec{z}$ Level 3, bottom) might make it a tool for spreading something out.

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

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

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

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

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

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

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

- a usually *fast* (level 5)
- *forward movement* is performed (level 4, d shrinks)
- *against* the target surface (level 3, $\vec{t} \perp \vec{z}$)
- with a *shovel-shaped* (Level 2, small AR)
- *concave* tool (level 1),

where the tool is pushed into something, like a pile of sand, to shovel it up. Now you could continue this movement staying in the same ontological class throwing the content off in a similar forward motion (e.g. continuing the shoveling motion). Or by comparison you could use the tuple (Action "empty[4]", Tool shovel) where this is the movement by which the filled shovel is turned to empty it and for which a different branch in the ontology exists.

X. Conclusion

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

[4]Means: to empty something.

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

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

classifiers where all this variance needs to be put into the training set. In contrast to our proposed method those will continue to fail as soon as human invention designs (artistic) objects, which humans – but not such machines – can easily classify.

ACKNOWLEDGMENT

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

REFERENCES

[1] C. V. Ward, M. W. Tocheri, J. M. Plavcan, F. H. Brown, and F. K. Manthi, "Early pleistocene third metacarpal from Kenya and the evolution of modern human-like hand morphology," *Proceedings of the National Academy of Sciences*, vol. 111, no. 1, pp. 121–124, 2014.

[2] T. Torigoe, "Comparison of object manipulation among 74 species of non-human primates," *Primates*, vol. 26, no. 2, pp. 182–194, 1985.

[3] F. Guerin, N. Kruger, and D. Kraft, "A survey of the ontogeny of tool use: from sensorimotor experience to planning," *IEEE Transactions on Autonomous Mental Development (TAMD)*, vol. 5, no. 1, pp. 18–45, 2013.

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

[5] I. Biederman, "Recognition-by-components: a theory of human image understanding," *Psychol Rev*, vol. 94, no. 2, pp. 115–147, 1987.

[6] ——, "Human object recognition: Appearance vs. shape," in *Shape Perception in Human and Computer Vision*, ser. Advances in Computer Vision and Pattern Recognition, S. J. Dickinson and Z. Pizlo, Eds. Springer London, Jan. 2013, pp. 387–397.

[7] ——, "The neural coding of parts and relations in object recognition," in *ECCV Workshop on Parts and Attributes (ECCVW)*, 2012.

[8] W. A. Richards and D. D. Hoffman, "Codon constraints on closed 2d shapes," in *Computer Vision, Graphics, and Image Processing*, 1985, pp. 265–281.

[9] D. D. Hoffman and W. A. Richards, "Parts of recognition," *Cognition*, vol. 18, pp. 65–96, 1984.

[10] M. Behrmann, M. A. Peterson, M. Moscovitch, and S. Suzuki, "Independent representation of parts and the relations between them: Evidence from integrative agnosia." *Journal of Experimental Psychology: Human Perception and Performance*, vol. 32, no. 5, pp. 1169–1184, 2006.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[12] N. Ahuja and S. Todorovic, "Connected segmentation tree; a joint representation of region layout and hierarchy," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[13] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 33, no. 5, pp. 898–916, 2011.

[14] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision (IJCV)*, vol. 57, no. 2, pp. 137–154, 2004.

[15] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan,

[9]

"Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 32, no. 9, pp. 1627–1645, 2010.

[16] P. Arbelaez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik, "Semantic segmentation using regions and parts," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3378–3385.

[17] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *European Conference on Computer Vision (ECCV)*, 2012, pp. 746–760.

[18] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 564–571.

[19] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski, "CAD-model recognition and 6DOF pose estimation using 3D cues," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 585–592.

[20] S. Stein, M. Schoeler, J. Papon, and F. Wörgötter, "Object partitioning using local convexity," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[21] B. Yao and L. Fei-Fei, "Modeling mutual context of object and human pose in human-object interaction activities," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 17–24.

[22] H. Pirsiavash and D. Ramanan, "Detecting activities of daily living in first-person camera views," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 2847–2854.

[23] C. Desai, D. Ramanan, and C. Fowlkes, "Discriminative models for static human-object interactions," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2010, pp. 9–16.

[24] A. Farhadi and M. A. Sadeghi, "Phrasal recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 35, no. 12, pp. 2854–2865, Dec. 2013.

[25] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[26] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman, "Discovering object categories in image collections," in *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[27] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1385–1392.

[28] D. Huber, A. Kapuria, R. Donamukkala, and M. Hebert, "Parts-based 3D object classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004, p. II82.

[29] M. Tenorth, S. Profanter, F. Balint-Benczedi, and M. Beetz, "Decomposing cad models of objects of daily use and reasoning about their functional parts," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 5943–5949.

[30] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *International Journal of Computer Vision (IJCV)*, vol. 61, no. 1, pp. 55–79, 2005.

[31] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Trans. Comput.*, vol. 22, no. 1, pp. 67–92, Jan. 1973.

[32] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang, "Contextual part analogies in 3d objects," *International Journal of Computer Vision (IJCV)*, vol. 89, no. 2, pp. 309–326, 2010.

[33] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 951–958.

[34] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter, "Voxel cloud connectivity segmentation - supervoxels for point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[35] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European Conference on Computer Vision (ECCV)*, 2010, pp. 356–369.

[36] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2011, pp. 2987–2992.

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

[38] L. A. Alexandre, "3D descriptors for object and category recognition: a comparative evaluation," in *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[39] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-dimensional object recognition and 6 DOF pose estimation," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 80–91, 2012.

[40] L. Torresani, M. Szummer, and A. Fitzgibbon, "Efficient object category recognition using classemes," in *Proceedings of the 11th European Conference on Computer Vision: Part I*, ser. European Conference on Computer Vision (ECCV), 2010, pp. 776–789.

[41] A. Uderzo and R. Goscinny, "Astérix et les normands," p. 17, 1967.

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

[43] S. Biasotti, S. Marini, M. Spagnuolo, and B. Falcidieno, "Sub-part correspondence by structural descriptors of 3D shapes," *Computer-Aided Design*, vol. 38, no. 9, pp. 1002–1019, 2006.

[44] H. Laga, M. Mortara, and M. Spagnuolo, "Geometry and Context for Semantic Correspondences and Functionality Recognition in Man-made 3D Shapes," *ACM Trans. Graph.*, vol. 32, no. 5, pp. 150:1–150:16, 2013.

[45] E. Rubin, *Visuell wahrgenommene Figuren. Copenhagen: Gyldenalske Boghandel 1915. Reprinted as: Figure and Ground*. edited by Beardslee, D. C. and Wertheimer, M and Princeton, N. J., Publ.:Van Nostrand, 1958.

[46] J. J. Koenderink and A. J. van Doorn, "The shape of smooth objects and the way contours end," *Perception*, vol. 11, no. 2, pp. 129–137, 1982.

[47] D. D. Hoffman and W. A. Richards, "Parts of recognition," *Cognition*, vol. 18, no. 13, pp. 65–96, 1984.

[48] M. L. Braunstein, D. D. Hoffman, and A. Saidpour, "Parts of visual objects: an experimental test of the minima rule," *Perception*, vol. 18, pp. 817–826, 1989.

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

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

Markus Schoeler studied physics at the University of Würzburg, Germany. He received his M.Sc. in physics for his research at the IOF-Fraunhofer Institute from the University of Jena in 2010. Currently he is pursuing his Ph.D. degree in Computer Vision at the University of Göttingen. Among his research interests are bottom-up object and scene partitioning, robot vision, object classification, and object cognition.

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

Summary: Function Perception

In this chapter we treated object perception at the *function* level. We see *function* perception as being tightly linked to part perception. Therefore, our main goal was an algorithm which is able to assign *function* to novel objects and their parts. In order to achieve this, we additionally needed to segment object-parts. Besides the supervised Object Function Assignment (OFA) algorithm, this required developing a series of unsupervised algorithms for the Object Partitioning (OP). Altogether, we contributed to solving the following four problems:

Problem 1: Doing computations in 3D is expensive.

Our contribution: The Voxel Cloud Connectivity Segmentation (VCCS) algorithm which creates an over-segmentation for a Point Cloud. It combines voxels into so-called supervoxels, which are very efficient to use (they have adjacency information stored) and do not cross object boundaries.

Problem 2: In order to recognize novel objects, we need to be able to segment unknown objects in a scene.

Our contribution: The Locally Convex Connected Patches (LCCP) algorithm which uses existing concavities between objects for the separation.

Problem 3: In order to get parts of objects, we need to be able to do unsupervised segmentation of parts (OP) for unknown objects.

Our contribution: The Constrained Planar Cuts (CPC) algorithm which uses concavities found in objects to induce cuts.

Problem 4: Assigning function to full objects as well as their parts.

