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

Deliverable Number:	D2.3.3
Deliverable Title:	Transfer of Affordances and Categories: Technical report or scientific publication on how to use the developed representations of affordances and categories within the architecture and in the final demonstration
Type (Internal, Restricted, Public):	PU
Authors:	D. Kraft, M. T. Thomsen, W. Mustafa, N. Krüger, H. Xiong, J. Piater, S. Szedmak, B. Ridge, A. Ude, M. Schoeler, F. Wörgötter
Contributing Partners:	SDU, UIBK, JSI, UGOE

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

Contents

Chapter 1

Executive Summary

1.1 Role Within the Description of Work

This Deliverable is part of Work Package 2.3 *Affordances, Object-Action Complexes and Categories*. Quoting from the Description of Work Package 2.3:

> **Objectives:** In Xperience the object affordances need to be rich enough to allow for sophisticated action sequencing required for the structural bootstrapping in WP3 and WP4. The central objective of this WP is to address the generation (by learning), understanding and transfer (to planning) of object affordances. This leads over to the required learning of higher level entities (action rules) and finally to the learning of objectaction categories.

Within this Work Package, this Deliverable reports in particular on the below tasks:

Task 2.3.1 Learning object affordances by means of semantic sensorimotor categories (OACs)

Task 2.3.3 Exploration based learning of object categories

Given the nature of this document we will report on the scientific progress within WP2.3 (in Section 2) in Year 4but also on how these results will and/or could be transferred to the two integration scenarios in the project (Section 3).

1.2 Links to Other Work Packages

WP 2.3 draws together work from WP 2.1 *Sensorimotor Experience* and WP 2.2 *Motor Actions*. Moreover, it constitutes a bridge to WP 3.1 *Structural Bootstrapping on Sensorimotor Experience*, as category formation is an essential building block of structural bootstrapping.

1.3 Outline of Results

This report presents work in year four within the Xperience project on the acquisition and modelling of categories. We limit this report to concise overviews (Chapter 2) referring to the actual content in publications attached to the report.

Section 2.1 describes work on the analysis of *action-grounded features*, where action-grounded 3-D shape features are compared to non-grounded 3-D shape features features in an object affordance categorisation setting.

In Section 2.2 we investigate the learning of undirected graphical models and in particular its application in finding early visual features. Several novel learning methods are introduced and studied, with both theoretic analysis and empirical comparisons to other existing methods. Because of the generality of the study, it can be also applicable to other tasks, *e.g.* visual segmentation, multi-label prediction.

Section 2.3 gives an overview over work that allows to learn visual categories (geometry as well as appearance based) in a unsupervised scheme. An hierarchical category learning approach is used. We expect this work to be used as a basis for affordance transfer based on visual properties (see Section 3.2).

We have extended our previous work on grasping of unmodelled objects based on visual information and previously acquired ECVxPMF statistics (vision, action cross space). This work is presented in Section 2.4 and the transfer to the Xperience scenarios is discussed in Section 3.3.

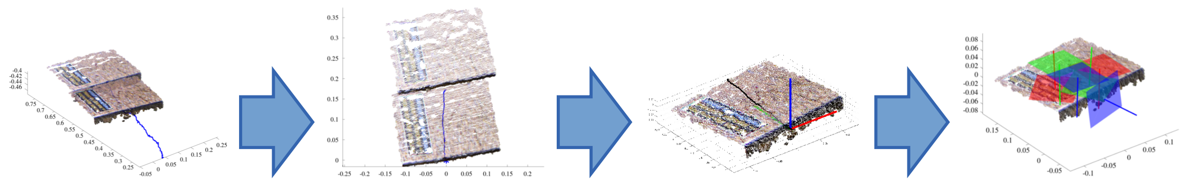
Section 2.5 summarizes work for the decomposition of objects into functional parts as well as the part and part-to-part description in order to automatically bootstrap one or multiple affordances for an unknown object.

Chapter 2

Description of Scientific Results

2.1 Action-Grounded 3-D Shape Feature Analysis

In our work for the previous year [1] on the subject of bootstrapping object categories via push affordances, one of the enabling concepts that was developed was that of *action-grounded features*, that is, object features that are defined dynamically with respect to manipulation actions. Rather than using pose-invariant visual features, as is often the case with object recognition, such features are grounded with respect to the manipulation of the object, for instance, by using shape features that describe the surface of an object relative to the push contact point and direction, as was the case in our previous work [1]. An example of how such features might be extracted is shown in Figure 2.1.



(a) Object point clouds before/after interaction + hand push trajectory

(b) Point clouds + trajectory transformed to push action coordinate frame.

(c) Pre-push point cloud action coordinate frame grounding.

(d) Fitting planes to pre-push object cloud parts, grounded with respect to action.

Figure 2.1: Action-grounded shape feature extraction pipeline.

Leading on from that work, in a paper submitted this past year [RUU15], we shifted the focus from the bootstrapping aspect and decided to analyse the action-grounded features themselves. Thus, we provided an experimental comparison between action-grounded features and non-grounded features in an object affordance classification setting using an experimental platform that gathers 3-D data from the Kinect RGB-D sensor, as well as push action trajectories from an electromagnetic tracking system. Experimental results, some of which are shown in Table 2.1, were provided that demonstrate the effectiveness of this action-grounded approach across a range of state-of-the-art classifiers.

Table 2.1: Mean 10-Fold Cross-Validation Results: Test Sets

	Non-Grounded	Action-Grounded
GRLVQ	87.3 ±10%	92.6 ±8%
SRNG	88.3 ±9%	93.0 ±7%
SVM	82.5 ±10%	86.2 ±9%
MLR	74.6 ±12%	78.1 ±13%
RF	86.0 ±10%	95.8 ±6%

2.2 Novel Learning Methods for Early Visual Features

Good visual features can be of significance to visual understanding, in both 2D and 3D cases, by providing more informative bases to higher-level processes, *e.g.* classification or clustering. In [XSRSP14], [XSP14a] and [XSP14b], we developed some novel training methods of undirected graphical models (UGMs) for extracting low-level visual features. More concretely, restricted Boltzmann machines (RBMs), which are fundamental building blocks of deep learning, are used for modeling the statistics of images in certain domain by finding latent bases. An RBM is a two-layer, bipartite neural network. It is a *restricted* version of the Boltzmann machine with interconnections only between hidden layers and visible layers. Input data are binary and N_v-dimensional. They are fed into N_v units in the visible layer $\mathbf{v}$. The N_h units in the hidden layer $\mathbf{h}$ are stochastic binary variables, *i.e.* $\mathbf{v} \in \{0,1\}^{N_v}$, $\mathbf{h} \in \{0,1\}^{N_h}$. The joint probability of $\{\mathbf{v}, \mathbf{h}\}$ is

$$p(\mathbf{v},\mathbf{h}) = \frac{1}{\mathbf{Z}} \exp(-E(\mathbf{v},\mathbf{h})) \qquad E(\mathbf{v},\mathbf{h}) = -\mathbf{v}^\top \mathbf{W}\mathbf{h} \tag{2.1}$$

where $\mathbf{W} \in \mathbb{R}^{N_v \times N_h}$ is the matrix of symmetric weights, and $\mathbf{Z} = \sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}))$ is the partition function for normalization. Because of the restricted connections in RBMs, the hidden units h_j are independent of each other conditioned on the visible data $\mathbf{v}$. Given training data $\mathcal{D} = \{\mathbf{v}^{(l)}\}_{l=1}^{L}$, an RBM can be learned by maximizing the average log-likelihood of $\mathcal{D}$:

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} \mathcal{L}(\mathcal{D}) = \arg\max_{\mathbf{W}} \frac{1}{L} \sum_{l=1}^{L} \left(\log \sum_{\mathbf{h}} p(\mathbf{v}^l, \mathbf{h}) \right) \tag{2.2}$$

Based on (2.1), the gradient of $\mathcal{L}(\mathcal{D})$ is computed as

$$\nabla \mathcal{L}(\mathcal{D}) = \frac{1}{L} \sum_{l=1}^{L} \left[\mathbb{E}_{\mathbf{v}^{(l)} \in \mathbf{V}, \mathbf{h} \sim p(\mathbf{h}|\mathbf{v}^{(l)})} (\mathbf{v}^{(l)} \mathbf{h}^\top) - \mathbb{E}_{\mathbf{v},\mathbf{h} \sim p(\mathbf{v},\mathbf{h})} (\mathbf{v}\mathbf{h}^\top) \right], \tag{2.3}$$

where $\mathbb{E}_p(\cdot)$ denotes the expected value with respect to p. Clearly, the sampling $\mathbf{v}, \mathbf{h} \sim p(\mathbf{v},\mathbf{h})$ makes learning practically infeasible because it requires a large number of Markov chain Monte Carlo (MCMC) iterations to reach equilibrium. Therefore, there are various approximation methods to estimate the second term in (2.3), *e.g.* MCMC maximum likelihood (MCMCML), (persistent) contrastive divergence (PCD), tempered transition and parallel tempering, just to name a few.

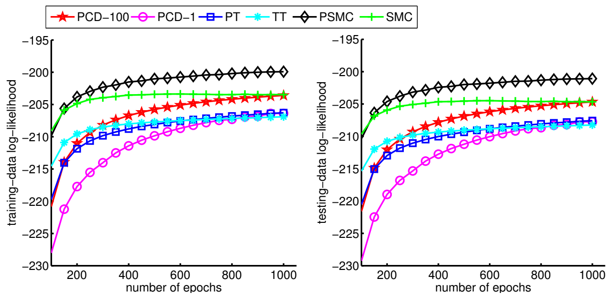
2.2.1 Persistent Sequential Monte Carlo: A Closer Approximation of Maximum Likelihood Learning

A new learning method, persistent sequential Monte Carlo (PSMC), for general undirected graphical models (UGMs, thus also for RBMs) was proposed in [XSP14b]. Most existing learning methods for UGMs can be considered as special cases of Robbins-Monro's stochastic approximation procedure (SAP) with different Metropolis transitions (*e.g.* PCD is a SAP with Gibbs transitions). By linking SAP and sequential Monte Carlo (SMC), we cast PCD and other state-of-the-art learning algorithms into a SMC-based interpretation framework. Moreover, within the SMC-based interpretation, two key factors which affect the performance of learning algorithms are disclosed: *learning rate* and *model complexity.* Based on this rationale, the strengths and limitations of different learning algorithms can be analyzed and understood in a new light.

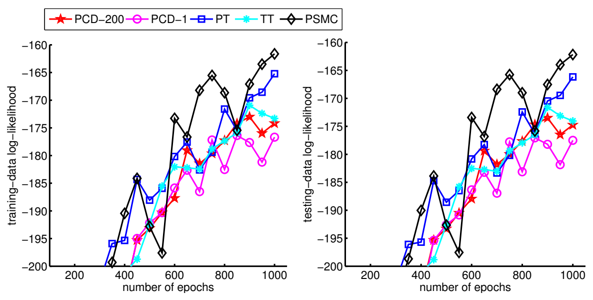
Inspired by the understanding of learning UGMs from a SMC perspective, and the successes of global tempering used in parallel tempering and tempered transition, we put forward PSMC to approach the maximum likelihood solution in learning UGMs. The basic idea is to construct a long, persistent distribution sequence by inserting many tempered intermediary distributions between two successively updated distributions. According to our empirical results on learning several UGMs, the proposed PSMC outperforms other learning algorithms in challenging circumstances, *i.e.* large learning rates or large-scale models.

In particular, we tested PSMC on a hand-written digit image database MNIST[1]. Two RBM models (one with 10 hidden nodes and the other with 500 hidden nodes) were constructed. Our empirical results (Figure 2.2) show that PSMC outperforms other state-of-the-art learning methods by yielding higher log-likelihoods.

[1]http://yann.lecun.com/exdb/mnist/index.html



(a)



(b)

Figure 2.2: The performance comparison of PSMC and other algorithms on two RBMs. (a): with 10 hidden nodes; (b) with 500 hidden nodes. Check [XSP14b] for details about the comparison.

2.2.2 Training with binary Hamiltonian Monte Carlo

In [XSP14a], we conducted an empirical study of the Robbins-Monro's stochastic approximation procedure (SAP) with an alternative Metropolis transition: binary Hamiltonian Monte Carlo. Hamiltonian Monte Carlo (HMC) is a Metropolis algorithm with a proposal distribution analogous to Hamiltonian dynamics. Compared to the random walk as used in the standard Metropolis algorithm, HMC can propose a distant jump while still preserving a high acceptance rate. Suppose that we are interested in sampling from $p(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^D$. An auxiliary variable $\mathbf{q} \in \mathbb{R}^D$ with $\mathbf{q} \sim \mathcal{N}(\mathbf{q}; \mathbf{0}, \mathbf{M})$ is introduced (usually $\mathbf{M} = c \cdot \mathbf{I}_D$). A Hamiltonian function can be constructed as

$$\mathcal{H}(\mathbf{x}, \mathbf{q}) = U(\mathbf{x}) + K(\mathbf{q}) \tag{2.4}$$

where $U(\mathbf{x})$ and $K(\mathbf{q})$ are negative logarithms of $p(\mathbf{x})$ and $p(\mathbf{q})$. The changes of $\mathbf{x}$ and $\mathbf{q}$ over time ν are

$$\dot{\mathbf{x}}(\nu) = \frac{\partial \mathcal{H}}{\partial \mathbf{q}(\nu)} = \mathbf{M}^{-1}\mathbf{q}(\nu), \quad \dot{\mathbf{q}}(\nu) = -\frac{\partial \mathcal{H}}{\partial \mathbf{x}(\nu)} = -\frac{dU(\mathbf{x})}{d\mathbf{x}(\nu)}. \tag{2.5}$$

HMC can yield more effective sampling by making use of gradient information of the target distribution's density function. We can also see, from (2.5), that HMC can only be applied on continuous distributions of which the partial derivatives of the log density function can be computed. Therefore, applying HMC to samples from a RBM is not straightforward. We applied binary HMC (bHMC), a novel sampling strategy for arbitrary binary distributions, for our purpose of learning. In [XSP14a], we found that bHMC resembles Gibbs sampling but with a different acceptance criterion. To explore its practical applicabilities, we compared it against Gibbs sampling in SAP on training a toy Boltzmann machine ($D = 10$). Our empirical results suggest that, unfortunately, the SAP with bHMC is inferior to the one with Gibbs, i.e. PCD.

2.2.3 Training with Inhibition

The maximum-likelihood training criterion sometimes is not enough if some prior knowledge on the model is available. Usually, maximum a-posteriori (MAP) is applied to integrate the prior. For example, by considering RBM as a neural network (hidden nodes being neurons), it may be desirable to increase its biological plausibility by achieving some extra properties as well as maximize likelihood. In [XSRSP14], we exploited how Bayesian learning of a restricted Boltzmann machine (RBM) can discover more biologically-plausible early visual features. The study is mainly motivated by the sparsity and selectivity of visual neurons' activations in area V1. Most previous work in computational modeling emphasizes selectivity and sparsity independently, which neglects the underlying connections between them. In [XSRSP14], a prior on parameters is defined to simultaneously enhance these two properties, and a Bayesian learning framework of RBM is introduced to infer the maximum posterior of the parameters. The proposed prior acts as lateral inhibition between neurons. According to our empirical results, the visual features learned from the proposed Bayesian framework yield better discriminative and generalization capability than those learned with maximum likelihood or other state-of-the-art training strategies (see features in Figure 2.3).

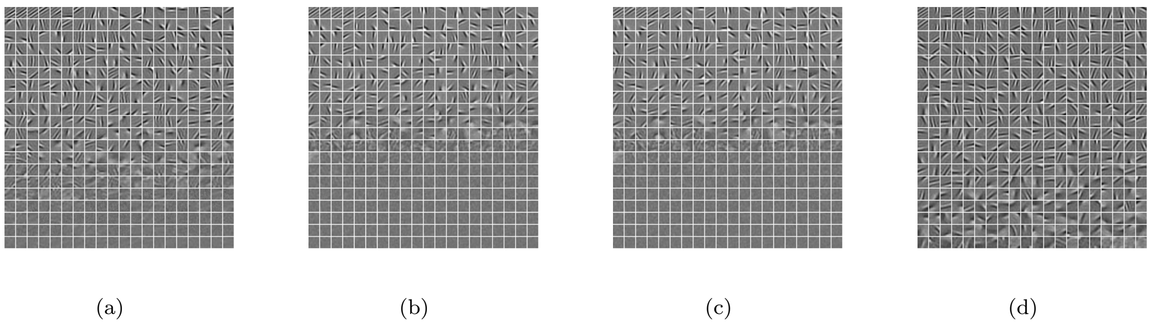


Figure 2.3: The receptive fields of neurons learned with (a) the contrastive divergence (CD) algorithm, (b) sparse CD, (c) selective CD, and (d) our Bayesian strategy. See [XSRSP14] for more details.

2.3 Extracting visual categories by hierarchical clustering

In [MKK15] we introduce an object categorization system which uses hierarchical clustering to extract categories. The system is able to assign multiple, nested categories for unseen objects. In our system (Figure 2.4), objects are represented with global pair-wise relations computed from 3D features extracted by three RGB-D sensors. Hierarchical clustering allows similar categories to be nested within larger clusters forming a structure in which more generic categories are found on top of less generic ones (see Figure 2.5).

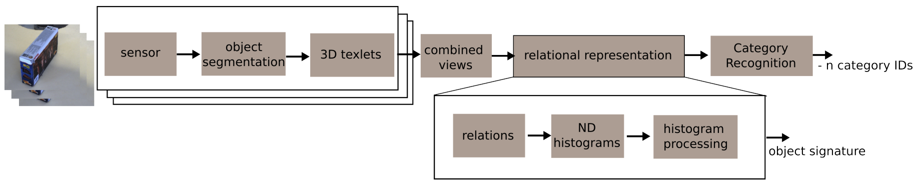


Figure 2.4: Object categorization system.

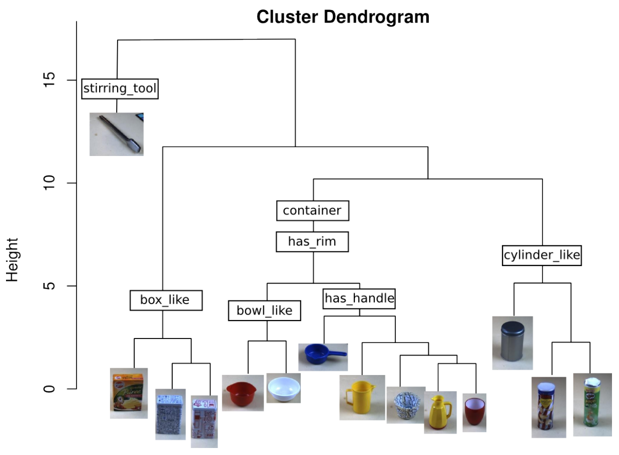


Figure 2.5: Examples of hierarchical clustering.

We show that our system outperforms a state-of-the-art approach particularly when only a few training samples are used. To evaluate the system, we hand-labeled our benchmark object set with a number of visual categories. We use the labeled categories from the training subset to find the best matching ones in the hierarchy. Then, we evaluate the performance of the system on the test subset.

2.4 Identifying relevant feature-action associations for grasping unmodelled objects

In [TKK15], we investigated the space of visual triggered action affordances in terms of a combined feature-action space. We spatially combine visual features with actions, both from a simulated environment in terms of simulated visual sensors and simulated grasps. We vary important dimensions in the visual space such as the granularity (size) of the surface patches, the order (one or two combined features) and the feature abstraction (a surface patch, a surface patch with a border label and a surface patch with border label and direction to the border). In our search for promising regions (visual triggered affordances) in the space, we perform a neighbourhood analysis and extract the success-probability and the amount of support of the different feature-action particles and save the results in a database. This database enables grasp predictions on previously unseen objects. Based on the method, we investigate

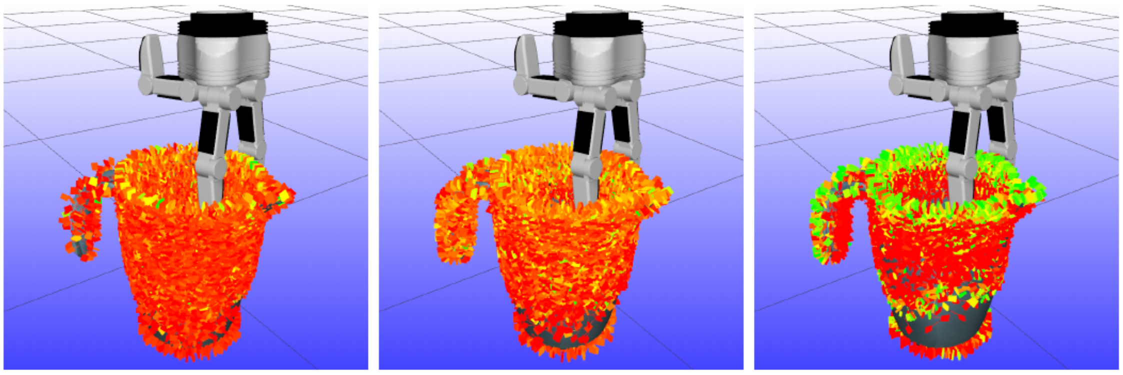


Figure 2.6: Visualisation of the grasp predictions for a pitcher object with feature relations of different order and with different semantic. The colour depict the predicted likelihood for success. Green meaning a success likelihood of 1.0 and red meaning a success likelihood of 0.0.

the visual action space as described above and show how the ability to make grasp predictions is strongly depending on the visual descriptor used. In particular it is seen how the addition of a border label and direction improve the performance significantly for specific object categories such as bowls and cups, essentially extracting parts of the open structure properties that these objects poses. This property is visualised in Figure 2.6, where the ability to predict grasps at the edge of an open structure, here a pitcher, are clearly differentiated by the introduction of the border label with direction to the border (Figure 2.6c) showing significant better predictions as compared to the result without border and direction (Figure 2.6a).

Note that [TKK15] is an extended version of [3] introduced in D2.1.2. Introduction and state of the art have been extended extensively. Furthermore additional simulated experiments have been performed in a scenario with gravity acting upon the objects to justify the use of a simplified simulation environment for the main results.

2.5 Part-based affordance analysis

Due to the vast variety of objects, which show similar affordance (e.g. containing substances, cutting other objects) we developed a framework, which introduces a part-based object representation, were each part fulfills a specific and much more restricted function (e.g. the blade and the handle of a knife). Additionally, this approach reduces variance among objects with the same affordance as functionality often depends on the existence of certain parts and in addition their respective alignment to each other. The framework consists of two building blocks: First the detection of parts in an object and second the description of objects using found parts to allow the classification of new objects to learned functionalities. For detecting parts we developed a purely data-driven Constrained-Planar-Cuts (*CPC*) algorithm in [SPW15]. It uses concavities in objects (similar to [2]) as evidence to propose a hierarchical object-decomposition using planar cuts found by a novel weighted and directed RANSAC scheme. Being a bottom-up method allows it to be applied to a huge variety of different objects without training and respective ground-truth as we show in the paper. Moreover we were able to show that the decomposition of the object mimics human decomposition very well (see Figure 2.7). This contribution provides a substantial improvement of our older method [2], which had also been based on concavities but without the cutting plane approach.

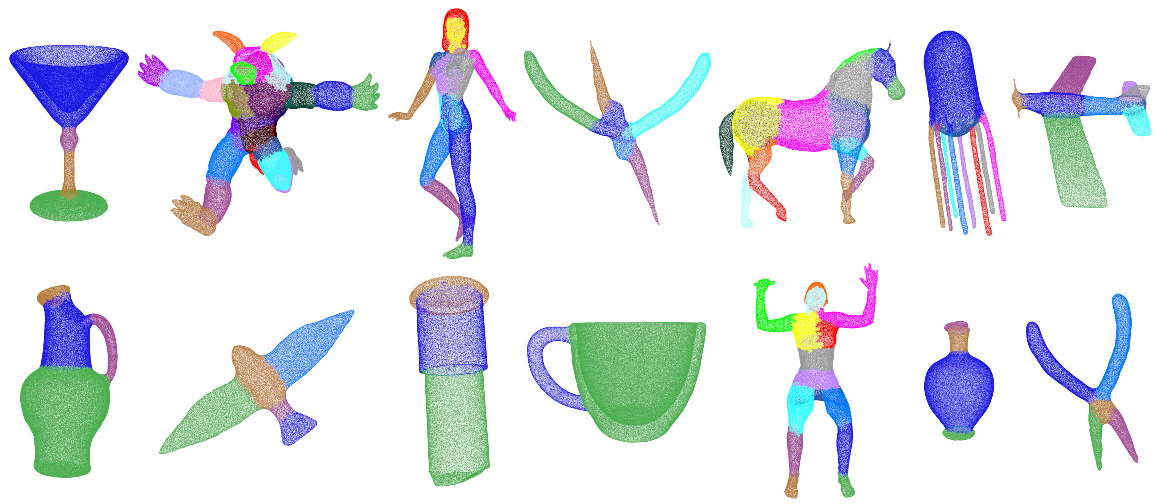


Figure 2.7: Unsupervised segmentations produced by the *CPC* algorithm using fixed parameters.

In the following, second step objects will be described by their part signatures (composition) as well as part-to-part relation signatures. The latter will contain information of how and where parts are attached to each other. Compared to full object signatures this representation is more fine-grained, which allows machine learning techniques to generalize in a more powerful way (e.g. an object for containing substances can have zero, one or multiple different handles attached and still will function as long as a container is present).

Besides assigning a single functionality to multiple objects, the system is also able to assign multiple functionalities or even makeshift solutions to one object when the part configuration allows it.

For example the left object in Figure 2.8 can be used for hitting with part A being the head and part B being the handle or for drilling holes with B being the tip and A being the handle. Although the hollow skull on the right side is different from any learned object the system can assign a makeshift functionality of contain to it.

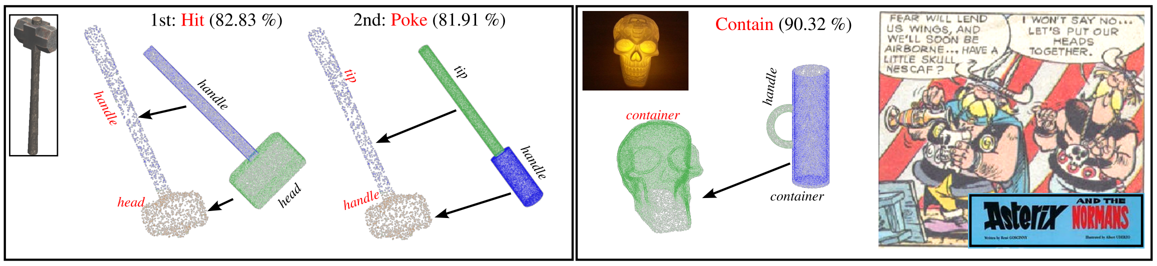


Figure 2.8: The system assigns two highly compatible functions to the hammer-like object on the left and the makeshift function of contain to the hollow skull on the right. Black labels: Parts of the most similar training objects. Red labels: Labels assigned by the system to the parts. Numbers in percent: Similarity of the new object to the learned objects based on parts and part-to-part relations.

Chapter 3

Transfer of Affordances and Categories for final implementation

3.1 Learning Undirected Graphical Models

Undirected graphical models constitute a very general and powerful framework for probabilistic modeling and inference, and our novel learning methods described in Section 2.2 (check attached articles [XSRSP14, XSP14a, XSP14b] for details) are widely applicable. Within Xperience, they can potentially be used for learning visual features. Moreover, in constructing ROAR (repository of objects with attributes and roles) for structural bootstrapping, a conditional random field can be constructed to model the function map from object attributes to roles, in which the conditional inter-role dependencies are modeled as a Boltzmann machine (see D3.1.3 for more details).

However, there are currently no concrete plans to integrate these methods into Xperience demonstrators.

3.2 Visual categories by hierarchical clustering

Our work on the creation of visual categories [MKK15] is able to determine categories for objects that we are able to perceive but do not actually know (e.g., an object not in our recognition database). As such it allows us to deal with objects that are, on the symbolic level, unknown to us. We will integrate this work with the repository of objects&attributes with roles (ROAR), see also D3.1.3.

The visual categorisation system described here will use a segmented point cloud (of an unknown object) to create a set of visual categories. These categories will be used by the ROAR to find objects that are visually similar to be able to suggest if the new, unknown object can be used to replace another object in the scene for a specific task. This replacement functionality will be integrated into both scenarios.

3.3 Grasping of unmodelled objects

The work on learning visual triggered grasp affordances for unmodelled objects introduced in [TKK15] will be transferred to parts of both scenarios.

In the “making a salad” scenario—in year four at the UGOE platform—the method is to be used for grasping unknown knives with the purpose of cutting. This has previously been shown in the platform at SDU but will now be shown integrated into the bigger scenario. By utilising a grasping and cut simulator for learning the affordances in a simulated environment and applying the learned affordances in a real scenario. Our system will be provided with a segmented point cloud of the knife (from the UGOE system) as an input and will in return make a prediction of a grasping point. This grasp point is then to be taken by the grasp execution component to actually execute the grasp. The grasp prediction module will be transferred to the UGOE platform and integrated with the segmentation component and a grasp execution component.

In the “setting a table”/“rearrange the room and setup table” scenario, we are working on applying our method for the subtask of moving chairs around. We want to learn from simulation what visual triggered grasp affordances are suitable for pushing chairs. By utilising a chair pushing simulator made available by JSI, we evaluate the effect of a grasp for pushing later to be used for grasping and pushing an unknown chair in the real scenario. The method will again rely on a segmented point cloud of the chair as input and will then suggest a grasping point for the robot to grasp the chair and execute a pushing behaviour

3.4 Part-based object analysis

Part-based object description turns out to be a very powerful way for bootstrapping object functionality even if the objects are semantically unknown to the system. Within Xperience this could provide new ways for object replacement and make-shift object use, given certain tasks to be executed. However, integration into Xperience demonstrators is currently not planned.

References

[1] Barry Ridge and Aleš Ude. Action-grounded push affordance bootstrapping of unknown objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2791–2798, Tokyo, Japan, November 2013.

[2] Simon Stein, Markus Schoeler, Jeremie Papon, and Florentin Wörgötter. Object partitioning using local convexity. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[3] Mikkel Tang Thomsen, Dirk Kraft, and Norbert Krüger. Identifying relevant feature-action associations for grasping unknown objects. Technical Report 2014–1, Cognitive and Applied Robotics Group, The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, 2014.

Attached Articles

[MKK15] Wail Mustafa, Dirk Kraft, and Norbert Krger. Extracting categories by hierarchical clustering using global relational features. In *Pattern Recognition and Image Analysis, 7th Iberian Conference on*, 2015. (submitted).

[RUU15] Barry Ridge, Emre Ugur, and Aleš Ude. Analysis of action-grounded 3-D features for object affordance classification. In *International Conference on Advanced Robotics (ICAR)*, 2015. (submitted).

[SPW15] Markus Schoeler, Jeremie Papon, and Florentin Wörgötter. Constrained planar cuts - object partitioning for point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (submitted).

[TKK15] Mikkel Tang Thomsen, Dirk Kraft, and Norbert Krüger. Identifying relevant feature-action associations for grasping unmodelled objects. *Paladyn. Journal of Behavorial Robotics*, 2015. (accepted).

[XSP14a] Hanchen Xiong, Sandor Szedmak, and Justus Piater. Comparing binary hamiltonian monte carlo and gibbs sampling for training discrete mrfs wit stochastic approximation (extended abstract). In *17th International Conference on Artificial Intelligence and Statistics (AISTATS14)*, 2014.

[XSP14b] Hanchen Xiong, Sandor Szedmak, and Justus Piater. Towards maximum likelihood: Learning undirected graphical models using persistent sequential Monte Carlo. In *6th Asian Conference on Machine Learning (ACML14)*, 2014. (To appear.).

[XSRSP14] Hanchen Xiong, Sandor Szedmak, Antonio Rodríguez-Sánchez, and Justus Piater. Towards sparsity and selectivity: Bayesian learning of restricted boltzmann machine for early visual features. In *24th International Conference on Artificial Neural Networks (ICANN14)*, 2014.

Extracting Categories By Hierarchical Clustering Using Global Relational Features

Wail Mustafa, Dirk Kraft, and Norbert Krüger

The Mærsk Mc-Kinney Møller Institute, University of Southern Denmark,
Campusvej 55, DK-5230 Odense M, Denmark,
wail@mmmi.sdu.dk

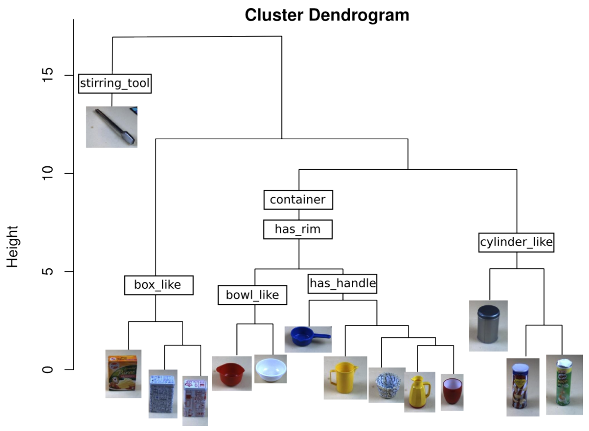
Abstract. We introduce an object categorization system which uses hierarchical clustering to extract categories. The system is able to assign multiple, nested categories for unseen objects. In our system, objects are represented with global pair-wise relations computed from 3D features extracted by three RGB-D sensors. We show that our system outperforms a state-of-the-art approach particularly when only a few number of training samples is used.