Our contribution: We created an algorithm which describes objects at their part level. By comparing the part constellation of a new object to part constellations of known objects, it is able to assign one or multiple *functions* to objects and parts.

While all the algorithms in this chapter can be used in their own right, each of them contributed to achieve the main goal of assigning *function* to novel objects. This gives agents new and powerful ways to solve tasks using, for example, make-shift tool replacements.

7

Summary of Contributions

In this work we addressed Visual Object Perception (VOP) for artificial agents. We motivated the existence of the three levels of perception *Instance*, *Category* as well as *Functionality* and contributed to five sub-problems, namely Instance Recognition (IR) [1], Object Categorization (OC) [2, 3], Pose Estimation at the category level (PEC) (using a Deep Convolutional Neural Network (DCNN)) [4], unsupervised Object Partitioning (OP) (using three consecutive algorithms, Voxel Cloud Connectivity Segmentation (VCCS) [5], Locally Convex Connected Patches (LCCP)[6, 7], and Constrained Planar Cuts (CPC)[8]), and Object Function Assignment (OFA) [9]. All algorithms have been thoroughly benchmarked and achieved better than state-of-the-art results in their respective tasks. Before we summarize the particular contributions in the next sections, we would like to emphasize that most of the algorithms designed in the course of this work have been released to the public-domain and are available to the vision community as part of the Point Cloud Library (PCL).

7.1 Instance Recognition (IR)

We started to address the problem of IR in Chapter 4. A problem which requires dealing with the countless number of possible objects in our surrounding and objects which are visually similar to one another. Using the traditional classification strategy of offline-training on database-samples is not possible: There is an almost unlimited number of objects existing in the world. Consequently, creating a database for all potential objects is not possible. We there-

fore proposed a method [1] to create needed training samples on-the-fly using the agent itself with a highly autonomous algorithm. It can, by ways of exploration, be used to gather needed information about environment specific objects. We dealt with the low inter-class variance by using a novel *Radial Orientation Scheme*, which leads to highly invariant and discriminative object signatures.

7.2 Object Categorization (OC) and Category Pose Estimation (PEC)

While our IR pipeline allows an agent to discriminate objects, those remain isolated entities. Hence, it does not allow a machine to structure its world by combining objects into meaningful *categories*. In Chapter 5 we addressed this problem.

As we moved away from the countless number of individual objects to a smaller set of *categories*, we inevitably increased the intra-class variance due to many visually different objects being combined into single classes. To deal with this variance, one needs to use machine learning techniques which are able to draw complex and highly non-linear decision boundaries. Those algorithms, with DCNNs being among the most successful, are, due to the richness of their models, prone to over-fitting unless huge amounts of training data are available. We therefore proposed the *TransClean* algorithm (Section 5.1)[2, 3] which provides such methods with an almost unlimited number of high quality training samples by utilizing information from a label+verb or label+adjective pair. Such pairs can be extracted from commands given to agents or manually defined. Additionally, it allows for disambiguation of polysemic classnames. Even though both, our previous IR-classification pipeline as well as the *TransClean* algorithm, are capable of generating training sets on-demand, both approaches fundamentally differ. The former generates training sets by capturing samples; *TransClean* maps word-based given labels and contexts to the image domain using online word-databases, image-databases, and translation services in concert.

While plain categorization of objects in scenes is desirable for many applications, it is insufficient for complicated robotic manipulation of those objects. In this case one needs to do Pose Estimation at the category level of the encountered objects. While pose estimation is usually addressed by finding the 3D-affine transformation of known models to new observations in the context of *instance* perception, it is not applicable at the *category* level, because we do not have information (models) about particular objects. This motivated us to design our combined Categorization-Localization-Pose Estimation framework (Section 5.2)[4] which operates on object *categories* rather than *instances* by using a DCNN architecture. Here we treated all three problems in unison using a common large network. We solved the need for huge amounts of annotated training data for such network by automatically assembling synthetic scenes with thousands of different models from various *categories*. To close the gap between

training on synthetic scenes and testing on real scenes, we simulated the model and geometry of real RGB-D sensors. Additionally, we adapted the trained DCNN using transfer-learning on the *NYU Depth V2* training-data. Finally, we tested on the *NYU Depth V2* testing-data and achieved better than state-of-the-art results.

7.3 Object Partitioning (OP) and Function Assignment (OFA)

Up to here the agent has obtained the ability to annotate objects in a scene with a *category* as well as a 6 DoF pose. Still this did not allow the transfer of existing knowledge to unusual objects and novel *categories*. As this is a precursor to bootstrapping tool usage and makeshift tool replacement in agents, we designed a sequence of algorithms to tackle this OFA problem at the part level (Chapter 6). Since we deal with unknown objects most of the employed algorithms needed to operate model- and category-independent, thus unsupervised. In a first step, we therefore designed the VCCS algorithm (Section 6.1)[5] as a bridge between the sensory output and higher level processing. By combining voxels from a scene to supervoxels, which adhere to object and part boundaries, we were able to reduce noise from the sensors and the computational cost of the following algorithms to a feasible amount. Building upon the psychophysical evidence that humans use concavity information for object and part perception, we designed two algorithms LCCP (Section 6.2)[6, 7] and CPC (Section 6.3)[8] which are used to segment full scenes into separate objects and those objects into constituent parts.

This finally allowed us to create an algorithm (Section 6.4)[9] which treats objects merely as an assembly of individual parts. It analyses parts as well as part constellations in order to make predictions about an object's compatibility to certain tool-functionalities. Using this intermediate part level between an object and higher level classification not only reduced the intra-class variance to a viable degree but also provides us with information about the *functionality* of each particular part (like being a blade or a handle). This is indispensable for the actual tool-use by an agent.

8

Conclusion and Future Work

At the beginning of this thesis we named Visual Object Perception (VOP) as one of the fundamental problems an artificial agent (*e.g.*, a robot) has to master before being of any assistance to humans. Motivated by our own experience, we stated that object perception is closely connected to the task an agent is set to solve. Therefore, not taking *action* into account when treating artificial perception leads to an ill-posed problem. We, as the most advanced biological agents, are able to perceive our surrounding with apparent ease. Without much thinking we can process the complex visual stream into meaningful entities. What we define as an entity, again, depends on the task. While it is still an open question how humans actually do it, here we contributed to the solution of this problem for artificial systems. For this we created algorithms which produce results that resemble those from visual cognition found in biological agents. Perceiving objects in such a comprehensive way gives machines fundamentally powerful ways of making sense of and interacting with their environments. Applications include robots working in household environments, doing construction, or assisting humans.

Still, some limitations exist. The algorithms we developed have been used with the operator dictating which level of perception should be employed. For a fully autonomous system, we would need to combine all perception levels into one system, where pipelines run concurrently. This would allow a machine to see an object with a hierarchy of labels (*e.g.*, this is the blue-white cup, it is of the *category cup*, and it can *contain*). The machine would than need to find out which granularity of perception is required for solving the task. For example, the pres-

ence of a direct or indirect article in a command or instruction gives a strong cue whether we are dealing with Instance Recognition (IR) or not. Another potential use for such a concurrent system would be failure recovery. For example, if the exact *instance* is not available in the scene (*e.g.*, the blue-white cup), the system could propose to use another *cup* or even another fillable object instead.

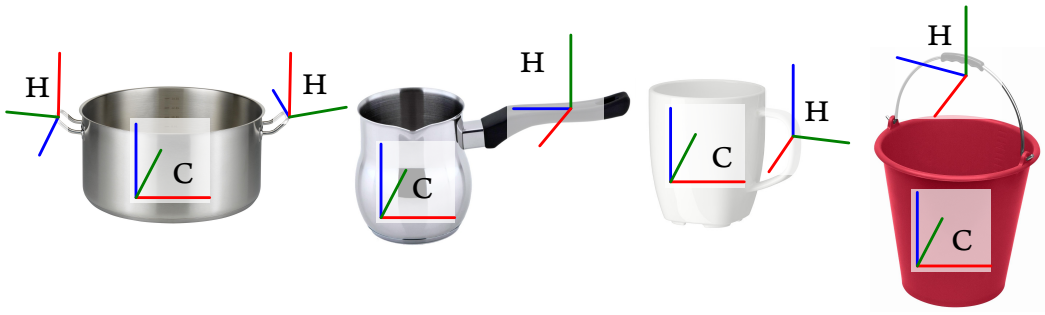


Figure 8.1: Part-based Pose Estimation at the function level (PEF) proposal. Generic pose for a *functional* object does not exist. Instead pose for parts needs to be defined as shown here for *contain* objects. The coordinate system x y z defines how a part needs to be used. For handles (H) z defines the axis around which a grasp needs to be done; y defines the direction of approach. For containers (C) z defines the direction of the opening.