1 Introduction

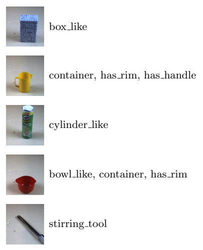
Object categorization is important for a variety of tasks especially when systems are expected to deal with novel objects based on previously acquired knowledge. Such knowledge is built from prior observations through identifying common structures in the visual data. For instance in robotic applications, categories can be linked to manipulation actions allowing for performing predefined actions on novel objects (see e.g., [1]). Categorizing novel objects is also useful in other applications such as driver assistance [2] and video surveillance [3]. In this paper, we introduce an object categorization method based on unsupervised clustering of 3D relational features. Clustering [4] is a powerful tool to automatically find structures in the data that can be, in this context, translated into categories.

In our system, visual data are provided in terms of view-point invariant representations of objects extracted from 3D sensors. These representations code the properties of objects by computing global, pair-wise relations from 3D features (i.e., 3D texlets [5]). This space of feature relations is then expressed in histograms, providing unique and specific object descriptors. Moreover, such descriptors provide a fixed-length feature space, which can be directly fed into the clustering algorithm. These representations have been found to achieve high performance on object instance recognition [6].

In this paper, we apply hierarchical agglomerative clustering [7]. In contrast to flat clustering algorithms such as k-means, hierarchical clustering allows for overlapping of categories (see Fig. 1a for an illustration). This means that very similar categories are nested within larger clusters forming a structure in which more generic categories are found on top of less generic ones. This provides flexibility in selecting the level of abstraction.



(a) Illustrative hierarchical clustering



(b) Examples of labeled objects

Fig. 1. Examples of hierarchical clustering and object labeling. The thumbnails are resized for better visualization and they don't necessarily reflect their actual relative sizes.

Existing object categorization methods assume that objects belong to mutually-exclusive (single) categories [8]. In this paper, we consider scenarios where objects can have multiple, nested categories (see Fig. 1b). Such scenarios are very common when dealing with everyday objects. One important aspect of this approach is that it is inherently capable of providing multiple categories. In contrary, other approaches will not only require learning multiple classifiers (see e.g., [8]) but also—as we show in this paper—perform poorly on nested categories.

To evaluate the system, we hand-labeled our benchmark object set with a number of visual categories. We use the labeled categories from the training subset to find the best matching ones in the hierarchy. Then, we evaluate the performance of the system on the test subset. Note that although the hierarchy is built unsupervised, finding the corresponding categories is done in a supervised way. This is, however, necessary for evaluation. This procedure is repeated using different parameterizations of the visual representations in order to empirically find the best set of parameters for each category.

We compare this approach to a supervised approach using Random Forests [9] where both use our visual features. In addition, we make a comparison with a state-of-the-art method (Hierarchical Matching Pursuit [8]) that works on RGB-D data and extracts distinct visual features. The main achievements of this work can be summarized as follows:

- We introduce a multi-category object categorization method that can predict more than one category.
- We show that—using hierarchical clustering for finding categories—we perform better than classification with Random Forests. Our method also outperforms a state-of-the-art method on our dataset.

– We demonstrate that the use of our visual features, compared to the features used by a state-of-the-art method, allow the system to perform well with already very few samples.

2 Related Work

Early research on object categorization focused on generic object representations that capture shape at high levels of abstraction (such as generalized cylinders [10], superquadrics [11], or geons [12]). The difficulty involved in reconstructing such abstractions from real objects has led to the development of solutions that could recognize only exemplar objects [13] (i.e., object recognition), which require little or no abstraction. Over the years, the gap between the low-level and the high-level abstractions has been narrowed by introducing representations that are invariant to a number of geometrical properties such as view-point, translation, rotation, and scaling. Such representations make use of typically descriptors of local image patches such as SIFT [14] and HOG [15] features.

Belongie et al [16] proposed representing objects using 'shape contexts', which uses relative shape information within a local neighborhood. The shape contexts were later extended to 3D in [17]. In this paper, we use shape relations of 3D features presented in [6], which are similar to shape context but are defined in a global context. Additionally, we go beyond the work in [6] by addressing the scale-invariance to obtain a more abstract representation that is important for object categorization.

Recently, hierarchical approaches for object representation have shown high performance on large dataset. Notably, Bo et al [8] introduced a multi-layer network that builds feature hierarchies layer by layer with an increasing receptive field size to capture abstract representations. They shows that their method achieves state-of-the-art performance in a large-scale RGB-D dataset of objects [18]. It is worth noting that these results are based on very large training data with significant computational cost.

Existing object categorization systems typically apply supervised learning to recognize object classes that correspond to labeled categories and associate only one category per object [1, 8]. In our approach, categories are learned in an unsupervised way. Using hierarchical agglomerative clustering introduced by [7] allows for associating more than one category per object. To do this, we use hierarchical agglomerative clustering introduced by [7]. Building such a hierarchy can be seen as a way to obtain higher levels of abstraction (from the visual features) where more generic categories are formed at the top of the hierarchy.

3 System Description

The components of the object categorization system introduced in this paper are shown Fig. 2. The system operates in a setup in which three views are captured by three Kinect sensors, which are mounted in a close to equilateral triangular configuration. The process starts with scene preprocessing for table removal and

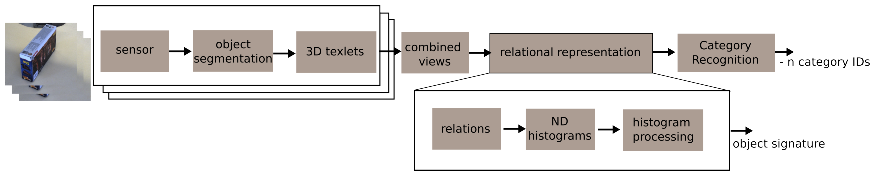


Fig. 2. System Overview: block diagram of the different components.

object segmentation in the 3D point cloud data. In following, we describe in detail the other components.

3.1 Object Representation Using Histogram of Relational Features

For the approach we introduce in this paper, object shapes are described as distributions of *relations* between pairs of 3D features. The relations we use are intrinsically pose-invariant.

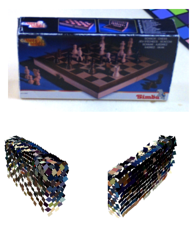
From RGB-D data (Kinect sensor), we extract our 3D features—*3D texlets* [5]. The 3D texlet has both position and orientation, and provides absolute informations (relative to an external reference frame) of objects in the 3D space. In our system, we combine 3D texlets from three view resulting in a rather complete object information (see Fig. 3a). To describe an object, we compute a set of pair-wise relations from all pairs of texlets belonging to the object.

Shape relations are similar to the 3D shape context introduced as local descriptors by [17], however, they are used here as global descriptors of objects. Having combined multiple 3D views of objects allows such global descriptors to become robust and rich representations for fast learning.

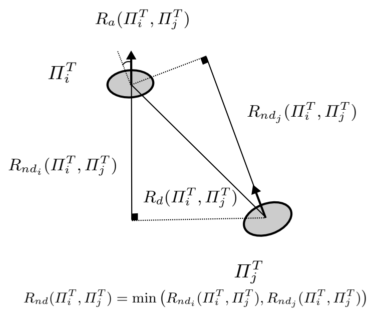
In [6], we defined in detail three shape relations used for object instance recognition, namely, Angle Relation $R_a(\Pi_i^T, \Pi_j^T)$, Distance (Euclidean) Relation $R_d(\Pi_i^T, \Pi_j^T)$, and Normal Distance Relation $R_{nd}(\Pi_i^T, \Pi_j^T)$—they are also depicted in Fig. 3b. Note that the relations transform an absolute pose-dependent representation into a relative pose-independent one. For instance, the distance relation $\mathcal{R}_d$ transforms texlets' positions into inter-texlet distances.

The two distance relations are scale-variant, which is suitable for object instance recognition where object size matters and shall be encoded. However, for object categorization, because what defines a category is usually independent of scale, scale-invariance is crucial. Therefore, we introduce a new scale-invariant distance relation referred to as *Scaled Distance Relation*, $R_{sd}(\Pi_i^T, \Pi_j^T)$. The scaled distance is computed by dividing the Distance Relation by the maximum distance within an object. For robustness against outliers, the maximum distance is calculated as the median value of the highest 10% distance relation values. Fig. 3c shows an example of two objects with different sizes (belong to the same category), comparing using distance and scaled distance when representing objects. One aspect we investigate in the experiment section is the performance of the system on each category when combinations of different relations are used.

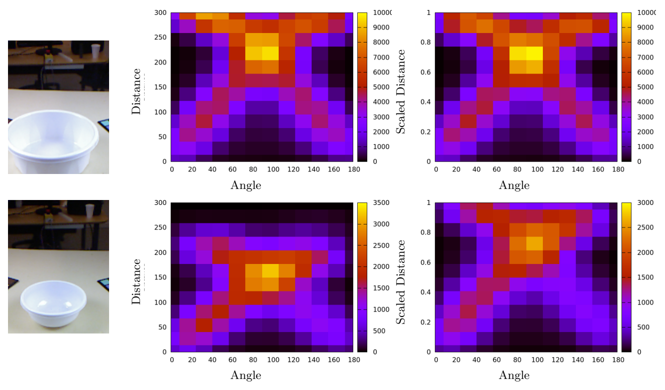
The final object representation is obtained by binning the selected relations in *multi-dimensional histograms*, which model the distributions of the relations



(a)


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

(b)



(c)

Fig. 3. Texlet's shape relations. (a) extracted 3D texlets of an object. (b) definition of three shape relations. (c) 2D histograms of two objects belonging to the same categorize. In the right column the distance is scaled

in fixed-sized feature vectors fed to the learning algorithm. Examples of 2D histograms are shown in Fig. 3c. Different values of binning size are also investigated in the experiments. Another process that is optionally performed on our histograms is *smoothing* using ND Gaussian filters to reduce the noise.

3.2 Finding Categories Using Hierarchical Clustering

Hierarchical Clustering. The quality and the invariance properties of the object representation presented in the previous section make it attractive for object categorization. We propose using unsupervised category learning through clustering using agglomerative hierarchical clustering [7] (R implementation [19]). By doing so, we build a hierarchy of clusters from unlabeled data where each cluster (branching point in the hierarchy) is considered as a potential category that can be linked (by an autonomous process) to an actual category. Note that we use the Euclidean distance as a dissimilarity measure (between all pairs of data samples or object instances in our case) whereas as a linkage metric, we use Ward's criterion, which aims at minimizing the total within-cluster variance [7].

Finding Categories From Human-Labeled Categories. Particularly for this paper, to validate our approach, we use human-defined labels from the training samples (Fig. 1b shows some examples) and then we find the corresponding categories in the hierarchy. Those definitions of categories are rather subjective and might not correspond to real ones. Therefore, we also compare our approach with other approaches—one of which even extracts different features from the raw RGB-D data.

To find a category in the hierarchy that correspond to a labeled one, we search for the cluster that contains the most similar set of object instances to the set of objects labeled as such. To compute the similarity, we use Jaccard's index [20], which measures the similarity between finite sample sets and is defined as the size of the intersection divided by the size of the union. The Jaccard's index rewards the existence of the object in the prospective cluster and also punishes for the absence thereof. This prohibits assigning categories to very specific (at the bottom of the hierarchy) or very generic clusters (at the top of the hierarchy).

Note that, although building the structure is done in an unsupervised way, finding the learned categories corresponding best to the labeled categories is performed in a supervised fashion (based on labeled data).

Predicting Categories for Novel Objects. To allow the system to categorize objects, the proposed method should provide a prediction mechanism. Traditionally for supervised learning, the learned model is used to make predictions for the novel object. Such a model usually forms a map of the feature space allowing for making predictions based on the features of the novel object. In our method, the principle concept we propose for prediction is to identify where the novel object falls in the learned (previously-built) hierarchy. This requires involving the training samples because the hierarchy is built directly from the training data. This seems computationally inefficient especially for large set of training samples. However, we show in this paper that our method requires few training samples.

To implement this, in the prediction phase, we first add the novel object to the objects previously used for training. Once the hierarchy is built again, we identify the closest sibling of the novel object. The novel object will then inherit all the branching points—including the ones associated with the labeled categories—from the sibling object. Finally, the predicted categories for the novel object will be all the inherited categories.

4 Dataset and Experiments

Dataset. To benchmark our approach, we use a dataset of 100 objects with 30 different samples (random poses) for each object[1]. The dataset was originally

[1] `http://caro.sdu.dk/index.php/sdu-dataset` (we will make it accessible by the final submission of this manuscript)

created to test the performance of the object instance recognition present in [6]. The selection of objects covers a wide range including industrial and household objects, some of them taken from the KIT dataset [21].

To validate our approach, we hand-labeled the objects in the dataset with purely visual as well action-related categories (see Fig. 1b). Note that a single object can have multiple (nested) categories. This allows us to study the performances of the different approaches on such cases.

Comparison Methods. The approach we introduce in this paper is compared with two different methods that use classical supervised learning. We apply those methods in N-classifier mode where N refers to the number of categories. This allows the methods to provide multiple categories per object and hence make them comparable with our approach. The first method we compare with is *a Random Forest classifier*, which uses the same features as the introduced approach (see Sect. 3).

The second method is called *Hierarchical Matching Pursuit (HMP)* [8], which is a state-of-the-art method. HMP is a multi-layer sparse coding network that builds feature hierarchies layer by layer with an increasing receptive field size to capture abstract representations from raw RGB-D data. Note that HMP was not designed to combine features from different views in the 3D space. Therefore, to make it comparable to our multi-view system, we provide all the three views in the training phase. Additionally, we compare all methods when only one view is used.

The comparison also includes a 'dummy' classifier that generates uniformly-distributed random category predictions. Comparing the different methods with this classifier may indicate whether a particular method has failed to achieve reasonably good performance on the category in question.

Histogram Variations. In the experiments below, we vary the parametrization of our object representation. The objective is to find out the set of parameters that yields the best performance on each category. Note that those variations are applied to the methods in which our object representation is used (namely, the proposed approach and the Random Forest approach). The object representation was discussed in detail in Sect. 3. The exact list of parameters we vary are the following:

– Set of relations: We vary what relations to use for representing objects from the ones defined in Sect. 3: Angle Relation, Distance Relation, Normal Distance Relation, and Scaled Distance Relation.
– Relational dimensionality: We vary how we combine relations in ND histograms. The combinations we apply are: 1D histograms of the individual relations and 2D histograms of Angle Relation with one of distance relations.
– Histogram binning: Here, we vary the ND histogram bin size among the following values: 10, 20, 50 and 100. For simplicity, the bin size is fixed across dimensions in the 2D case.
– Filtering: We analyze the impact of applying filtering with Gaussian kernel.

In addition to the above-mentioned parameters, we also experiment with the impact of performing vector normalization on the final object representation.

Experimental Procedure. In the following experiments, we study the performance of each method for categorizing novel objects. Therefore, in each experiment, the object dataset is divided into training and test subsets where sampling is performed in a way that prohibits the presence of samples from the same object in both subsets. Allowing otherwise, results in performing recognition of object instances rather than object categories in which in our tests we obtained significantly higher performance. The size of the test subset is set to 100 samples per category whereas the size of the training subset is allowed to vary—all samples are randomly chosen. Each experiment is executed 20 times from which the average F1 score and the standard deviation are computed. Note that the same training and test subsets are passed to each method.

In the result shown in figure 4b, the size of the training set varies among certain values: 1, 3, 5, 10, 15, 20, 30, 50, 70 and 100 per category. By doing this, we are able to study the performance of each method when only a small number of training samples are available and how that changes when the number increases.