A further limitation of the current work is the dependency on 3D of the unsupervised algorithms. This is especially true for the Constrained Planar Cuts (CPC) algorithm which works much more reliable for point clouds from multiple views, than for single views. The reason is that it needs to record the concavities of an object to induce cuts (if those are not visible, no cuts are being induced). While one can always argue that this approach can be combined with Structure from Motion (SfM) methods to record objects by ways of exploration, it would be a desirable property for a method to guess where concavities exist, by ways of *shape completion* for example [77, 78]. Furthermore, one could potentially use a Deep Convolutional Neural Network (DCNN) to partition objects into parts. For this we could combine our synthetic scene generation (Section 5.2) with our Object Partitioning (OP) algorithm (Section 6.3). This way we could simulate single-view scenes with ground truth partitioning for the training of a DCNN.

Finally, Pose Estimation at the function level (PEF) is another unsolved problem. Even defining a generic pose for *functional* classes is not trivial. We believe that it does not even exist for full objects. Still, there may be a potential way out: We use pose in our robotic context to describe how to use a perceived object. At the *functionality* level full-objects are, anyhow, seen as an assembly of various parts, some of which interact with the target and some of which can be handles for grasping. Our algorithm in Section 6.4 is able to label each part in context of a certain *functionality*. This allows us to define the tool-end and the handle of each object. While pose for full objects may not be defined, pose of handles or tool-ends could be defined. For example, the reference frame of handles could be defined in such way that the z-axis points along the axis around which a grasp needs to be done; the y-axis could point into the direc-

tion from which to approach the object. A containers z-axis could point in direction of the opening. This proposal is depicted in Fig. 8.1. Although we show discrete axes for the pose, exact orientation for some are irrelevant and/or not well defined due to symmetries (*e.g.*, the container axes x and y).

References

[1] M. Schoeler, S. Stein, J. Papon, A. Abramov, and F. Wörgötter. Fast self-supervised on-line training for object recognition specifically for robotic applications. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2014. ©2014 INSTICC, reprinted with permission.

[2] M. Schoeler, F. Wörgötter, M. Aein, and T. Kulvicius. Automated generation of training sets for object recognition in robotic applications. In *IEEE/RSJ 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, 2014. ©2014 IEEE, reprinted with permission.

[3] M. Schoeler, F. Wörgötter, J. Papon, and T. Kulvicius. Unsupervised generation of context-relevant training-sets for visual object recognition employing multilinguality. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2015. ©2015 IEEE, reprinted with permission.

[4] J. Papon and M. Schoeler. Semantic pose using deep networks trained on synthetic rgb-d. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. in press, ©2015 IEEE, reprinted with permission.

[5] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter. Voxel cloud connectivity segmentation - supervoxels for point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. ©2013 IEEE, reprinted with permission.

[6] S. Stein, F. Wörgötter, M. Schoeler, J. Papon, and T. Kulvicius. Convexity based object partitioning for robot applications. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. ©2014 IEEE, reprinted with permission.

[7] S. Stein, M. Schoeler, J. Papon, and F. Wörgötter. Object partitioning using local convexity. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. ©2014 IEEE, reprinted with permission.

[8] M. Schoeler, J. Papon, and F. Wörgötter. Constrained planar cuts - object partitioning for point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. ©2015 IEEE, reprinted with permission.

[9] M. Schoeler and F. Wörgötter. Bootstrapping the semantics of tools: Affordance analysis of real world objects on a per-part basis. *IEEE Transactions on Autonomous Mental Development (TAMD)*, 2015. in press, ©2015 IEEE, reprinted with permission.

[10] J. Papon, M. Schoeler, and F. Wörgötter. Spatially stratified correspondence sampling for real-time point cloud tracking. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2015. ©2015 IEEE, reprinted with permission.

[11] A. G. Buch. *In Search of 3D Poses: Advancing Feature Description, Matching and Estimation for Robotic Applications.* PhD thesis, University of Southern Denmark, 2015.

[12] S. Papert. The Summer Vision Project. 1966. URL http://dspace.mit.edu/handle/1721.1/6125. Accessed on 17.07.2015.

[13] C. Geib, K. Mourao, R. Petrick, N. Pugeault, M. Steedman, N. Krueger, and F. Wörgötter. Object action complexes as an interface for planning and robot control. Interational Conference on Humanoid Robots, 2006.

[14] F. Wörgötter, A. Agostini, N. Krüger, N. Shylo, and B. Porr. Cognitive agents - a procedural perspective relying on the predictability of object-action-complexes (oacs). *Robotics and Autonomous Systems*, 57(4):420––432, 2009.

[15] N. Krüger, J. Piater, F. Wörgötter, C. Geib, R. Petrick, M. Steedman, A. Ude, T. Asfour, D. Kraft, D. Omrcen, B. Hommel, A. Agostino, D. Kragic, J. Eklundh, V. Kruger, C. Torras, and R. Dillmann. A formal definition of object action complexes and examples at different levels of the process hierarchy, 2009.

[16] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010. draft.

[17] B. Yao, G. Bradski, and L. Fei-Fei. A Codebook-Free and Annotation-Free Approach for Fine-Grained Image Categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[18] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[19] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(10), 2005.

[20] E. Tola, V. Lepetit, and P. Fua. DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(5):815–830, May 2010.

[21] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding (CVIU)*, 110(3), 2008.

[22] A. E. Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, 1997.

[23] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *European Conference on Computer Vision (ECCV)*, pages 356–369. 2010.

[24] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[25] L. A. Alexandre. 3d descriptors for object and category recognition: a comparative evaluation. In *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[26] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[27] C. Papazov and D. Burschka. An efficient ransac for 3d object recognition in noisy and occluded scenes. In *Asian Conference on Computer Vision (ACCV)*, pages 135–148. Springer, 2011.

[28] H. Wolfson and I. Rigoutsos. Geometric hashing: an overview. *IEEE Computational Science Engineering*, 4(4):10–21, 1997.

[29] P. J. Besl and N. D. McKay. Method for registration of 3-D shapes. In *Robotics-DL tentative*, 1992.

[30] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Internation conference on 3D Digital Imaging and Modeling*, 2001.

[31] A. Gopnik and A. Meltzoff. The development of categorization in the second year and its relation to other cognitive and linguistic developments. *Child development*, pages 1523–1531, 1987.

[32] J. Piaget, M. Cook, and W. W. Norton. *The origins of intelligence in children*, volume 8. International Universities Press New York, 1952.

[33] J. Block. Assimilation, Accommodation, and the Dynamics of Personality Development. *Child Development*, 53(2):281–295, April 1982.

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

[35] G. Csurka and F. Perronnin. Fisher Vectors: Beyond Bag-of-Visual-Words Image Representations. *Computer Vision, Imaging and Computer Graphics: Theory and Applications*, pages 28–42, 2011.

[36] F. Perronnin and C. Dance. Fisher Kernels on Visual Vocabularies for Image Categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[37] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[38] L. Bo, X. Ren, and D. Fox. Hierarchical matching pursuit for image classification: Architecture and fast algorithms. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2115–2123. Curran Associates, Inc., 2011.

[39] L. Bo, X. Ren, and D. Fox. Multipath sparse coding using hierarchical matching pursuit. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[40] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1 edition, September 1998. ISBN 0471030031.

[41] D. Meyer, F. Leisch, and K. Hornik. The support vector machine under test. *Neurocomputing*, 55(1–2):169–186, September 2003.

[42] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[43] A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[44] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[45] F. Moosmann, E. Nowak, and F. Jurie. Randomized Clustering Forests for Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(9): 1632–1646, September 2008.

[46] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[47] Y. Amit and D. Geman. Shape Quantization and Recognition with Randomized Trees. *Neural Computation*, 9(7):1545–1588, 1997.

[48] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.

[49] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42, April 2015.

[50] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014.

[51] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2014.

[52] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[53] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*, 2014.

[54] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, 2014.

[56] J. J. Gibson. The theory of affordances. *Hilldale, USA*, 1977.

[57] J. McGrenere and W. Ho. Affordances: Clarifying and evolving a concept. In *Graphics Interface*, 2000.

[58] D. Norman. *The Design of Everyday Things*. Basic Books, 2002. ISBN 978-0-465-06710-7.

[59] W. SA. Wenger sets world record for most functional penknife. 2010. URL http://www.wenger.ro/wenger-sets-world-record-for-most-functional-penknife/index.html. Accessed on 02.08.2015.

[60] X. He, R. S. Zemel, and M. A. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[61] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[62] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3d mesh segmentation and labeling. *ACM Transactions on Graphics*, 29(4):1, July 2010.

[63] A. Desolneux, L. Moisan, and J. M. Morel. *From Gestalt Theory to Image Analysis: A Probabilistic Approach*, volume 34. Springer, 2007.