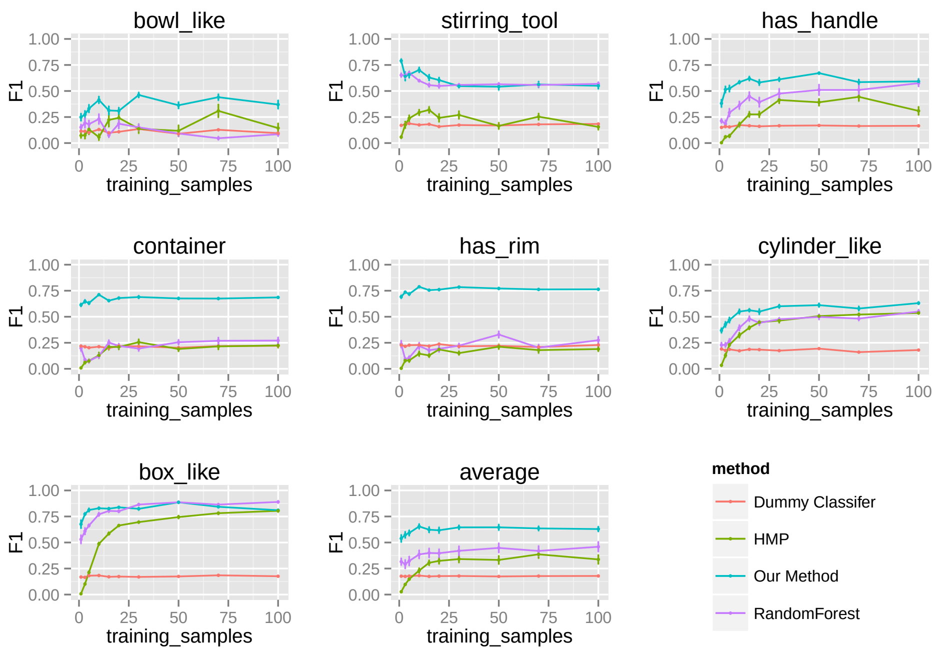
Results. Fig. 4 shows the performance of object categorization on 7 categories. Each sub-figure shows the average F1 score and the standard deviation for a varying number of training samples. The average performance on all categories is also shown in a separate sub-figure. The results show that the method introduced in this paper generally achieves the highest performance in identifying the categories particularly when a few training samples are used. This means that our method is able to learn faster and also generalize better when the training samples are limited.

Conclusion. Both the proposed method and the Random Forest approach use the same extracted visual representations of objects. This indicates that finding categories in clusters formed hierarchically in unsupervised way has a better generalization than the supervised learning of categories. Additionally, because both approaches outperform the HMP method, which extracts different visual representations, the results suggest that our visual representation provides strong features for describing object categories. This is particularly clear in the three-view case where our representation allows for combing the three views in 3D resulting in a rather complete object description.

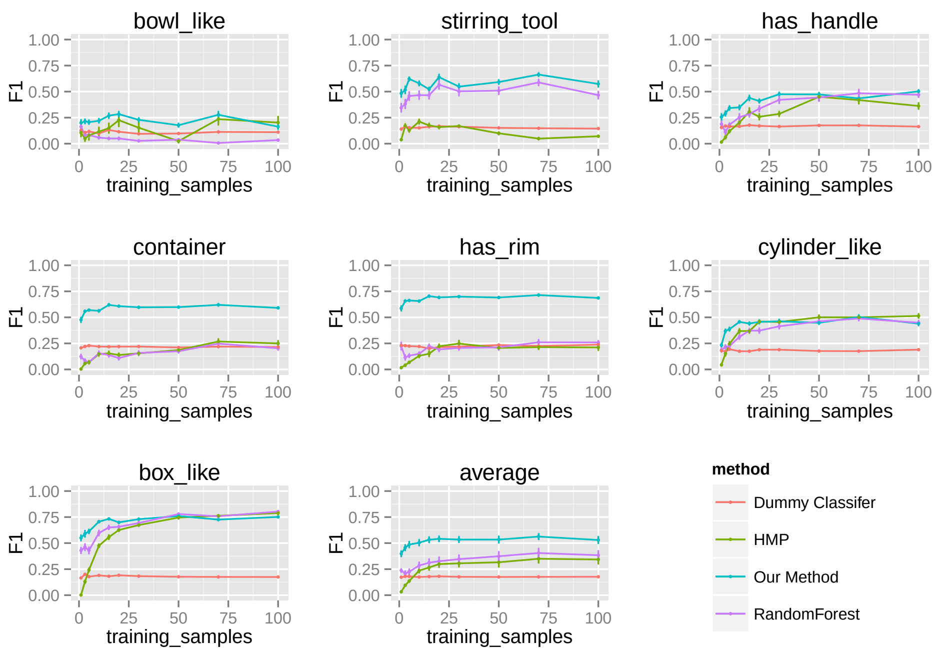
For some categories (namely, 'container' and 'has_rim'), our method achieves relatively good performance in identifying the categories whereas the other methods fail (perform comparatively the same as the dummy classifier). Those categories are nested categories (i.e., in this case, any container also has a rim). This indicates that our approach is able to identify the relation between the two categories. The supervised approaches, on the other hand, try to learn discriminatively the two categories, which for most samples have very similar representations. This may explain their failure in identifying the nested categories.

Acknowledgment

This work has been funded by the EU project Xperience (FP7-ICT-270273).



(a) three camera views



(b) one camera view

Fig. 4. The performance of object categorization on 7 categories. In (a) three camera views are used whereas in (b) one camera view is used. Using a 2D histogram of angle and scaled distance (with 10 bins at each dimension) yields the best performance in all categories except for 'bowl_like' (in this case, combining two 1D histograms of angle and scaled distance with 12 bins). Also, applying filtering and vector normalization helps achieving the best performance in all cases.

References

1. Marton, Z.C., Pangercic, D., Rusu, R.B., Holzbach, A., Beetz, M.: Hierarchical object geometric categorization and appearance classification for mobile manipulation. In: Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on, IEEE (2010) 365–370
2. Laika, A., Stechele, W.: A review of different object recognition methods for the application in driver assistance systems. In: Image Analysis for Multimedia Interactive Services, Eighth International Workshop on, IEEE (2007) 10–10
3. Graham, S., Wood, D.: Digitizing surveillance: categorization, space, inequality. Critical Social Policy **23**(2) (2003) 227–248
4. Xu, R., Wunsch, D., et al.: Survey of clustering algorithms. Neural Networks, IEEE Transactions on **16**(3) (2005) 645–678
5. Olesen, S.M., Lyder, S., Kraft, D., Krüger, N., Jessen, J.B.: Real-time extraction of surface patches with associated uncertainties by means of kinect cameras. Journal of Real-Time Image Processing (2012) 1–14
6. Mustafa, W., Pugeault, N., Buch, A., Krger, N.: Multi-view object instance recognition in an industrial context. Robotica (Accepted)
7. Ward, J.H.: Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association **58**(301) (1963) 236–244
8. Bo, L., Ren, X., Fox, D.: Unsupervised feature learning for rgb-d based object recognition. In: International Symposium on Experimental Robotics (ISER. (2012)
9. Breiman, L.: Random forests. Machine Learning **45**(1) (2001) 5–32
10. Binford, T.O.: Visual perception by computer. In: IEEE conference on Systems and Control. Volume 261. (1971) 262
11. Pentland, A.P.: Perceptual organization and the representation of natural form. Artificial Intelligence **28**(3) (1986) 293–331
12. Biederman, I.: Recognition by components: A theory of human image understanding. Psychological Review **94**(2) (1987) 115–147
13. Campbell, R., Flynn, P.: A survey of free-form object representation and recognition techniques. Computer Vision and Image Understanding **81**(2) (2001) 166–210
14. Lowe, D.: Object recognition from local scale-invariant features. In: Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on. Volume 2. (1999) 1150 –1157 vol.2
15. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision (IJCV) **2**(60) (2004) 91–110
16. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. Pattern Analysis and Machine Intelligence, IEEE Transactions on **24**(4) (2002) 509–522
17. Frome, A., Huber, D., Kolluri, R., Bulow, T., Malik, J.: Recognizing objects in range data using regional point descriptors. In: Proceedings of the European Conference on Computer Vision (ECCV). (May 2004)
18. Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view rgb-d object dataset. In: IEEE International Conference on Robotics and Automation (ICRA). (2011) 1817–1824
19. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. (2013) ISBN 3-900051-07.
20. Levandowsky, M., Winter, D.: Distance between sets. Nature **234**(5323) (1971)
21. Kasper, A., Xue, Z., Dillmann, R.: The kit object models database: An object model database for object recognition, localization and manipulation in service robotics. International Journal of Robotics Research (IJRR) **31**(8) (2012) 927–934

Analysis of Action-Grounded 3-D Features for Object Affordance Classification*

Barry Ridge†, Emre Ugur‡, and Aleš Ude†

Abstract— **Recent work in robotics, particularly in the domains of object manipulation and affordance learning, has seen the development of *action-grounded features*, that is, object features that are defined dynamically with respect to manipulation actions. Rather than using pose-invariant visual features, as is often the case with object recognition, such features are grounded with respect to the manipulation of the object, for instance, by using shape features that describe the surface of an object relative to the push contact point and direction. In this paper we provide an experimental comparison between action-grounded features and non-grounded features in an object affordance classification setting. Using an experimental platform that gathers 3-D data from the Kinect RGB-D sensor, as well as push action trajectories from an electromagnetic tracking system, we provide experimental results that demonstrate the effectiveness of this action-grounded approach across a range of state-of-the-art classifiers.**

I. INTRODUCTION

In the field of autonomous robotics, vision and control are often treated as distinct domains in which useful data and action commands are derived, processed, or manipulated separately to achieve a broader goal. In the context of robotic affordance learning [1], [2], this has traditionally been the standard approach, and simplifying assumptions are often made in order to make targeted problems more soluble. For example, in the case of object push affordance learning [3]–[7], if the desired result is to learn how the positions and orientations of objects change when pushed, the learning task can be simplified by selecting prior object models, using standard computer vision techniques to localise the object models within a scene, and inferring data such as end effector contact points on the objects using the models. However, when fewer assumptions are made about the shapes of objects, the types of push actions that might be performed, and the resulting affordances, such techniques may not be as feasible. On the other hand, the formation of dynamic object models, through interactive experience, that are semantically associated with actions at a more intrinsic level, is a promising research area in that it offers the potential to enhance developmental robotic learning and, ultimately, robotic autonomy.

In this paper, we explore the use of action-grounded 3-D object shape features in an object push affordance classification setting. Using an RGB-D sensor to gather 3-D point cloud data of objects, and using an electromagnetic tracking system to gather push trajectory data, objects on a table surface were pushed by a human experimenter (cf. Fig. 2) whose hand motion trajectories were tracked while 3-D point clouds of the objects were recorded. The objects were pushed from various different positions on their surfaces and from various different directions, exhibiting a number of different affordances such as forward translations, forward topples, left rotations and right rotations, depending broadly on the shapes of the objects, their orientations, and how they were pushed. Our chief point of investigation was to determine whether the proposed action-grounded 3-D shape feature approach to dynamic object affordance modeling in this setting offers any improvement over more standard methods.

A. Related Work

E. Gibson discusses that learning affordances refers to "narrowing down from a vast manifold of (perceptual) information to the minimal, optimal information that specifies the affordance of an event, object, or layout"[**?**]. Selection and use of the action-relevant features have already been studied and shown to be effective in learning and representing affordances. For example Montesano et al. [4] learned affordances only using the object properties that are found to be relevant to the corresponding actions, for more effective predictions and planning. Hart and Grupen also discussed the necessity of selecting the relevant features in order to capture the salient information while learning affordances [8]. Lately, it was also shown that use of relevant features in affordance learning can lead to the discovery of hierarchical structures in predicting interdependent affordances of different complexities[9]. While all these studies discuss how to select the relevant features from a general purpose feature set, we aim to find a transformed feature set based on the actions in consideration.

Implicit encoding of object manipulation information in object shape feature descriptors has been applied in the grasping field, often via the use of haptic or tactile sensors. Recently, Björkman et al. [10] employed tactile measurements from the haptic finger sensors on a robotic hand to enhance prior visual object models via implicit object surface modeling. Meier et al. developed a probabilistic spatial approach for building compact 3-D representations of unknown objects probed by tactile sensors using Kalman filters to build a probabilistic model of the contact point cloud.

Object shape features from 3-D vision have been used in

*This research has been supported by EU FP7 project Xperience (ICT-270273).

†B. Ridge and A. Ude are with the Laboratory of Humanoid and Cognitive Robotics, Department of Automation, Biocybernetics and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia. `barry.ridge at ijs.si`, `ales.ude at ijs.si`

‡E. Ugur is with the Intelligent and Interactive Systems Laboratory, University of Innsbruck, Austria. `emre.ugur at uibk.ac.at`

prior work on push affordance learning [6], [7], [11], [12], but grounding such features relative to pushing actions has not been studied as extensively. Recent work by Hermans et al. [13], [14] used shape features encoded in a coordinate frame defined by object centres and push locations based on 2-D projections of object point clouds. Krainin et al. developed an approach to building 3-D models of unknown objects for grasping based on a depth camera observing a robotic hand while moving an object and modeling the object surface dynamically using sets of small surface patches.

In our previous work [15], we developed a similar idea, employing full 3-D shape features grounded with respect 3-D action trajectories to perform bootstrap discovery and prediction of object affordance classes using a multi-view self-supervised learning algorithm. In this work, by contrast, we focus on examining action-grounded 3-D visual features in more detail and investigating whether or not they provide an improvement over features that are not implicitly defined with respect to the manipulation.

The remainder of the paper is structured as follows. In the following section, we give a brief overview of our experimental platform including the object point cloud segmentation process. In Section III we describe how the action-grounded features and non-grounded features are derived, including the object segmentation process, the transformation of both objects and action trajectories to the action coordinate frame, and the description of the features themselves. In Section IV we describe our experiments with various state-of-the-art classifiers and results. Finally, in Section V, we conclude and discuss potential future work.

II. EXPERIMENTAL SETUP

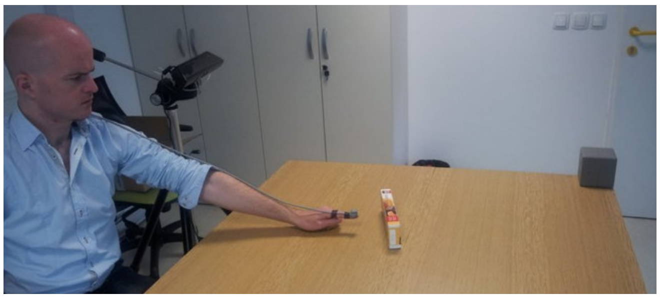


Fig. 2. Our setup for human object push affordance data gathering.

In our experimental setup for human object push data gathering, shown in Figure 2, we employed a Microsoft Kinect™ RGB-D sensor for gathering 3-D point cloud data of scenes and objects, and a Polhemus Patriot™ electromagnetic tracking system for gathering trajectory data of human hand motions. A wooden table with a wooden frame was used as the work surface in order to avoid electromagnetic interference from metallic objects in the environment. A tracking sensor was placed at the end of the index finger of a human experimenter, while the tracking source was located at a corner of the table with the Kinect facing the table at a $45°$ angle as shown in Figure 2. Objects were placed at arbitrary locations on the table surface where they were pushed from various directions and at various contact points by the experimenter. 3-D point clouds of the scene were recorded both before and after each push interaction while hand trajectories were tracked during the interaction. Both the point clouds and the trajectories were processed offline where the objects were segmented from the table surface, object point clouds and push trajectories were transformed into the push action coordinate frame, and action-grounded shape features were extracted.

A. *Object segmentation*

We used tools from the Point Cloud Library (PCL)[1] to perform dominant plane segmentation on scene point clouds in order to acquire segmented point clouds of the objects lying on the table surface. This involved using a pass-through filter to subtract points in the scene cloud outside certain range limits, using RANSAC [16] to fit a plane model to the scene cloud, subtracting those scene points that were plane inliers, and clustering the remaining points to find the objects using Euclidean clustering [17].

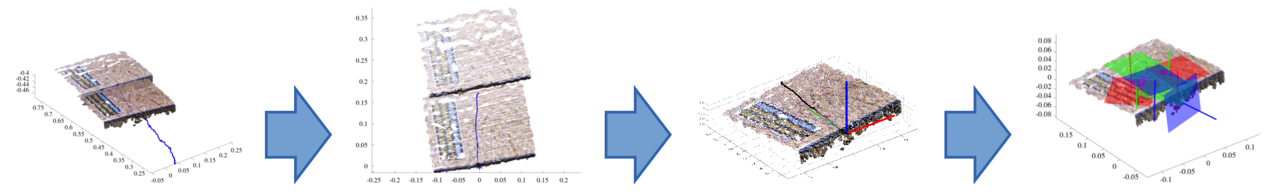
III. ACTION-GROUNDED VS. NON-GROUNDED 3-D SHAPE FEATURES

We define the action frame to be the coordinate system with its origin at the contact point on the object, its positive y-axis pointing in the direction of the pushing motion parallel to the table surface, its positive z-axis pointing upward from the table surface, and its positive x-axis pointing to the right of the object. In order to transform both the object point cloud and the push trajectory into the action frame, we perform the following procedure. Firstly, we transform the push trajectory from the Patriot tracker coordinate system to the Kinect coordinate system by using least-squares adjustment on a series of control points and calculating a rigid body transformation of the form $\mathbf{x}' = \mathbf{c} + \mathbf{R}\mathbf{x}$, where $\mathbf{x}'$ is the transformed vector, $\mathbf{x}$ is the initial vector, $\mathbf{c}$ is the translation vector, and $\mathbf{R}$ is a rotation matrix. The control points are gathered prior to performing pushing experiments by placing the tracking sensor at various positions in the workspace, recording the sensor position, recording the Kinect point cloud of the scene, then locating the sensor in the point cloud. Since the pushing motions performed in our experiments always follow an approximately linear trajectory, we proceed by using orthogonal distance regression via singular value decomposition to fit a 3-D line to the push trajectory. Finally, we find the point of intersection between this fitted line and the pre-push object point cloud, infer this to be the contact point, and finally transform the pre-push object point cloud as well as the push trajectory to the action frame as defined by the contact point and the fitted line. This process is visualized in Figure 1.

A. *Action-grounded 3-D shape features*

With the pre-push object point cloud now grounded in the action coordinate frame, we turn to generating a feature

[1]http://pointclouds.org



(a) Object point clouds before/after interaction + hand push trajectory

(b) Point clouds + trajectory transformed to push action coordinate frame.

(c) Pre-push point cloud action coordinate frame grounding.

(d) Fitting planes to pre-push object cloud parts, grounded with respect to action.

Fig. 1. Action-grounded shape feature extraction pipeline.

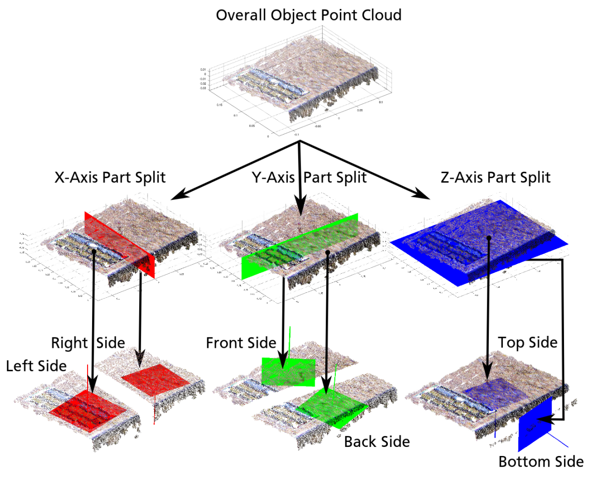


Fig. 3. Partitioning a sample object point cloud into sub-parts. Top row: original pre-push object point cloud. Middle row: partitioning planes divide the point cloud evenly in each dimension to create sub-parts. Bottom row: planes are fitted to each sub-part for feature extraction.

descriptor that describes the shapes of the object point clouds with respect to the pushing action and that is rich enough to capture the resulting affordance effects. The main idea behind our approach is to divide the object point clouds into cells of sub-parts and use the properties of the sub-parts of the point clouds as a basis for the feature descriptor. More concretely, we divide each object point cloud evenly with respect to its minimum and maximum points along each coordinate axis such that there are seven cells that overlap for redundancy: one for the overall point cloud, two for the x-axis, two for the y-axis, and two for the z-axis. We then use two types of feature descriptors in each cell. To gauge the position of the sub-part in each cell relative to the action frame, we find the centroid of the points in the cell, which gives us three features. To gauge the shape of the sub-part in each cell relative to the action frame, we fit a planar surface to the points within the cell and use the two largest coordinates of the plane normal as features. Examples of these features being extracted from different point clouds are shown in Figure 7.

Using these five types of features, three for relative part position and two for planar surface fit orientation, we extract the five features for each part. This results in the following list of 35 features that are extracted:

$O^a_{1\ldots5}$: Global point cloud centroid/plane features.
$O^a_{6\ldots10}$: x-split, left side centroid/plane features.
$O^a_{11\ldots15}$: x-split, right side centroid/plane features.
$O^a_{16\ldots20}$: y-split, front side centroid/plane features.
$O^a_{21\ldots25}$: y-split, back side centroid/plane features.
$O^a_{26\ldots30}$: z-split, top side centroid/plane features.
$O^a_{31\ldots35}$: z-split, bottom side centroid/plane features.

B. *Non-grounded 3-D shape features*

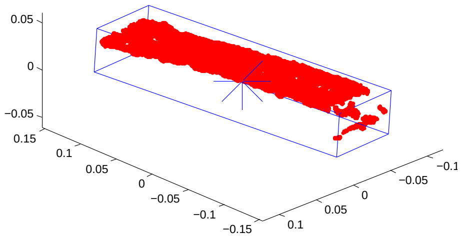


Fig. 4. Sample object point cloud with bounding box and centroid for definition of non-action-grounded coordinate frame.

For the non-grounded feature set, we used the same methodolgy in terms of the object parts division as described in the previous sub-section, and we extracted the same features as before, but crucially, the coordinate frame was not defined with respect to the push trajectory. Instead, we fit a bounding box to the object point cloud, found the centroid of the bounding box, and used that as the origin for our non-grounded coordinate system. This is illustrated for a sample object point cloud in Figure 4. This seemed like the most reasonable methodology for performing a direct comparison between the action-grounded features and similar non-action-grounded features. Thus, we have:

$O^b_{1\ldots35}$: non-grounded analogues of features $O^a_{1\ldots35}$.

C. *Action features from push trajectories*

In order to make it feasible for the classifiers to be able to predict the affordances of the objects using non-action-grounded visual features, it is necessary to include information about the action in the feature descriptor in some form. We approached this by augmenting the non-grounded 3-D features with features derived from the object contact point and the push action trajectory. These were as follows:

$O^b_{36\ldots38}$: x, y, z-coordinates of object contact point.
$O^b_{39,40}$: x, y-parameters of push trajectory line fit.

The z-parameter of the line that was fit to the push trajectory was deemed redundant since all pushes took place in the same plane, i.e. along the table surface. Combining these 5 features with the 35 features of the previous sub-section yielded a 40-dimensional feature vector for the non-grounded feature set.

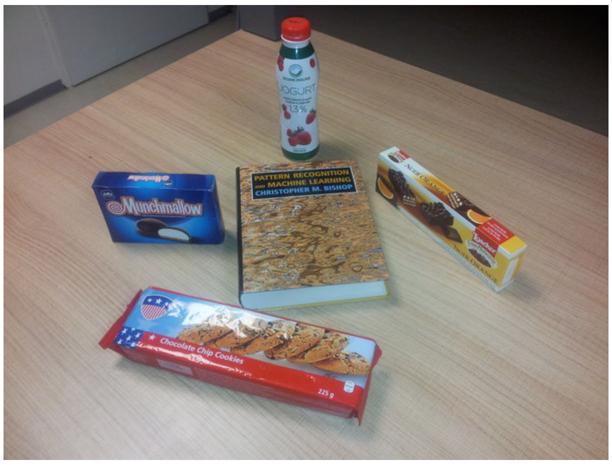


Fig. 5. Test objects used in our experiments.

IV. EXPERIMENTS

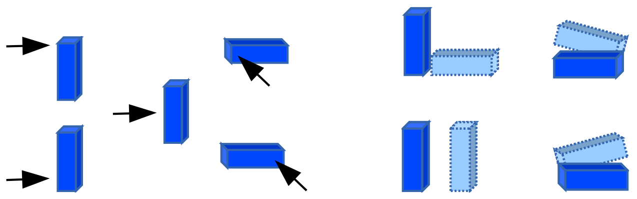


Fig. 6. The 5 different push types (left) and 4 affordance classes (right).

The experimental environment was set up as shown in Figure 2. We selected 5 household objects (cf. Fig. 5) for the experiments: 4 flat-surfaced objects; a book, a marshmallow box, a cookie packet, and a biscuit box, and 1 curved-surfaced object; a yoghurt bottle. A dataset was collected as follows, where object push tests were performed on each of the 5 objects and the resulting data was processed, leaving 120 data samples. Objects were placed at random start locations and in various poses within the workspace and within view of the Kinect sensor, and the human experimenter would perform straight-line pushes on the objects, attempting to keep the pushes within reasonable limits of 5 different push categories: pushing through the top, bottom, left, right and centre of the objects respectively, from the direction of the field of view of the Kinect. Table I contains a matrix detailing the numbers of samples collected for each of the objects in each different possible pose. Certain objects prohibited certain poses, e.g. neither the cookie pack nor the book could be placed in sideways or upright poses.

The data collection process resulted, by inspection, in 4 different affordance categories being produced, and the samples were hand-labelled with four ground truth labels to reflect this: *left rotation*, *right rotation*, *forward translation* and *forward topple*. These various push types and resulting affordances categories illustrated conceptually in Figure 6 are sample object interactions are shown in Fig. 7.

TABLE I
OBJECT/POSE MATRIX

	Flat	Sideways	Upright	Total
Cookie Pack	18	0	0	18
Marshmallow Box	9	15	9	33
Biscuit Box	9	15	9	33
Book	18	0	0	18
Yoghurt Bottle	9	0	9	18
Total	63	30	27	120

A. Evaluation Procedure

In order to compare the action-grounded and non-action-grounded feature sets we used 10-fold cross-validation with multiple different state-of-the-art classifiers. Two learning vector quantization-based algorithms [18], *generalized relevance learning vector quantization (GRLVQ)* [19] and *supervised relevance neural gas (SRNG)* [20] were used, alongside *multinomial logistic regression (MLR)*, *support vector machines (SVM)* and *random forests (RF)* [21]. The cross-validation was performed in 10 separate trials and the results were averaged in order to account for random prototype initialization and other variations between runs in the various algorithms.

In the cases of both GRLVQ and SRNG, following the notation used in [20], the main learning rate parameter ϵ^+ was set to 0.2, while ϵ^- was set to 0.04, and ϵ for the λ feature relevance update was set to 0.02.[2] A set of 60 prototype vectors were used in each case, divided into 15 prototypes per class. Training ran for 5000 epochs over the training data in each case to ensure convergence. The logistic regression model used a multinomial logit link function and a confidence interval of 95%. In the case of the SVM, parameters were optimized using cross validation over the training data prior to training.[3] The random forests model used 500 trees.[4]

B. Results

The discussion of the results is divided into two sub-sections. In the first of these, we analyse the performance of each of the classifiers on both the non-grounded and action-grounded datasets in an effort to evaluate whether action-grounding can result in classification performance gains. In the second, we look at how action-grounding affects the relevances of individual features by exploiting the feature relevance determination mechanism provided by the SRNG algorithm.

[2]http://www.mathworks.com/matlabcentral/fileexchange/17415-neural-network-classifiers

[3]SVM implementation: http://www.csie.ntu.edu.tw/ cjlin/libsvm/

[4]RF implementation: https://code.google.com/p/randomforest-matlab/

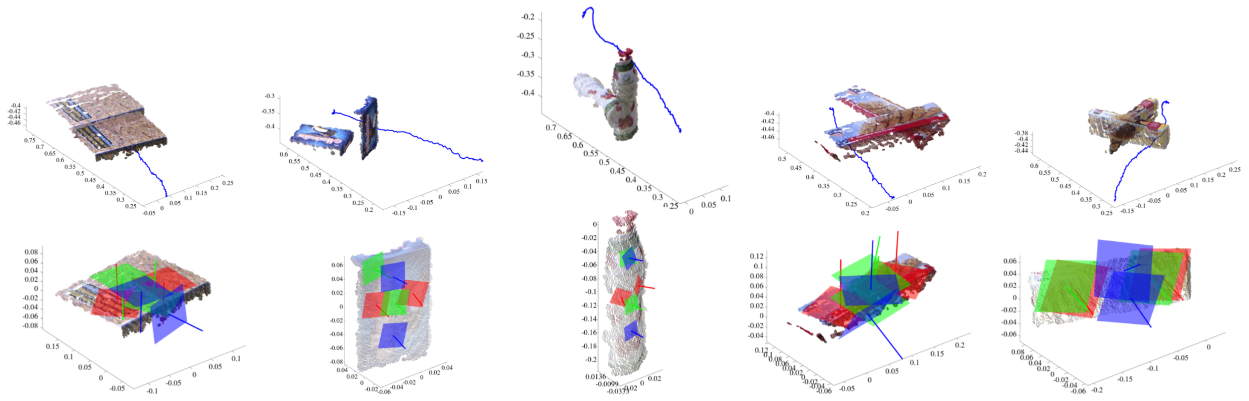


Fig. 7. Action-grounded shape feature extraction. Top row: pre-push and post-push 3-D point clouds and action trajectories for the five test objects being pushed in various different ways. Bottom row: action-grounded shape feature extraction (cf. Section III-A) for the pre-push point clouds. Plane fits are shown in red for the x-axis divisions of the point clouds, in green for the y-axis divisions, and in blue for the z-axis divisions. Four different affordances are visible in the columns from left to right: forward translation, forward topple, right rotation, and left rotation.

TABLE II

MEAN 10-FOLD CROSS-VALIDATION RESULTS: TRAINING SETS

	Non-Grounded	Action-Grounded
GRLVQ	94.9± 3%	97.3± 2%
SRNG	95.5± 3%	98.0± 1%
SVM	97.3± 2%	98.6± 1%
MLR	100.0± 0%	100.0± 0%
RF	100.0± 0%	100.0± 0%

TABLE III

MEAN 10-FOLD CROSS-VALIDATION RESULTS: TEST SETS

	Non-Grounded	Action-Grounded
GRLVQ	87.3 ±10%	92.6 ±8%
SRNG	88.3 ±9%	93.0 ±7%
SVM	82.5 ±10%	86.2 ±9%
MLR	74.6 ±12%	78.1 ±13%
RF	86.0 ±10%	95.8 ±6%

1) Classifier Performance: The main classifier performance results are shown in Tables II and III, where average cross-validation classification matching accuracies are shown for the 9-fold training sets and single-fold test sets respectively. GRLVQ and SRNG, the two learning vector quantization methods, perform robustly in all cases and benefit from action-grounding in both training and test cases. Interestingly, while the SVM also benefits from action-grounding, it does not perform quite as well as the LVQ-based methods, which are likely deriving a comparative benefit from their in-built feature relevance determination mechanisms which the SVM, in the implementation used here, does not share.

The MLR model, given its high performance on training data and relatively poor performance on test data in both the non-grounded and action-grounded cases clearly suffers here from overfitting. It could be the case that the high-dimensionality of the datasets causes it difficulty and that it would benefit from dimensionality reduction or regularization, though neither have yet been tested.

The random forests model, on the other hand, performs much more stably between training and testing sets and appears to benefit even more considerably than the other classifiers from the action-grounding of features, offering a $\sim 10\%$ performance increase in said case.

The action-grounded feature set, therefore, appears to induce superior performance over the non-grounded feature set in all classifier comparisons across both training and test sets in this experiment, which is encouraging, though as is discussed later in Section V, more extensive testing would be necessary before more general conclusions can be drawn.

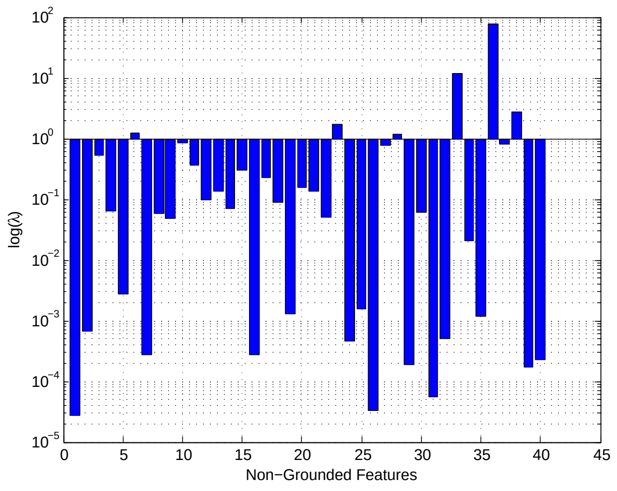


Fig. 8. SRNG feature relevance results for the non-grounded features dataset. Results are shown with a log scale.