[64] I. Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

[65] I. Biederman. The neural coding of parts and relations in object recognition. In *ECCV Workshop on Parts and Attributes*, 2012.

[66] W. Richards and D. D. Hoffman. Codon constraints on closed 2d shapes. In *Computer Vision, Graphics, and Image Processing*, 1985.

[67] D.D. Hoffman and W.A. Richards. Parts of recognition. *Cognition*, 18:65–96, 1984.

[68] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *European Conference on Computer Vision (ECCV)*, 2010.

[69] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011.

[70] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, 2007.

[71] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[72] I. Sobel and G. Feldman. A 3x3 isotropic gradient operator for image processing presented at the stanford artificial intelligence project (sail), 1968.

[73] X. Ren and J. Malik. Learning a classification model for segmentation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 2003.

[74] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, 2009.

[75] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes. Layered object detection for multi-class segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010.

[76] D. Hoffman and W. Richards. Parts of recognition. *Cognition*, 18(1–3):65–96, 1984.

[77] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: shape completion and animation of people. In *ACM Transactions on Graphics (TOG)*, 2005.

[78] S. Thrun and B. Wegbreit. Shape from symmetry. In *IEEE International Conference on Computer Vision (ICCV)*, 2005.

Appendices

A

Real-Time Point Cloud Tracking

Object perception in this thesis was limited to still frames. In robotic applications, however, one often deals with continuous visual streams. Employing our algorithms on each frame independently would cause a massive computational overload for systems, and would not make use of the coherence between consecutive frames. While this is not in the main focus of this thesis, the Voxel Cloud Connectivity Segmentation (VCCS) segmentation served as a strong basis for point cloud *tracking* as well. By continuously tracking the transformation which maps objects from frame to frame, object partitioning, pose, and label can be, once initialized, forwarded through a whole sequence. In the paper

[10] Papon, J. and **Schoeler, M.** and Wörgötter, F.: "Spatially Stratified Correspondence Sampling for Real-Time Point Cloud Tracking", *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2015. See page 147.

we show that one can use supervoxels as sampling points for estimating the correct 3D affine transformation between the model and the observation. While accuracy stays stable, it greatly reduces run-time on CPUs by orders of magnitudes, outperforming even state-of-the-art methods on GPUs. We demonstrate that this increase in efficiency permits online 6 DoF multi-target tracking on standard computer hardware.

Spatially Stratified Correspondence Sampling for Real-Time Point Cloud Tracking

Jeremie Papon, Markus Schoeler, and Florentin Wörgötter
Bernstein Center for Computational Neuroscience (BCCN)
III. Physikalisches Institut - Biophysik, Georg-August University of Göttingen
jpapon@gwdg.de, mschoeler@gwdg.de, worgott@gwdg.de

Abstract

In this paper we propose a novel spatially stratified sampling technique for evaluating the likelihood function in particle filters. In particular, we show that in the case where the measurement function uses spatial correspondence, we can greatly reduce computational cost by exploiting spatial structure to avoid redundant computations. We present results which quantitatively show that the technique permits equivalent, and in some cases, greater accuracy, as a reference point cloud particle filter at significantly faster run-times. We also compare to a GPU implementation, and show that we can exceed their performance on the CPU. In addition, we present results on a multi-target tracking application, demonstrating that the increases in efficiency permit online 6DoF multi-target tracking on standard hardware.

1. Introduction

Visual tracking is a crucial challenge for many computer vision applications such as visual surveillance, action recognition, and robotic demonstration learning. In these, visual tracking serves as a precursor to higher-level inference, making robust tracking fundamental to their success. Multi-target visual tracking (MTVT) is a well-established field which goes back over thirty years [6]. One popular tracking methodology is Sequential Bayesian Filtering (SBF), which recursively determines the time-changing posterior distribution of target states conditioned on previous observations. Particle Filtering has received considerable attention as a method of approximating this posterior. It was first introduced to the vision community by Isard and Blake [8] for single targets, and was subsequently extended to multiple targets [7, 19, 20].

There are two standard approaches that have been used to extend the Particle Filter to multiple targets. The first represents all targets jointly in a single particle filter by assigning individual particles to particular labels [18]. This means that, for a given total number of particles, there will be fewer for each individual target - resulting in reduced accuracy. The second approach is to add additional dimensions to the state space for each additional target [17]. Unfortunately, this approach quickly increases the dimensionality of the state space, which also results in a need for a very high number of particles for the filter to remain accurate.

In both of the above approaches, the computational complexity increases exponentially as targets are added (for constant level of accuracy). As a consequence of this, it is beneficial to use a separate particle filter for each target. One way of doing this is to add factors to the observation and/or process models of the filters which explicitly model occlusions and interactions between targets [9, 13]. Alternatively, one can use a discrete processing step to resolve the association of target detections [10, 1].

While these approaches have attempted to address the problem of multiple targets, in general they suffer from one fundamental problem - they all significantly increase the computational resources required. This increase can be seen in the need for more particles - due to assignment of particles to individual targets, a larger state space, or independent filters for each target. While there has been work addressing this problem by offloading processing to a GPU [3], in this work we take a different approach, and search for fundamental changes to the point cloud correspondence particle filter which can reduce computational complexity without affecting accuracy.

The primary contribution of this work is the use of a supervoxel-based stratified sampling approach to greatly reduce the computational complexity of point cloud correspondence particle filtering. We show that the approach allows performance (on a standard CPU) exceeding that which can be obtained on a recent GPU implementation [3]. Additionally, we present extensive experiments demonstrating the benefits of this approach, as well as show qualitative results from a real-world application. We have released

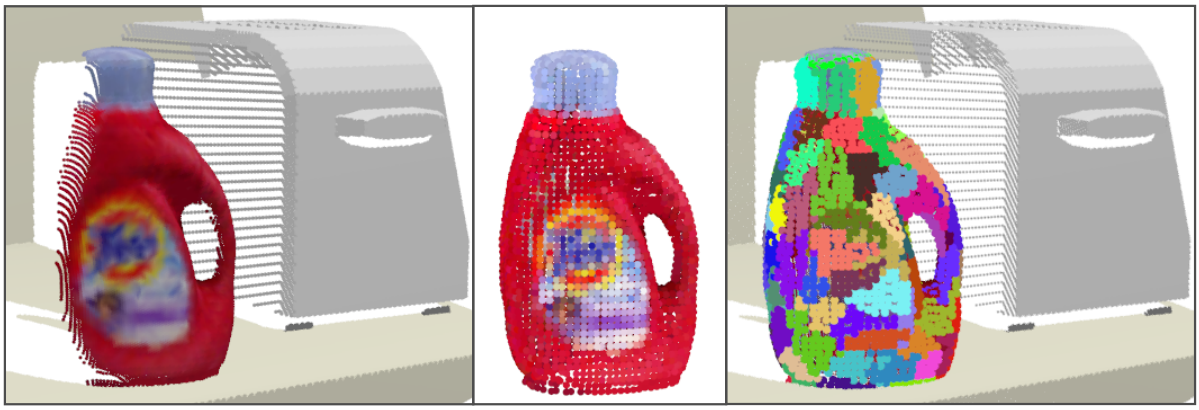


Figure 1. Example of data from "Tide" sequence. The left frame shows an example of the raw input cloud. Sampling effects from the synthetic RGB-D camera are visible in the quantization of points, especially on the edges of objects. The middle frame shows the voxelized model representation we use, while the right frame shows an example of supervoxel strata used for sampling with $R_{seed} = 0.07m$.

code for our tracker as part of the Point Cloud Library [1] so that the community may take advantage of its quick performance in their applications.

This paper is organized as follows: In Section 2 we present the point cloud correspondence-based particle filter, and then present the sampling approach in Section 3. We then give results from experiments on both synthetic and real-world data in Section 4 and discuss the impact of sampling on computational complexity and tracker error.

2. Particle Filters in 3D

The underlying mechanics of 3D point cloud correspondence particle filtering remain the same as in other particle filters, and so we shall not discuss them extensively here; for a detailed introduction to the topic, we refer the reader to [5] or [20]. Rather, we shall only discuss the aspects that differentiate it - the models and how they are scored and propagated. The models here consist of point clouds, and the measurement function relies on point to point correspondence for scoring, rather than a global per-detection metric (such as a histogram distance, commonly used in 2D trackers). The dynamic model uses real-world 3D coordinates which also include orientation, rather than 2D pixel coordinates in the image plane. The primary novelty of the approach we present here lies in how we score individual particle predictions using the measurement model.

2.1. Model Representation

In this work we use voxelized 3D models of tracked objects, allowing tracking through any change in pose, and additionally allowing accurate tracking of pose itself. We represent objects as clouds of voxels corresponding to the surface of the object. A visual representation of such a model is given in Figure 1.

Points for objects are stored in a model-centered reference frame (which we shall denote with superscript ^{m}), with each containing an XYZ position, an HSV color, as well as a surface normal vector. That is, each point p of the model k consists of a nine-dimensional vector:

$$p_k^m = [x^m, y^m, z^m, H, S, V, n_x, n_y, n_z], \quad (1)$$

and a model for an object O_k consists of a vector of n_k such points p^m:

$$O_k^m = [p_0^m ... p_{n_k}^m]. \quad (2)$$

Points of an object model given above are model-relative - they must be transformed into the world coordinates in order to evaluate their fit to observations.

2.2. Dynamic Model

Our trackers use a time-dependent state vector consisting of translation and rotations around the object reference frame x-axis (roll - γ), y-axis (pitch - β), and z-axis (yaw - α). This yields a position state vector for particle j at time t of

$$\mathbf{x}_t^j = [d_x, d_y, d_z, \gamma, \beta, \alpha]. \quad (3)$$

Each object model is tracked using a set of N such particles. Additionally, we have velocity state vector

$$\mathbf{v}_t = [v_x, v_y, v_z, v_\gamma, v_\beta, v_\alpha], \quad (4)$$

which is not tracked individually per particle, but rather as a whole for the model. While the use of independent per-particle velocity states potentially helps in complicated tracking scenarios, in our experiments we were unable to observe any tangible benefit. Moreover, in order to avoid

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

instability in the tracking results we needed to significantly increase the number of particles for a given noise level. As such, we have chosen to use the "group-velocity", and leave it to future work to investigate the possibility of independent velocity states.

Motion is modeled using a constant velocity model in discrete time with a variable sampling period T, giving the dynamic model

$$\mathbf{x}_t = \mathbf{x}_{t-1} + T\mathbf{v}_{t-1} + \omega, \tag{5}$$

with noise vector ω assumed to be zero mean Gaussian with fixed covariance. Particle velocities are updated after weighting of individual particles using the measurement model, and are a weighted average of the change in position

$$\mathbf{v}_t = \frac{1}{TN}\sum_{j=1}^{N} w_j(\mathbf{x}_t^j - \mathbf{x}_{t-1}^j), \tag{6}$$

where w_j is the normalized weight for particle j.

2.3. Measurement Model

As points for the model are given in a model-centered frame of reference, $p^m = [x^m, y^m, z^m, 1]$, we must transform them to the world frame using a 3D affine transformation quaternion. This yields positions in the world frame for each of our η model points for a particular particle j:

$$\begin{bmatrix}\mathbf{p}_1^j\\ \mathbf{p}_2^j\\ \vdots\\ \mathbf{p}_\eta^j\end{bmatrix} \begin{bmatrix}[x_1, y_1, z_1, 1]^\mathsf{T}\\ [x_2, y_2, z_2, 1]^\mathsf{T}\\ \vdots\\ [x_\eta, y_\eta, z_\eta, 1]^\mathsf{T}\end{bmatrix} = diag(\mathbf{B}^j) \begin{bmatrix}[x_1^m, y_1^m, z_1^m, 1]^\mathsf{T}\\ [x_2^m, y_2^m, z_2^m, 1]^\mathsf{T}\\ \vdots\\ [x_\eta^m, y_\eta^m, z_\eta^m, 1]^\mathsf{T}\end{bmatrix}. \tag{7}$$

Once we have our transformed points, we then must establish correspondences between each particle's model points and a world point so that we can score how well a state matches the current world model observation. That is, for each transformed point $\mathbf{p}_{1\ldots\eta}^j$, we select corresponding point $\mathbf{p}^*$ in the observation which has minimal spatial distance.

To find these correspondences, we first compute a KD-tree of the spatial dimensions of the world model points. This allows us to efficiently search for the nearest correspondence for each transformed point. We create this tree for the world model rather than the transformed model (even though the former has more points) as there is only one world, but many particles and models. Computing it for the models would require a KD-tree for each particle in each model. Additionally, computing it for the world allows us to take advantage of sampling strategies which significantly reduce run-time complexity. Finally, using the world-model allows us to take advantage of the sequential octree first presented in [15], greatly improving performance in the case of full and partial occlusions.

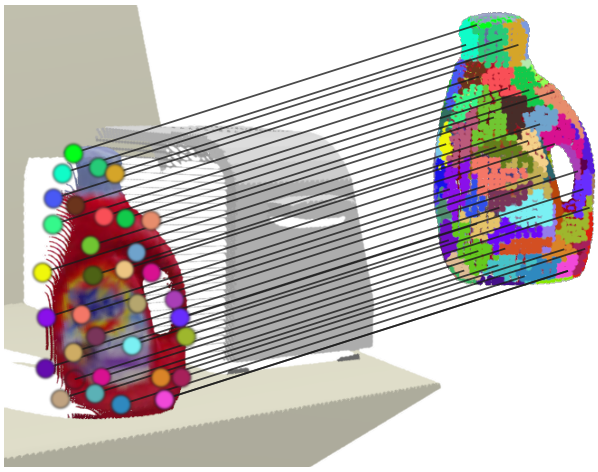


Figure 2. The model is divided into strata (shown as separate colors) by the supervoxel algorithm. Each particle independently selects a random sample (or samples) from each stratum for correspondence matching, and then searches for a correspondence for it in the observation.

Once we have selected (with replacement) an observed point correspondence for each model point, we must calculate a weight $\tilde{w}^j$ corresponding to the global similarity of the transformed points to the world observation. This is accomplished by summing the individual correspondence scores computed using weighted Euclidean distance in normalized world-spatial-, color-, and normal-space. In our experiments we set the weighting factors to have a 1:1:2 ratio (spatial:normal:color), as this balances the scoring between color and geometric shape, and found experimentally that it produced consistently good tracking results. The calculated particle weights $\tilde{w}^j$ are then normalized, and a final state estimate can be computed by taking the weighted average of all particles

$$\mathbf{x}_t = \sum_{j=1}^{N} w_j \mathbf{x}_t^j, \tag{8}$$

and the group-velocity can be computed using Equation 6.

3. Supervoxel Stratified Correspondence Sampling

While the tracking methodology discussed above works, in practice its run time performance is very poor, even for single objects. Moreover, speed of tracking is highly dependent on the size of object models as well as voxel resolution used. To address this, we propose a sampling scheme which selects a limited number of points from the model to transform and test. By doing this, we achieve linear asymptotic time complexity for the particle filter with respect to the number of particles - there is no dependence on the number

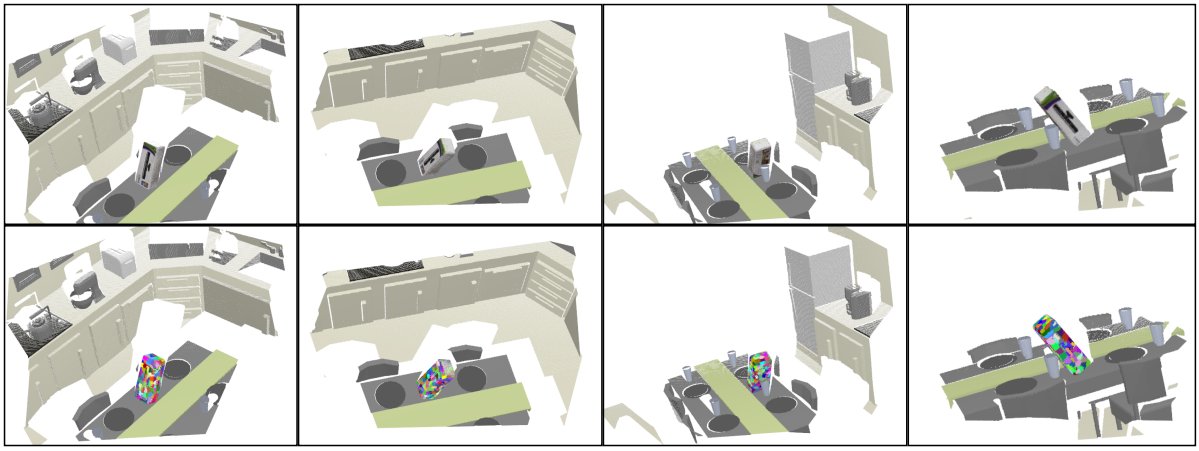


Figure 3. Tracking on the artificial "Kinect Box" sequence. The top row shows tracked output overlaid on input data, while the bottom row shows the supervoxl strata that are used for sampling.

of points in the models or the voxel resolution used. The only step which is dependent on the number of input points is the KD-tree construction, but this is only done once for the world model independent of the number of trackers, and is done as a pre-processing step regardless (for normal computation).