2) Feature Relevance: Figures 8 and 9 show average feature relevance bar charts as derived from the SRNG classifier over the 10 10-fold cross-validation trials for the non-grounded and action-grounded feature sets respectively. Perhaps the most prominent result here is the more dis-

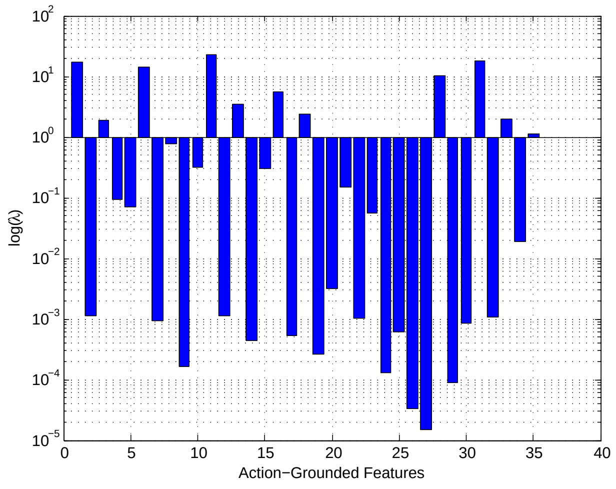


Fig. 9. SRNG feature relevance results for the action-grounded features dataset. Results are shown with a log scale.

tributed spread of different features that show significance in the case of the action-grounded dataset. Of the non-grounded features shown in Fig. 8, only two of the push action features, those being the y-coordinate of the object contact point and the x-parameter of the push line fit, as well as the z-coordinate of the z-axis bottom side part from the 3-D features, show significant relelvance. This matches with intuition as to what might make good predictors given the affordance classes involved in this experiment, since these features, crucially, encode both push height and planar direction, necessary for distinguishing topples vs. translations and left vs. right rotations respectively.

It is crucial to note here that SRNG outperforms all the other classifiers in test cases (cf. Table III) because it is able to single out these important features owing to its in-built feature relevance determination mechanism. Action-grounding spreads this information out via intrinsic encoding across the feature set, thus allowing the other classifiers lacking such a feature relevance determining component to also benefit from it.

V. CONCLUSIONS AND FUTURE WORK

To conclude, in this paper, we have presented an experimental comparison between action-grounded and non-action-grounded features derived from 3-D point clouds of objects. The experimental results demonstrated that action-grounded features can be effective for scenarios like object affordance learning by showing increased performance over similar non-grounded features across a range of state-of-the-art classifiers.

With regard to future work, we would like to explore the possibilities of using different point cloud divisions, parts-based structures and potentially part-hierarchies, beyond what has been presented here with bipartite axis splits. It would also be interesting to investigate the use of different shape features within the parts and sub-parts. From the affordance learning perspective, we also aim to implement regression capabilities that would allow for continuous prediction of object and object part positions.

Although the results presented here are encouraging, it is also difficult to draw broader conclusions about the generalisation capabilities of action-grounded features from this one study alone. Therefore, we hope to expand on this study in a longer-form journal publication with more objects, affordances, and action types, as well as implementation on a humanoid robot platform.

REFERENCES

[1] J. J Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, 1979.

[2] E. Sahin, M. Cakmak, M. R Dogar, E. Ugur, and G. Ucoluk. To afford or not to afford: A new formalization of affordances toward affordance-based robot control. *Adaptive Behavior*, 15(4):447, 2007.

[3] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini. Learning about objects through action-initial steps towards artificial cognition. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, 2003.

[4] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory-motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26, 2008.

[5] D. Omrčen, C. Boge, T. Asfour, A. Ude, and R. Dillmann. Autonomous acquisition of pushing actions to support object grasping with a humanoid robot. In *Proceedings of the 9th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 277 –283, December 2009.

[6] Barry Ridge, Danijel Skočaj, and Aleš Leonardis. Self-supervised cross-modal online learning of basic object affordances for developmental robotic systems. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5047–5054, Anchorage, USA, May 2010. IEEE.

[7] M. Kopicki, S. Zurek, R. Stolkin, T. Morwald, and J. Wyatt. Learning to predict how rigid objects behave under simple manipulation. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5722 –5729, May 2011.

[8] Stephen Hart and Roderic Grupen. Intrinsically motivated affordance discovery and modeling. In *Intrinsically Motivated Learning in Natural and ArtificialSystems*, pages 279–300. Springer, 2013.

[9] Emre Ugur and Justus Piater. Emergent structuring of interdependent affordance learning tasks. In *IEEE Intl. Conf. on Development and Learning and on Epigenetic Robotics*. 2014.

[10] M. Bjorkman, Y. Bekiroglu, V. Hogman, and D. Kragic. Enhancing visual perception of shape through tactile glances. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3180–3186, November 2013.

[11] E. Ugur, E. Sahin, and E. Oztop. Self-discovery of motor primitives and learning grasp affordances. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3260 –3267, October 2012.

[12] E. Ugur, E. Oztop, and E. Sahin. Goal emulation and planning in perceptual space using learned affordances. *Robotics and Autonomous Systems*, 2011.

[13] Tucker Hermans, Fuxin Li, James M. Rehg, and Aaron F. Bobick. Learning contact locations for pushing and orienting unknown objects. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, 2013.

[14] Tucker Hermans, Fuxin Li, James M. Rehg, and Aaron F. Bobick. Learning stable pushing locations. In *Proceedings of the Third Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob)*, Osaka, Japan, August 2013.

[15] B. Ridge and A Ude. Action-grounded push affordance bootstrapping of unknown objects. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2791–2798, November 2013.

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

[17] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[18] T. Kohonen. *Self-organizing maps*. Springer, 1997.

[19] B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002.

[20] B. Hammer, M. Strickert, and T. Villmann. Supervised neural gas with general similarity measure. *Neural Processing Letters*, 21(1):21–44, 2005.

[21] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Constrained Planar Cuts - Object Partitioning for Point Clouds

Anonymous CVPR submission

Paper ID 1976

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 to separate parts, concave connections are nevertheless indicative of inter-part boundaries. In this work we use a re-

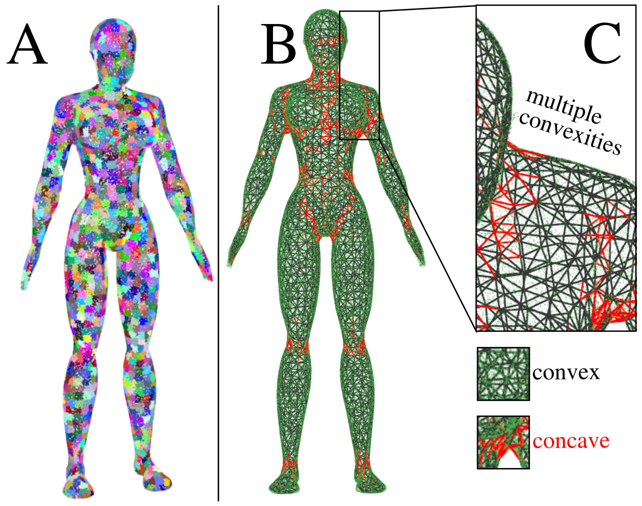


Figure 1. In complex objects parts are often only partially separated by concavities. **A)** Input object together with extracted Supervoxels. **B)** Supervoxel adjacency graph with convex/concave edge classification. **C)** Magnification of the shoulder showing how parts are not always strictly isolated 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.

laxed 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 partitioning 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

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

with

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

Likewise, a connection is concave if

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

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, that is, other segments are ignored. Cuts do not necessarily bi-section segments (as most graph cut methods), but as we

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

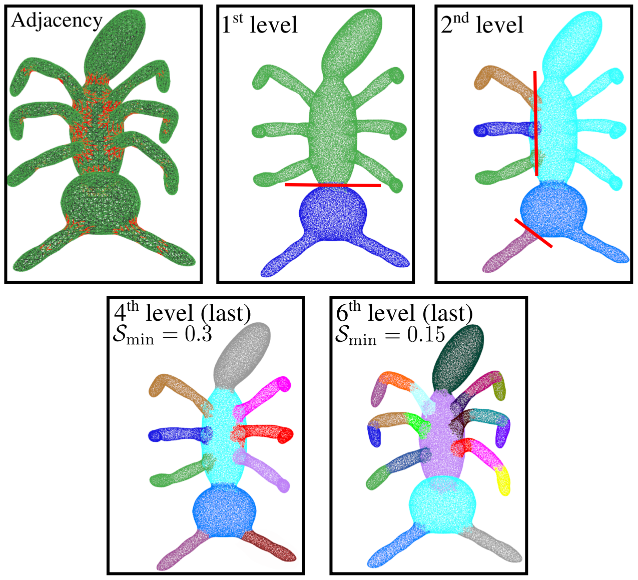

Adjacency
1st level
2nd level
4th level (last)
$\mathcal{S}_{\text{min}} = 0.3$
6th level (last)
$\mathcal{S}_{\text{min}} = 0.15$

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.

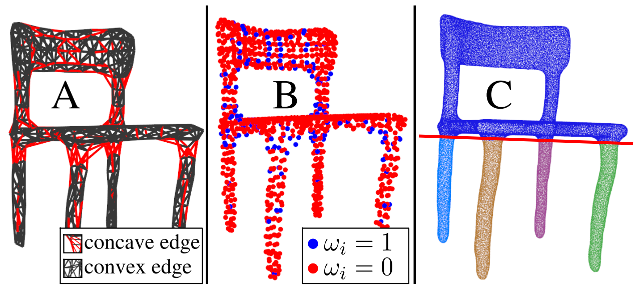


Figure 3. A chair from the Princeton Benchmark. **A:** Adjacency graph. **B:** Euclidean edge cloud extracted from the adjacency graph together with color-coded point weights ω_i. **C:** The first euclidean planar cut splits off all 4 legs, with concavities from each leg refining the cut's model.

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, that is, the vector connecting the supervoxels,

$$\vec{d} = \frac{(\vec{x}_2 - \vec{x}_1)}{||\vec{x}_2 - \vec{x}_1||_2} \tag{4}$$

together with the angle α between the normals of both supervoxels $(\vec{n}_1, \vec{n}_2)$:

$$|\alpha| = \cos^{-1}(\vec{n}_2 \cdot \vec{n}_1). \tag{5}$$

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:

$$\mathcal{S}_m = \frac{1}{|\mathcal{P}_m|} \sum_{i \in \mathcal{P}_m} \omega_i. \tag{6}$$

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:

$$\omega(\alpha) = \mathcal{H}(\alpha - \beta_{\text{thresh}}) \tag{7}$$

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. 6. 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. 4. To favor cutting edges with a plane that is orthogonal to the edge, we add a term to the scoring of concavities:

$$\mathcal{S}_m = \frac{1}{|\mathcal{P}_m|} \sum_{i \in \mathcal{P}_m} \omega_i t_i \tag{8}$$

$$t_i = \begin{cases} |\vec{d}_i \cdot \vec{s}_m| & i \text{ is concave} \\ 1 & i \text{ is convex.} \end{cases} \tag{9}$$

The notation $\cdot$ refers to the dot-product and $|x|$ to cardinality or absolute value. The idea behind Eq. 9 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. 8 to operate on the local clusters instead of $\mathcal{P}_m$:

$$\mathcal{S}_m^n = \frac{1}{|\mathcal{P}_m^n|} \sum_{i \in \mathcal{P}_m^n} \omega_i t_i. \tag{10}$$

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 (*i.e.* mode) 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] 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 $R_{\text{voxel}} = 0.0075$. We denoted the degree of supervision required for the algorithms using color codes (green: unsupervised orange: weakly supervised and red: supervised/learning). Unsupervised methods (such as ours) do not take model specific parameters into account and use

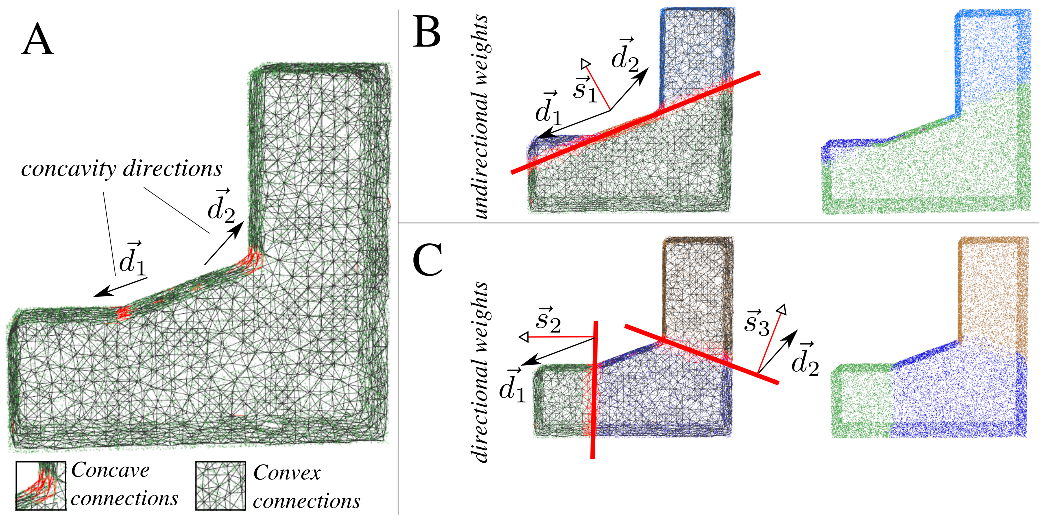


Figure 4. The highest scoring splits for undirectional and directional weights. **A)** Input object and adjacency graph. **B)** Using undirectional weights the best cut matches all concavities. However, this cut gets a lower score with directional weights due to the factors $|\vec{s}_1 \cdot \vec{d}_1|$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

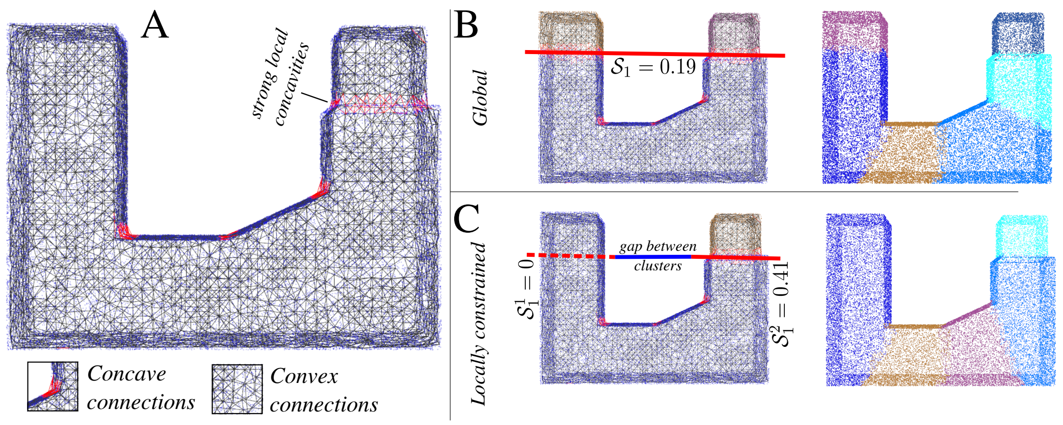


Figure 5. Comparison between locally constrained and global cuts. **A)** Input object and adjacency graph. **B)** Due to the strong local concavities on the right the algorithm will cut trough a perfectly convex part of the object (left). **C)** Using locally constrained cuts will find two clusters (along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

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 are depicted in Fig. 7. Additionally, we show Kinect for Windows V2 recordings from `http://www.kscan3d.com` in Fig. 8. We should emphasize that our algorithm does not require full scans of objects, that is, it can be applied to single views as shown in Fig. 8 E.

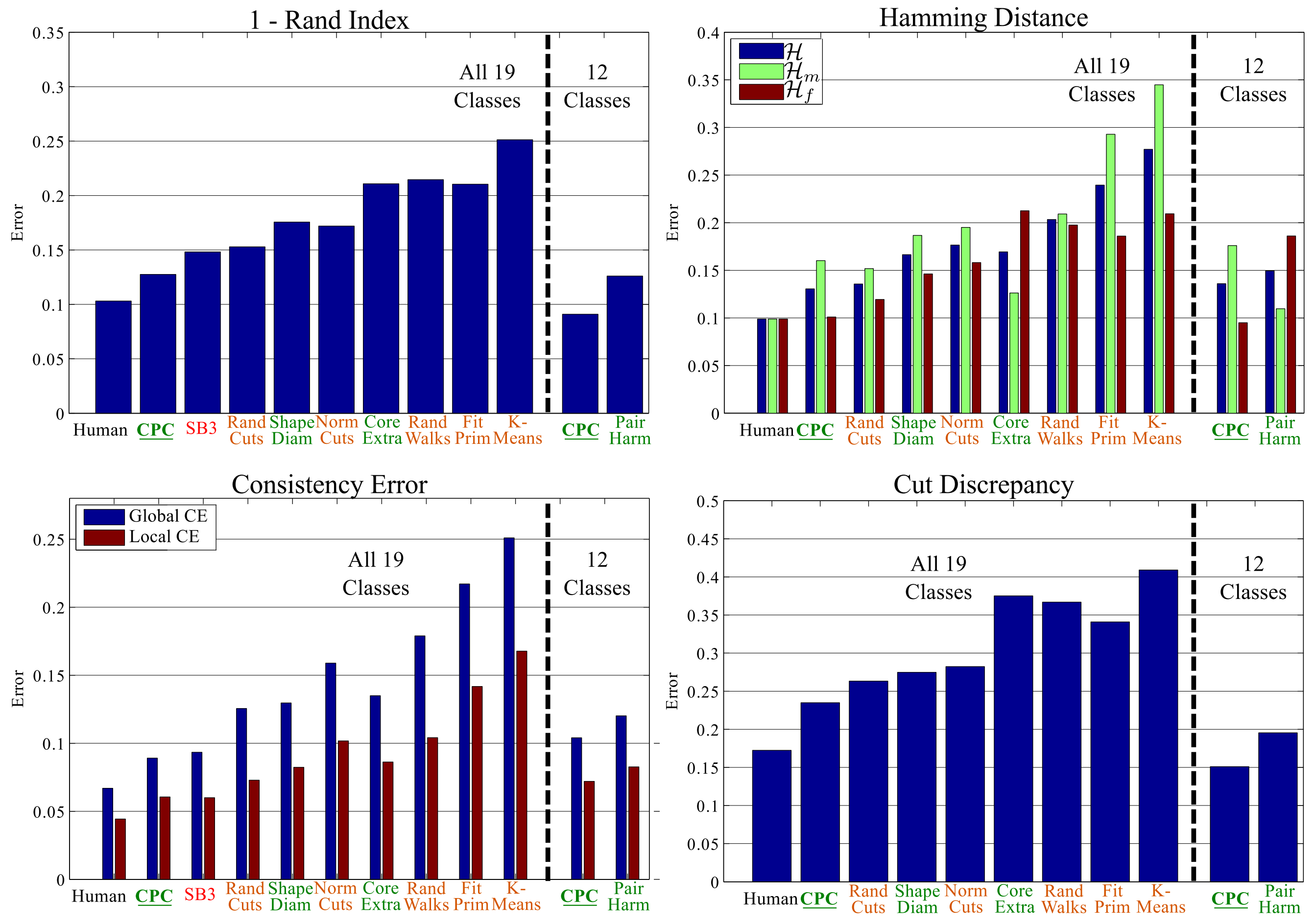


Figure 6. Comparison of proposed *CPC* algorithm to results published on the Princeton benchmark. Green algorithms are unsupervised, orange algorithms are weakly-supervised and red denotes supervised (*i.e*. training). For SB3 [9] results on some error measures had not been published. As Zheng *et al*. [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.

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
Fitting Primitives	4.6	0.210
Random Walks	1.4	0.214
K-Means	2.5	0.251

Table 1. Comparison of averaged computational time 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. 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 using 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.

Finally, we should emphasize that our method is learning-free, which has many advantages. Most impor-

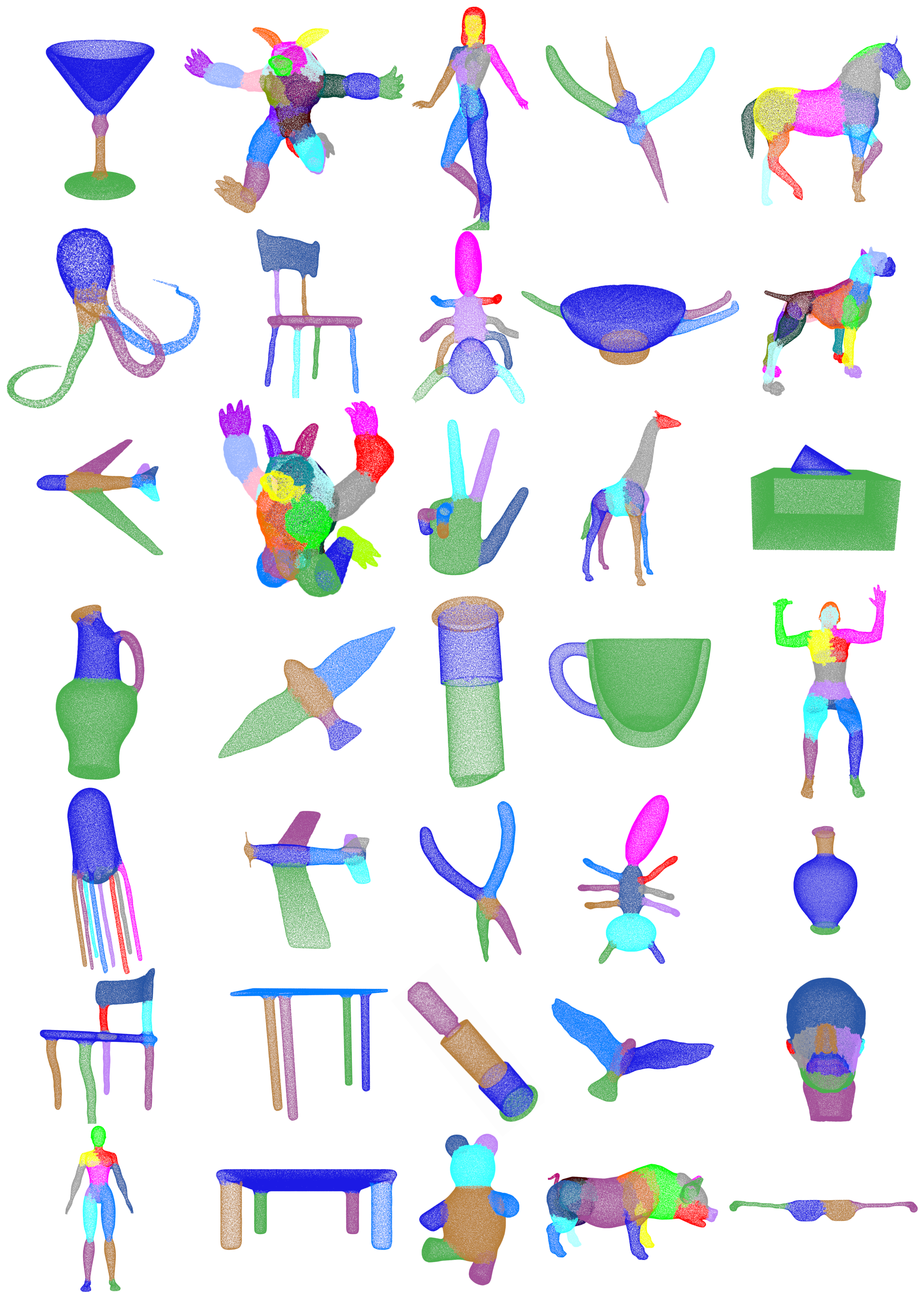


Figure 7. Qualitative results on the Princeton benchmark. All objects have been segmented with proposed algorithm using a single set of parameters ($\mathcal{S}_{\text{min}} = 0.16$, $N_{\text{min}} = 500$).

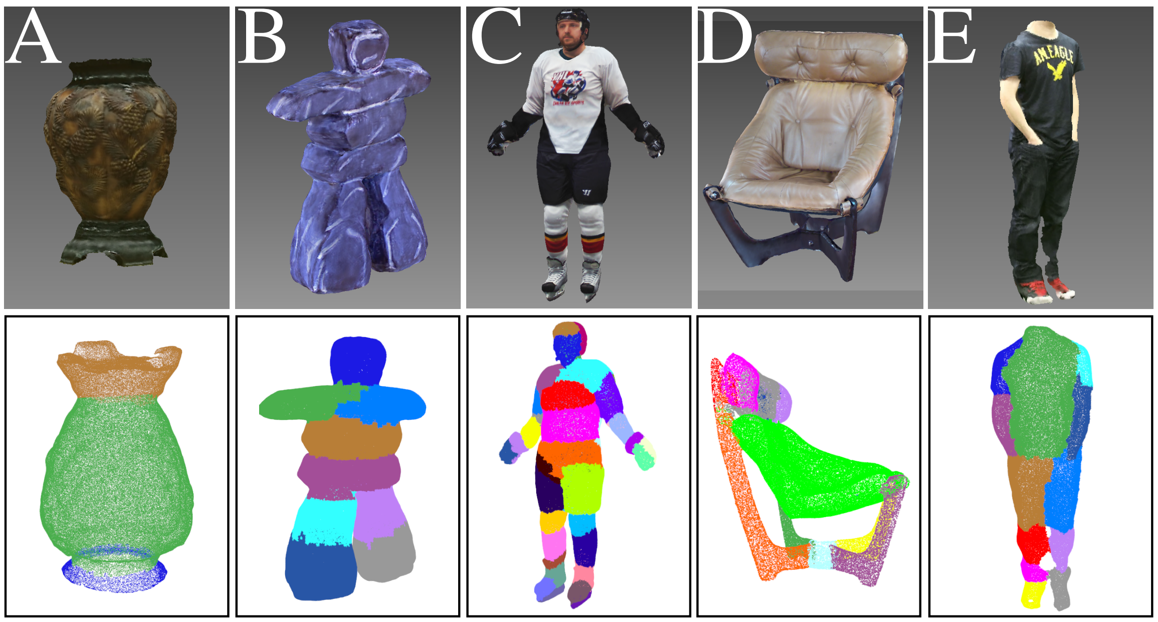


Figure 8. Qualitative results for the Kinect for Windows V2 gallery recordings from `http://www.kscan3d.com` using proposed algorithm. **A-D:** Full recordings. **E:** Single view recording.

tantly, 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. This means that the method is directly applicable as the first step in an automated bootstrapping process and can segment arbitrary unknown objects.

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

[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

Research Article **Open Access**

Mikkel Tang Thomsen*, Dirk Kraft, and Norbert Krüger

Identifying relevant feature-action associations for grasping unmodelled objects

Abstract: Action affordance learning based on visual sensory information is a crucial problem within the development of cognitive agents. In this paper, we present a method for learning action affordances based on basic visual features, which can vary in their granularity, order of combination and semantic content. The method is provided with a large and structured set of visual features, motivated by the visual hierarchy in primates and finds relevant feature action associations automatically. We apply our method in a simulated environment on three different object sets for the case of grasp affordance learning. For box objects, we achieve a 0.90 success probability, 0.80 for round objects and up to 0.75 for open objects, when presented with novel objects. In this work, we in particular demonstrate the effect of choosing appropriate feature representations. We demonstrate a significant performance improvement by increasing the complexity of the perceptual representation. By that, we present important insights in how the design of the feature space influences the actual learning problem.

Keywords: Human Vision, Affordance Learning, Cognitive Robotics

***Corresponding Author: Mikkel Tang Thomsen:** The Maersk Mc-Kinney Moller Institute, Faculty of Engineering, University of Southern Denmark, Niels Bohrs Allé 1, DK-5230 Odense M, Denmark, E-mail: mtt@mmmi.sdu.dk
Dirk Kraft: The Maersk Mc-Kinney Moller Institute, Faculty of Engineering, University of Southern Denmark, Niels Bohrs Allé 1, DK-5230 Odense M, Denmark, E-mail: kraft@mmmi.sdu.dk
Norbert Krüger: The Maersk Mc-Kinney Moller Institute, Faculty of Engineering, University of Southern Denmark, Niels Bohrs Allé 1, DK-5230 Odense M, Denmark, E-mail: norbert@mmmi.sdu.dk

1 Introduction

Identifying sensory features indicating action affordances is a crucial problem to be solved by cognitive agents since it allows for the identification of "action opportunities". A fundamental problem is the design of the perceptual feature space in which affordances emerge. This space can make the problem rather trivial (e.g., in case features that have a strong link to specific affordances are already provided). It can be also very difficult, when the link between affordances and actions can only be established by a high order combination of simple features (e.g., on the pixel level as in [1]).

In this paper, we investigate grasp affordances which are triggered by visual features of different order (see Fig. 1a), different granularity (see Fig. 1b) and semantic abstraction (see Fig. 1c). We are aware that the feature space we span is still of much lower complexity than what the human visual system provides in the occipital cortex. However, we investigate variation along three important dimensions of this feature space as further discussed in section 2.1.

In this paper, we introduce a method for finding feature-action associations in a complex visual feature space. The method for affordance learning described in the paper is not specific for a certain type of affordances, it can be in principal applied to any parameterizable action affordance. In this paper we however choose grasping as an example problem because of three reasons: First, due to the general importance of grasping. Second, we can simplify the learning problem by neglecting certain feature dimensions provided by the human visual system. For example colour can be ignored as a relevant dimension for grasping. In this paper, we also neglect 2D shape information, which however might already be a more questionable design decision. A third reason for addressing grasping is that there exists already relevant related prior work: In [2] (see Fig. 5), grasp affordances have been manually designed as first and second order relations of visual entities (local surfaces and 3D edges/contours). By that, we could already reach grasp performance of around 30% success. In [2],

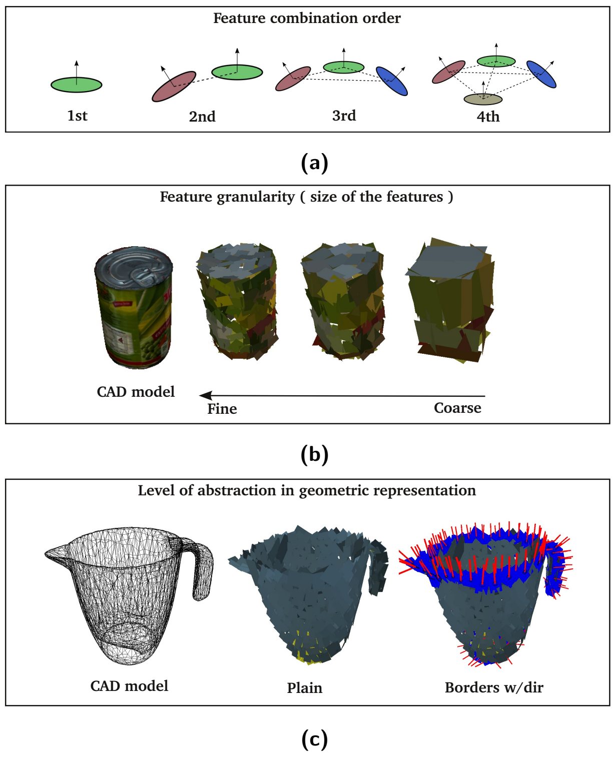

Feature combination order
1st
2nd
3rd
4th
(a)
Feature granularity (size of the features)
CAD model
Fine
Coarse
(b)
Level of abstraction in geometric representation
CAD model
Plain
Borders w/dir
(c)

Fig. 1. Overview of different aspects of the perceptual space that are investigated throughout this paper. In (a), it is shown how we can increase complexity to the perceptual representation by means of combining multiple features into more elaborated structures. In (b), it is shown how we can increase/decrease the complexity of the perception side by changing the size of the features. In (c) it is shown how the level of abstraction of the feature representation can be raised by means of semantic (here adding a boundary label and a boundary direction to a surface patch).

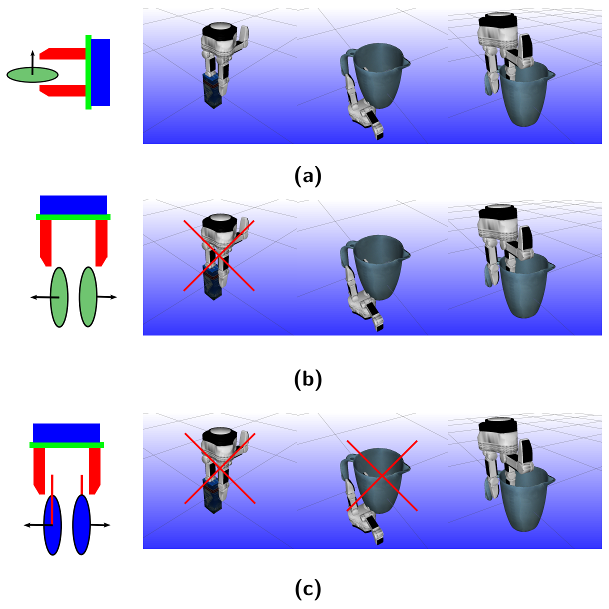


Fig. 2. Illustration of how different perceptual spaces can be used to limit the amount of grasp options. (a) shows a single feature grasp association which would not be able to distinguish between the three grasping situations on the left from which only the very left one leads to a success. (b) shows a second order-feature grasp association being rich enough to distinguish the left grasp situation as non successful. (c) shows a two-feature grasp association for which also the boundary direction (red line) is taken into account. This enriched features allows for distinguishing that only the very right situation leads to a success.

the grasp affordances however were defined "by hand" but in this paper, we aim at — besides improving performance — replacing such a manual design step by learning.