The proposed sampling scheme is as follows. We select a spatial sampling resolution R_{seed} based on the number of desired sample points per particle N_s. We then divide the model into strata, where each stratum is a supervoxel using the method of Papon et al.[14]. Supervoxels are a voxel-based surface patch representation that use connectivity, colors, and normals so that their edges conform well to object part boundaries. The strata are evenly divided over the spatial structure of the model, as seen in Figure 1. Additionally, using supervoxels as the strata ensures that we sample the important features of the models - for example in the model of Figure 1, we have a stratum for the brand logo, as well as ones for the concavities of the handle.

For each particle, we randomly select a point from each stratum using uniform sampling, and then transform and score it as described in the previous Section. As an additional step, we also select $\frac{N_s}{4}$ points uniformly from the entire model. Using strata reduces the noise which occurs when sampling from the whole model exclusively, while sampling randomly from the entire distribution improves occlusion performance.

While sampling will tend to produce noisier tracking results for low N_s, it also greatly reduces the computational complexity, as we only need to transform and test a small subset of the model points. This allows one to greatly increase the number of particles for a given frame-rate. Importantly, each particle is testing a separate random subset of model points. This results in the product of N_s, the number of sample points per particle, and N, the number of particles, reaching a critical level where coverage becomes sufficient that error is equivalent to sampling all points. In the results presented below, we shall demonstrate that this critical level can be used to significantly decrease run time for a given level of error. That is, we shall show that the number of points that must be tested overall, for a given level of error, is lower when stratified sampling is used. This means that we can significantly increase accuracy for a given frame-rate, reducing run-time complexity to the point that we can track 6 DoF pose for multiple objects in real-time.

4. Results

In this Section we first present results on a set of synthetic videos to quantify the effect of the stratified sampling, and compare results to a state of the art GPU particle filter [3]. We then present qualitative results on real videos in a robotic learning application, where we track multiple interacting targets with significant occlusions. In both synthetic and real cases, input consists of RGB-D sequences. Trackers were initialized using an external pose - in the synthetic case, from ground truth, and in the real case, using a pose estimation algorithm [2]. Object models were generated by registering multiple views of the objects using the same RGB-D sensor employed for tracking. All experiments were performed on a standard desktop computer (Intel i7 3.2Ghz), using four cores, and real data was obtained using a Kinect RGB-D camera.

4.1. Synthetic Data

In our first experiment, we demonstrate the effectiveness of our stratified sampling strategy using four synthetic tracking videos from [3]. These RGB-D sequences are set

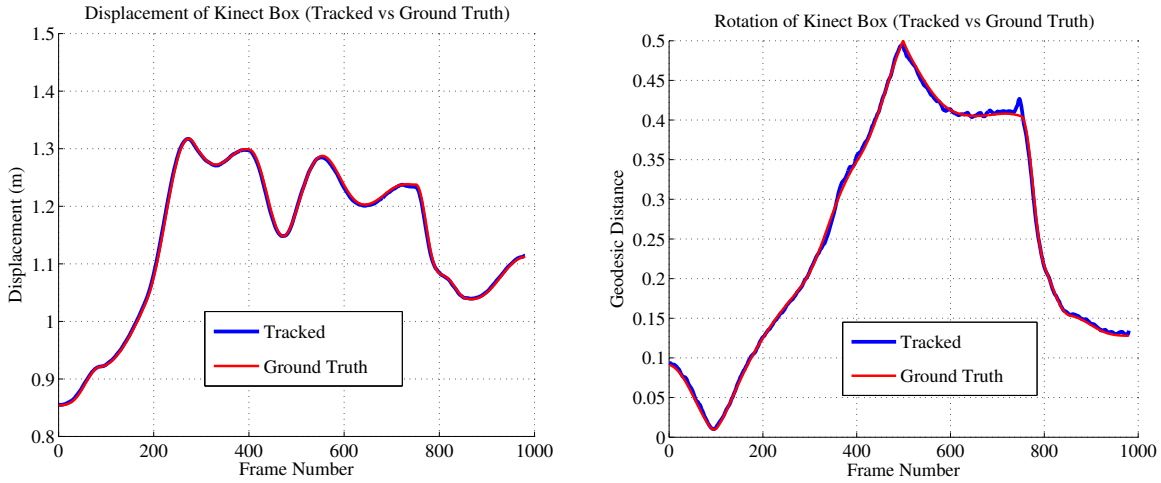


Figure 4. Displacement and rotation ground truth, with an example tracked result from a single run at $N_{samples} = 100$ and $N_{particles} = 1000$ (a frame rate of 20 fps).

in a virtual kitchen (see Figure 3) and each contain a single item to track as the camera moves. Ground truth trajectories of the cameras can be found in [3], but for our purposes it is sufficient to note that the trajectories are complex, consisting of large variations in position, orientation, and velocity.

To evaluate our approach, we compute root mean square (RMS) error in both translation and orientation, averaged over 25 test runs for each sequence. Computation times are measured in ms per frame, and are also averaged across all frames of the 25 test runs. In order to compare with [3], we have combined their RMS error results for each dimension (x, y, z, roll, pitch, yaw) into two measurements - displacement and rotation. Rotation is calculated using the unit quaternion distance metric [11], which is equivalent to the geodesic distance on the unit sphere. This combination reduces the amount of data to compare without loss, as the choice of orientation of the dimensions is arbitrary and without import. Example displacement and geodesic ground truths for the "Kinect Box" sequence can be found in Figure 4.

Timing results are given in Figure 5, showing results for the "Kinect Box" sequence (the most challenging of the four) scanning across number of particles and number of sample points. Plots for the other sequences can be found in the supplementary material and on the author's website. One can observe that, for a given level of sampling, the RMS error decreases for both displacement and geodesic as the number of particles increases. More importantly, it is also apparent that, for a given level of error, run-time per frame can be minimized by reducing the number of samples used and increasing the number of particles. Additionally, one can observe that RMS error appears to be asymptotic, with lower sampling levels approaching the asymptote at lower run-times.

We should also note that the minimum error asymptote observed is likely a consequence of the sampling resolution of the synthetic Kinect camera. For example, in the "Kinect Box" sequence, average distance to neighboring points (8-neighborhood) on the tracked box surface is 3.3 mm. This corresponds almost exactly to our observed error asymptote. This can be observed in all four sequences - our minimal error corresponds closely to the average point to point resolution of the observations on the model.

Our performance compares favorably to the results of Choi and Christensen [3] - for a given level of error, we achieve per-frame run times that are between half and a tenth of their published results. Additionally, we consistently reach the error asymptote, at considerably lower run times. We should also note that the highest sampling level shown corresponds to a complete sampling of the model, and is equivalent to the baseline PCL implementation, although we have made some slight modifications to the resampling and dynamic model which improve results. As can be seen, we are at least an order of magnitude faster than this base implementation.

4.2. Real Sequences

One application of our tracker is to provide semantic understanding and imitation of assembly tasks. This can be accomplished by tracking all interacting parts of an assembly as a human demonstrates, and then using the trajectories and poses in order to train a robot to replicate the construction. Additionally, the tracked output can be used as an input for the robot during construction in order to verify that

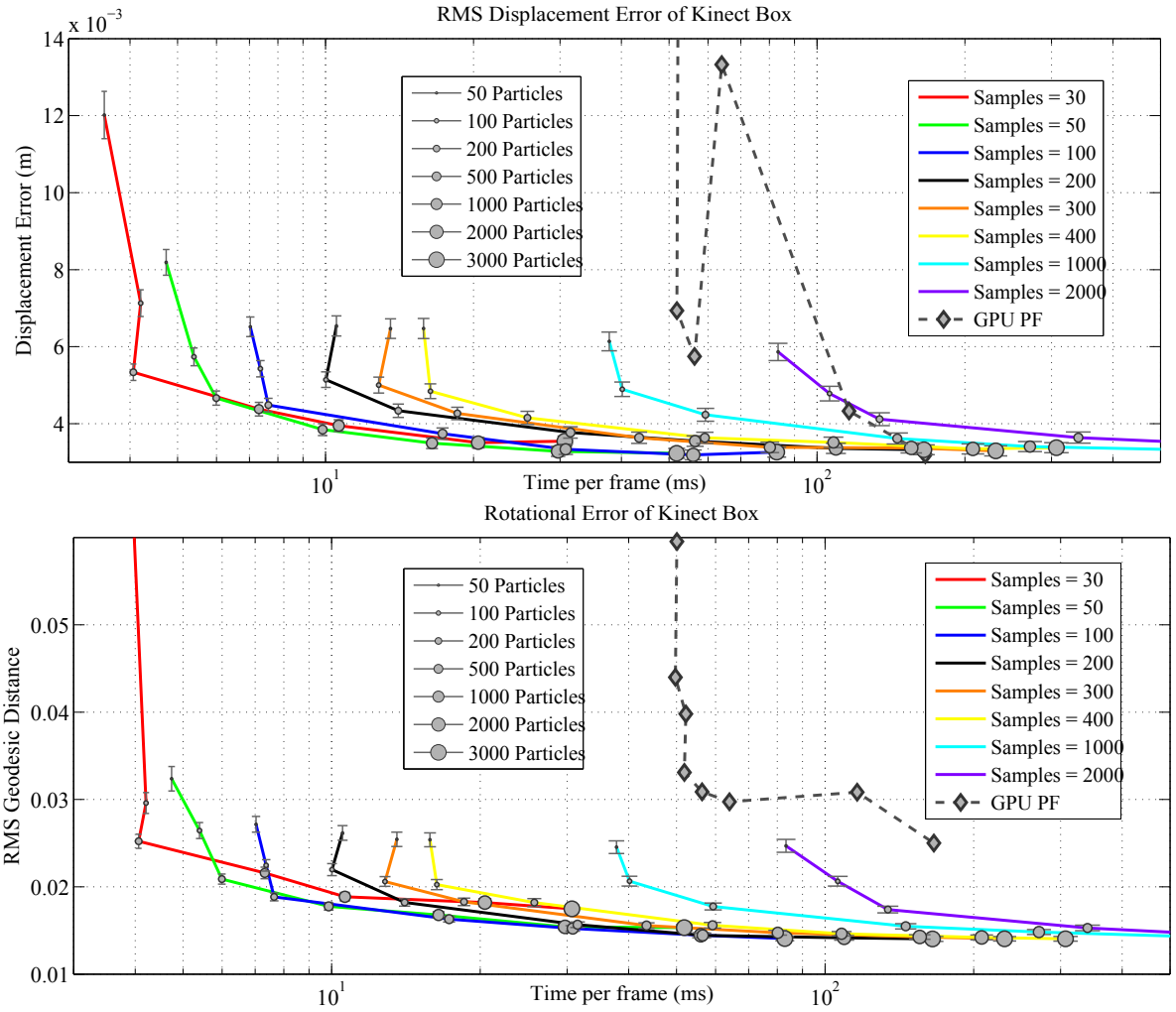


Figure 5. Results on the Kinect Box artificial sequence. Each colored curve represents a certain number of samples, and gives mean RMS error averaged over 25 trial runs for increasing numbers of particles (shown by increasing circle sizes). It is clear that using more samples and using more particles tends to decrease error. The black dotted lines give the results of the GPU method of Choi and Christensen [3].

it has successfully completed each step of the task.

As a demonstration of this, we use the well established "Cranfield" benchmark set [4]. This set consists of eight pieces which can be assembled in a number of different orders. In our experiments, models consist of voxelized point clouds derived from high-resolution models of the pieces, and initial poses for tracking are found using a combined object recognition and pose estimation algorithm [2]. Each object is tracked using an independent particle filter, with $N_{samples}$ set to 50, and $N_{particles}$ set to 1000.

Figure 6 shows a montage of screenshots captured as a human demonstrates assembly of the benchmark. As can be seen, all pieces are successfully tracked from start to finish, with each tracker outputting smooth trajectories that can be used for training a robot using Dynamic Motion Primitives (DMP) [12]. In Figure 7 we show tracks from multiple different human demonstrations - one can observe the different strategies that people employ in assembling the benchmark. The tracks in the lower right corner of the Figure are from a robot reproducing the assembly after being trained on the human demonstrations [16].

5. Conclusion

In this paper we have presented a novel spatially stratified sampling approach which greatly reduces the computational complexity of 3D Point Cloud particle filters. We evaluated the tracker using synthetic sequences for which precise ground truth exists, as well as real sequences of

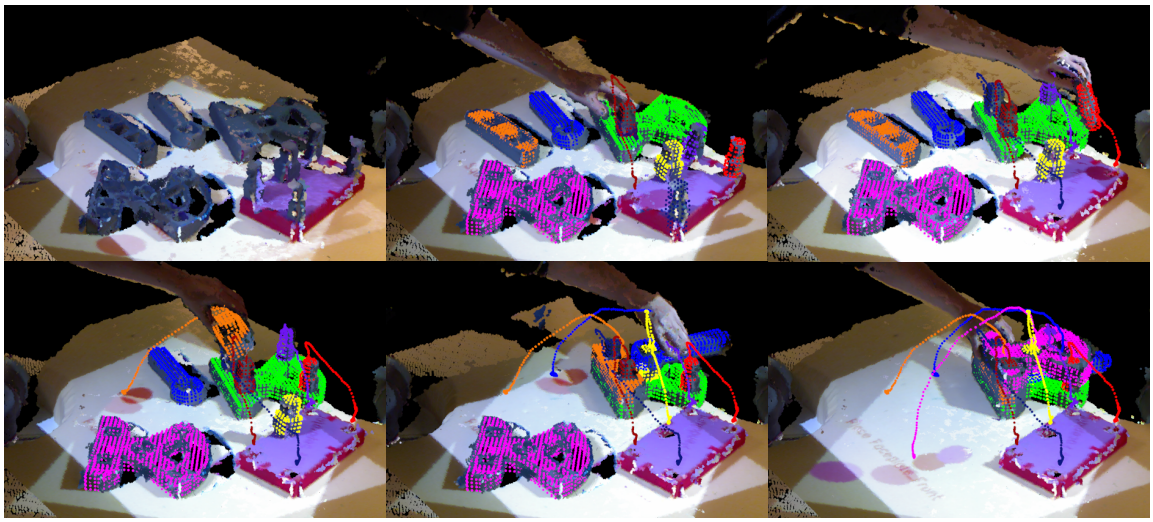


Figure 6. Human demonstration of assembly of the Cranfield Scenario on data from two fused RGB-D cameras. Tracking runs live for all objects at once at sufficient frame rates to track the whole task. Extracted trajectories are shown as traces.

a robot-teaching application. To demonstrate the effect of stratified sampling on performance, we conducted a sweep over the parameter space of number of particles and samples. This sweep showed the clear effectiveness of the proposed method in matching and even out-performing a GPU implementation on the CPU.

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] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Robust tracking-by-detection using a detector confidence particle filter. In *Computer Vision, 2009 IEEE 12th International Conference on*, Sept 2009. 1

[2] A. Buch, Y. Yang, N. Krger, and H. Petersen. In search of inliers: 3d correspondence by local and global voting. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, June 2014. 4, 6

[3] C. Choi and H. Christensen. Rgb-d object tracking: A particle filter approach on gpu. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013. 1, 4, 5, 6

[4] K. Collins, A. J. Palmer, and K. Rathmill. The development of a benchmark for the comparison of assembly robot programming systems. In *Robot technology and applications*, 1985. 6

[5] A. Doucet, N. De Freitas, and N. Gordon, editors. *Sequential Monte Carlo methods in practice*. 2001. 2

[6] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Multi-target tracking using joint probabilistic data association. In *Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on*, Dec 1980. 1

[7] C. Hue, J.-P. Le Cadre, and P. Perez. Tracking multiple objects with particle filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):791 – 812, jul 2002. 1

[8] M. Isard and A. Blake. Condensation conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998. 1

[9] Z. Khan, T. Balch, and F. Dellaert. Mcmc-based particle filtering for tracking a variable number of interacting targets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(11):1805–1819, Nov 2005. 1

[10] S. Koo, D. Lee, and D.-S. Kwon. Multiple object tracking using an rgb-d camera by hierarchical spatiotemporal data association. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013. 1

[11] J. Kuffner. Effective sampling and distance metrics for 3d rigid body path planning. In *Robotics and Automation (ICRA) 2004. IEEE International Conference on*, April 2004. 5

[12] 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.*, 28(1):145–157, 2012. 6

[13] O. Lanz. Approximate bayesian multibody tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(9):1436–1449, Sept 2006. 1

[14] 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. 4

[15] J. Papon, T. Kulvicius, E. E. Aksoy, and F. Wörgötter. Point cloud video object segmentation using a persistent su-

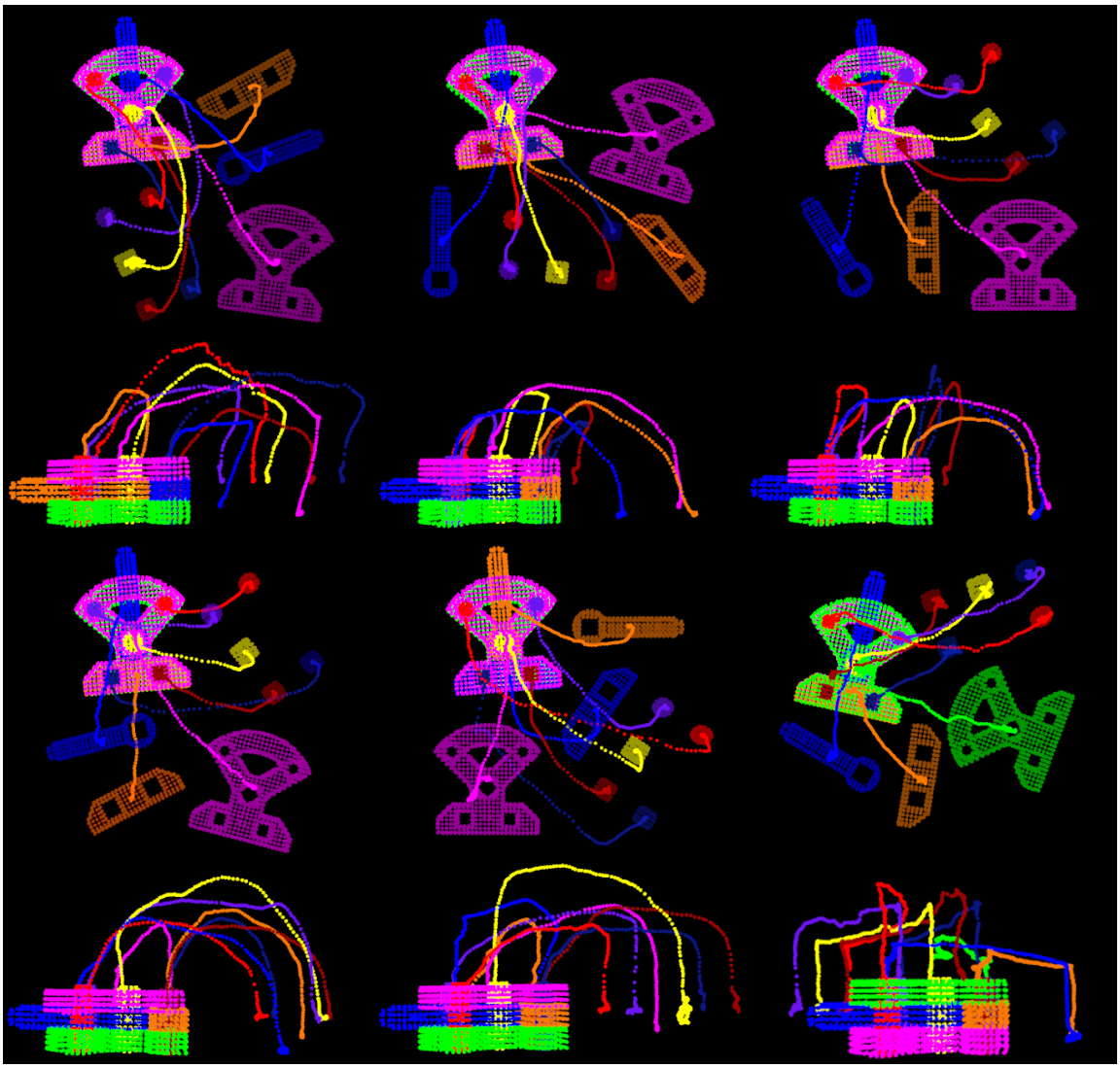


Figure 7. Tracking results from six different recordings of the Cranfield Scenario. The tracks in the bottom right corner are from the robot constructing the object, while the other five are from five different human demonstrators. In the overhead views, starting poses are shown (in slightly darker colors) for the objects.

pervoxel world-model. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013. 3

[16] J. Rossmann, N. Wantia, E. E. Aksoy, S. Haller, and J. Piater. Active learning of manipulation sequences. In *Robotics and Automation (ICRA) 2014. IEEE International Conference on*, 2014. 6

[17] D. Schulz, W. Burgard, D. Fox, and A. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Robotics and Automation (ICRA) 2001. IEEE International Conference on*, 2001. 1

[18] J. Vermaak, A. Doucet, and P. Perez. Maintaining multimodality through mixture tracking. In *Computer Vision 2003. IEEE International Conference on*, Oct 2003. 1

[19] J. Vermaak, S. Godsill, and P. Perez. Monte carlo filtering for multi target tracking and data association. *IEEE Transactions on Aerospace and Electronic Systems*, 41(1):309 – 332, jan. 2005. 1

[20] B.-N. Vo, S. Singh, and A. Doucet. Sequential monte carlo methods for multitarget filtering with random finite sets. *IEEE Transactions on Aerospace and Electronic Systems,*, 41(4):1224 – 1245, oct. 2005. 1, 2

C U R R I C U L U M V I T A E

PERSONAL DATA

Family name:	Schoeler
First name:	Jan Markus
Nationality:	German

HIGHER EDUCATION

2012 – present — Georg-August-Universität Göttingen, Germany
- **Work-group:** Prof. Wörgötter
- **Institute:** Drittes physikalisches Institut
- **Area of Research:** Computer Vision and Machine Learning
- **Topics:** Object Recognition, Segmentation, Deep Learning

2010: — Friedrich-Schiller-Universität Jena, Germany
- **Diploma in physics (M.S. equivalent)**
- **Grade:** 1.1

2009 – 2010: — Fraunhofer Institut Jena (IOF), Germany
- **Diploma thesis**
- **Area of research: Functional properties of coated glass samples**

2007 – 2008: — Scholarship for the Universtity of New Mexico, Albuquerque, USA
- **Studies in quantum cryptography and laser physics**

2004 – 2009: — Julius-Maximilians-Universität Würzburg, Germany
- **Diploma studies in physics**

EDUCATION

1996 – 2004: — König-Heinrich-Schule Fritzlar
- **Abitur (general qualification for university entrance)**

RESEARCH EXPERIENCE

Topics:	Computer Vision, Object Classification, Segmentation, Deep learning
Programming:	C/C++, Matlab, Python, Delphi, Pascal
Version Control:	Git, SVN, Github
Libraries:	OpenCV, Point Cloud Library (PCL), Robot Operating System (ROS), Boost, NumPy, SciPy, Theano, Vlfeat, OpenMP, POSIX Threads, MPI
Others:	Linux, Latex, Lynx, OpenSCAD, Microsoft Office, OpenOffice, Inkscape, Gimp, OpenShot

PUBLICATIONS (PEER-REVIEWED)

Papon, J. and **Schoeler, M.** (2015)
Semantic Pose using Deep Networks Trained on Synthetic RGB-D. International Conference on Computer Vision (ICCV), accepted.

Schoeler, M. and Wörgöttter, F. (2015)
Bootstrapping the Semantics of Tools: Affordance analysis of real world objects on a per-part basis. IEEE Transactions on Autonomous Mental Development (TAMD), submitted.

Schoeler, M. and Papon, J. and Wörgötter, F. (2015)
Constrained Planar Cuts – Object Partitioning for Point Clouds. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Schoeler, M. and Wörgötter, F. and Papon, J. and Kulvicius, T. (2015)
Unsupervised generation of context-relevant training-sets for visual object recognition employing multilinguality. IEEE Winter Conference on Applications of Computer Vision (WACV).

Papon, J. and **Schoeler, M.** and Wörgötter, F. (2015)
Spatially Stratified Correspondence Sampling for Real-Time Point Cloud Tracking. IEEE Winter Conference on Applications of Computer Vision (WACV).

Aksoy, E. E. and **Schoeler, M.** and Wörgötter, F. (2014)
Testing piagets ideas on robots: Assimilation and accommodation using the semantics of actions. IEEE International Conferences on Development and Learning and Epigenetic Robotics (ICDL-Epirob).

Schoeler, M. and Wörgötter, F. and Aein, M. and Kulvicius, T. (2014)
Automated generation of training sets for object recognition in robotic applications. 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD).

Schoeler, M. and Stein, S. and Papon, J. and Abramov, A. and Wörgötter, F. (2014)
Fast Self-supervised On-line Training for Object Recognition Specifically for Robotic Applications. International Conference on Computer Vision Theory and Applications (VISAPP).

Stein, S. and Wörgötter, F. and **Schoeler, M.** and Papon, J. and Kulvicius, T. (2014)
Convexity based object partitioning for robot applications. IEEE International Conference on Robotics and Automation (ICRA).

Stein, S. and **Schoeler, M.** and Papon, J. and Wörgötter, F. (2014)
Object Partitioning using Local Convexity. Conference on Computer Vision and Pattern Recognition (CVPR).

Papon, J. and Abramov, A. and **Schoeler, M.** and Wörgötter, F. (2013)
Voxel Cloud Connectivity Segmentation – Supervoxels for Point Clouds. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

RESEARCH PROJECTS

2015 – present: MOSES project for Samsung Electronics Co., Ltd.
Senior research scientist

2012 – present: EU project Xperience (FP7-ICT-270273)
Research scientist

2013 – present: EU project ACAT (FP7/2007-2013)
Research scientist

OPEN SOURCE PROJECTS

2014 – present Point Cloud Library
Contributor

2014: Google Summer of Code for the Point Cloud Library
Participant
Project: Automated benchmark generation for object-part segmentation

2012: Oculus Vision System
Contributor

2012: OpenCV Library
Contributor