For this we want to explore the cross space of surface features and their combination, as shown in Figs. 1a–1c, and grasping actions. Fig. 2 shows how the variation of complexity of the input feature relates to the learning task. In Fig. 2a, left, we see a surface patch being related to a grasp. Learning grasp affordances with high success from this kind of weak feature is impossible: To exemplify this, we can imagine that the feature-grasp association would predict successes for all surface patches occurring in the scene. However in the three examples given, actual successes would only occur for the grasp at the right but not for the other two grasps shown in Fig. 2a. When we extend the feature space to second order combinations of surface patches (see Fig. 2b), the grasp on the left could be recognized as a non-successful one from the fact that no surface patch at the inside of the object is observable. However, it is impossible to recognize that the middle grasp cannot be successful. However, when we also add the concept of a boundary and its direction to the surface patch (see Fig. 2c), the system could be able to distinguish that only the right grasp can be successful in the given context. Similarly in this paper, we investigate the consequences for learning, when we vary dimensions of the feature space such as the order of features or their semantic content.

The algorithm we apply for that is a rather simple clustering method combined with a voting approach and part of the investigations is to explore the potential but also the limitations of such an approach. The complexities associated to our approach primarily stem from two sources:

Appropriate action bias: Non-successful actions are of limited usefulness for action affordance computation — although these can be used for sorting out non-interesting areas — and hence the system needs to be able to initially perform actions with a certain percentage of success likelihood. This can be achieved by intro-

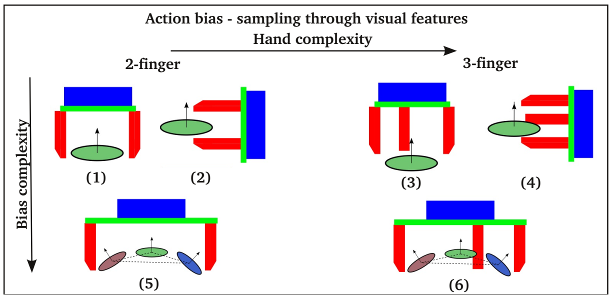


Fig. 3. An illustration of how different kinds of bias for grasping actions for a two or three finger hand can be defined. The space is defined by two measures of complexity. The first is the feature bias for the simple grasp reflex (based on either a single feature (1),(2),(3) and (4) or by multiple features (5) and (6)) and the second is the complexity of the manipulator here exemplified by a two and three fingered manipulator.

ducing action bias (see Fig. 3), e.g., by designing simple feature based heuristics that trigger actions with sufficient success likelihood (as in, e.g., [2]). In our case, we define rather weak biases that already lead to reasonable success likelihoods between 10–50% depending on the object class.

Feature space design: A further problem is to provide a feature space which covers features that are sufficiently correlated to successful actions. The feature space applied in this work does not provide feature coefficients that are independent. On the contrary, the feature space is highly structured: It provides geometric relations between surface patches which require appropriate parametrisations, careful choices of metrics as well as proper association of semantics.

Which features actually are relevant might depend significantly on the actual task and as we show most features are highly uncorrelated to action successes and therefore insignificant. The richer the visual space we provide, the more complex the learning problem will be, since then feature actions need to be found in a larger space. This holds in particular when feature relations of high order are computed since this will very quickly lead to a dimensionality which cannot be explored exhaustively anymore (dimensionality explosion). As a way to reduce the learning problem, the semantic content of features can be increased (as indicated in Fig. 1c). This however usually requires the introduction of additional heuristics and by that would jeopardize the genericness of the approach. In our work, we also need to deal with this trade-off. In particular we will show how the different design choices change the statistical distributions of particles in the feature space and by that also the structure and complexity of the learning problem. In this paper, we will describe how we approach the above mentioned complexities. We demonstrate how the affordance learning problem constitutes itself when important parameters such as the order of features, their granularity and their semantic complexity are varied.

In particular we show:

- That grasp affordance predictions comparable to the heuristically defined grasp affordances in [2] can be learned as second order combination of surface features. In that way, heuristics depending on the insight of the designer could be replaced by learning from experience.
- That the complexity of the feature space we span is of significant importance for the ability to learn affordances with a high rate of success. In particular we show that we can improve the quality of affordance prediction by combining multiple features and adding semantic information. By that we are able to identify grasp affordances for a set of different object types with a high likelihood of success.
- That suboptimal choices of feature representations lead to insufficient information to be considered as a good basis for grasp affordance learning.

This paper is structured as follows: We relate our work to the state of the art in grasp affordance learning and other relevant work in section 2. The problem formulation our approach is based on is outlined and formalised in section 3. The approach to address the problem domain is presented in section 4. In section 5, the experimental settings are explained, whereas the experimental results are presented in section 6. Finally the paper is concluded in section 7.

2 State of the art

In the following we will relate our work to state of the art, first in terms of the analogies to the primate's visual processing and second to the work within grasp affordance learning.

2.1 Analogies to the primate's visual processing

It is in general acknowledged that for humans, vision is a strong cue for affordance generation [3]. More than half of the primate's cortex is connected to visual tasks. As already pointed out in [4], the primate visual space is fundamentally of higher complexity compared to the action space. This in the first place concerns the dimensionality of visual information compared to a still rather low dimensionality of action parametrisation connected to the limited number of joints to be actuated.

The human visual system constitutes a deep hierarchy, covering a large number of complementary feature descriptors at different levels of granularity, different order and semantic abstraction (see Fig. 4 and [6] for a review of today's knowledge about the human visual system). More than 2/3 of the visual cortex (the so called "occipital cortex") is associated to task-independent feature processing displayed as yellow areas in Fig. 4. In these areas, a rich set of visual feature descriptors covering different aspects of visual information such as colour, 2D and 3D shape as well as motion are extracted. At least at early stages of processing, this is done in largely separated processing streams [6].

As shown in Fig. 4 and described in its caption, the level of abstraction of feature representation as well as the receptive field size increases (and by that the granularity of the features decreases) in this hierarchical process. As a consequence and as modelled than in our approach, the human visual system can search for affordances in rather different feature spaces ranging from, e.g., low-level 2D contrast information in retinal ganglion cells to 3D edge information with semantic association (such as border ownership [5]) in area V2. Moreover, it is not only the features themselves but their combination — as we will also investigated in our work — that provide relevant affordance cues (see Fig. 5b). From search tasks it is for example known, that feature combinations up to third order are computed in parallel in the human visual system, which results in so called "pop-out effects" in visual search tasks [7]. Hence, finding structures relevant for affordance programming in this high dimensional space at appropriate levels of granularity, order and semantic abstraction poses one of the major problems for affordance learning.

2.2 Related work on computing grasp affordances

Visual triggered action affordance learning is important for the development of cognitive agents. Within the grasping community typically an object is grasped to be further manipulated. However, affordance work like [8–10] take a more generic approach towards affordance learning, with the aim of finding what visual features afford actions.

In [10], visual triggered affordance learning was investigated, with the purpose of finding what visual 2D feature cues of an object afford graspability. A supervised learning approach was employed, where a robot interacts with an object to discover graspability and link it to extracted feature cues. A different approach is adopted in [9], were affordance cue's are extracted from inspection of human interaction. By identifying which areas of an object are occluded by the human during a grasp/action, it is learned what local areas of an object afford grasping, e.g., a handle.

In our work, we take a similar generic approach towards affordance learning, but while the authors of [9] learn object properties, e.g., graspability, we learn the coupling of visual features and actions, that enable a specific action. In that sense our work is more in line with the work in [8], where grasping points are learned from local visual descriptors, resulting in particular grasping points with associated probabilities.

Given the grasping application in our work, also approaches towards learning of grasping unknown objects are of interest. This topic has been extensively investigated due to its importance for robotic applications. For the problem of grasping unknown objects, two different strategies have generally been adopted, either feature based methods or shape based method. Examples of feature based approached are [2, 11–14], where a hand designed grasp hypothesis is proposed given a certain situation. These works stretch from grasp hypotheses based on a single or a combination of two simple features in [2] to grasp hypotheses based on a circle-fitting approach for cylindrical objects [14].

In contrast to feature based approaches, shape driven approaches like [1, 15–17], where the agent has a shape model in its database with associated grasps. The shape is matched to the new scene and in case a good match to a shape primitive is found, the grasps associated to this shape are performed. In [17], a set of prototypical object instances are captured with associated grasps from human demonstration and afterwards used for matching in novel situations. Other approaches like [16] and [15] approximates the object in terms of a oriented bounding box [16] or multiple bounding boxes [15] and then suggest grasp hypotheses based on the configuration of the bounding box. In a similar sense [18] decomposes an object into super quadratics to get an approximated object on which grasping can be performed. Another example of a model based approach is [19], where object shape, based on height maps extracted from 3D data and human demonstrated grasps, are learned and matched against new scene context.

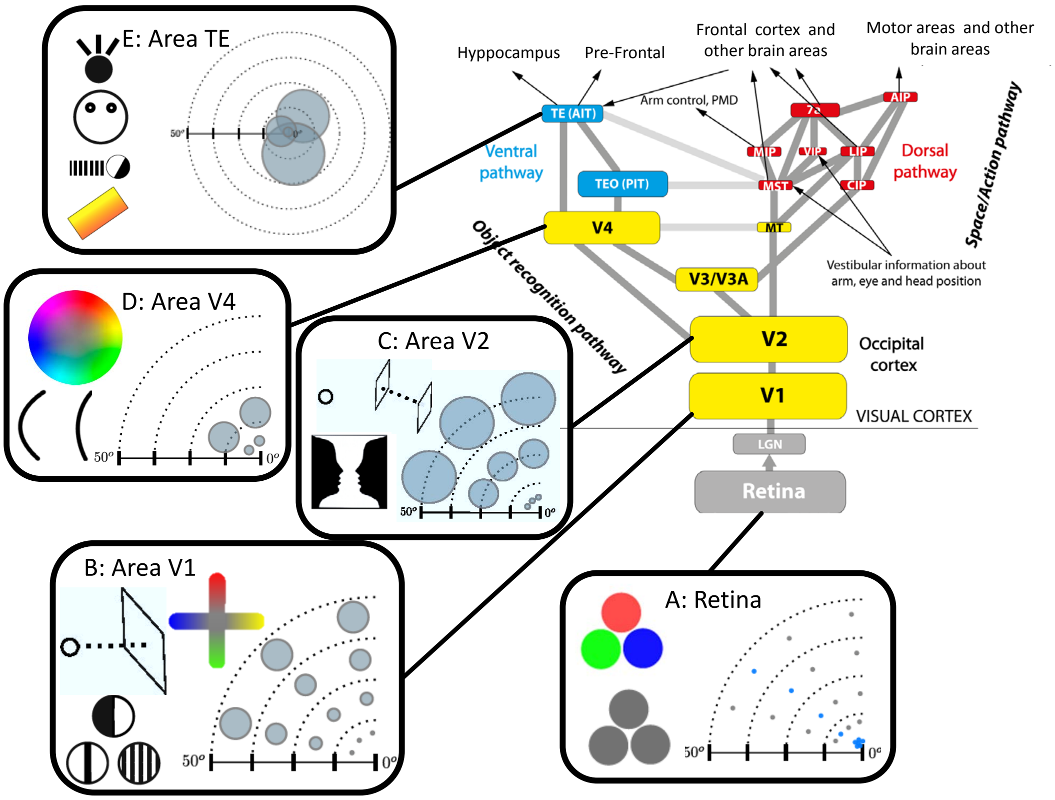


Fig. 4. The primate's visual cortex: The figure shows the deep hierarchical organization of the human visual system with the brain areas of the occipital cortex, the ventral and the dorsal pathway at the top right. For selected visual areas, the receptive field size of neurons as well as some of the features that are assumed to be processed in the specific areas are shown for the retina, area V1, V2, V4 and TE in the boxes a-e. The receptive field sizes are represented as they occur in the upper left quarter of the visual field in the right part of the boxes a-d. For area TE (box d), which has neurons with large receptive fields, the receptive fields are indicated in the whole visual field. Note that usually the receptive filed sizes are bigger for neurons representing the periphery of the visual field (which is very clearly visible in area V1 and V2). It is evident that the receptive field sizes also increase with the level of the hierarchy. Note for example the smaller receptive fields of V1 neurons compared to V2 neurons. Also the abstraction of features assumed to be processed at the specific levels (as indicated in the left part of the boxes a-e) increases. For example in rods and cones in the retina (box a) information similar to RGB pixel information in a camera while in area V1 edge information and a more advanced colour representation is applied. In V2 than even more abstract concepts such as border ownership [5] are computed. This figure uses material from [6] which we also refer to for further details.

Another branch within grasp affordance learning is the utilisation of a closed loop structure, by adding tactile feedback. By introducing tactile feedback from the finger contact–points, the stability of the grasp can be assessed before execution or replanning, hereby enabling a better chance of grasping successfully. In [20] it is shown how a grasp is planned based on an initial grasping pose acquired from rather simple vision and then evaluated by the tactile feedback before eventually a grasp or replanning is taking place. In another work based on haptics [21] it is shown how tactile feedback before grasp execution in combination with a predictor based on visual information can complement each other for grasp prediction. In [22] tactile feedback is utilised to refine knowledge of an unknown object, hereby enabling for planning a suitable grasp based on the acquired geometry. For a broader overview of the grasping domain see [23], where data driven grasp synthesis of known, familiar and unknown objects are surveyed extensively, including some of the work mentioned here.

Our work is a feature based approach, as we introduce simple feature constellation with associated actions, to be used for action prediction. The work can be seen as an extension to the work performed in [2], but with the advantage that we learn feature to action constellation by exploring different visual representations. In a recent work [24], it was in a similar way shown, how deep learning techniques were used to learn a feature representation suitable for learning grasp affordances, as compared to a previous work with a hand designed

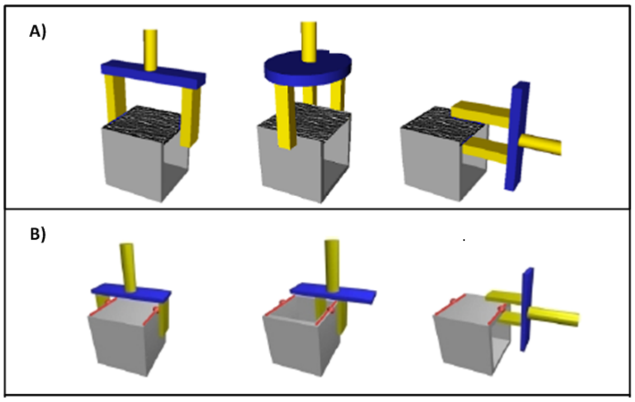


Fig. 5. Simple manually defined grasps: (A) Grasp affordances defined with respect to a single 3D surface feature (hence defined in respect to a first order feature relation), (B) Grasp affordances defined with respect to two 3D contours (hence defined in respect to second order feature relation). Source [2].

feature representation [25]. In contrast to [24], in our work we provide some kind of hierarchy to the learning algorithm which can then pick out promising candidates from this hierarchy. However, as discussed in the next paragraph, our approach can be seen as a step toward the learning of a deep hierarchy.

The focus on the underlying visual representation also links to work in non action domains, namely the work by the group of Ales Leonardis on learning hierarchical representations [26]. In this work, visual hierarchies are built up layer by layer. Each higher level entity is a combination of usually three elements of a lower level, where such combinations represents a certain spatial arrangement of simpler features. The selection of such combinations is done unsupervised for lower levels of the hierarchy based on, e.g., the criterion of frequency of occurrence and in an supervised fashion at higher levels. Our work can be understood as a step towards such hierarchy building, since relevant particles derived in this paper (see equation 4) are also spatial constellations of simpler entities which could be used as input to a higher level of a deep hierarchical structure. Different from Leonardis' work, we however apply 3D entities instead of 2D entities and we also have task specific evaluation criteria already on rather early levels of processing.

3 Problem description and formalisation

The main topic we investigate throughout this paper is the cross-space between perceptual features and actions. We explore how different aspects of the visual representation can provide relevant information for predicting action affordances in a reliable way.

3.1 Formalisation

To be able to perform these investigations, we initially formalise the building blocks, that we will utilise throughout the paper. The general space we are working in is a cross-space of perception and (grasping) action. We represent the perception side using 3D surfling features. 3D surfling features describe small surface patches in terms of a pose. In addition, we introduce a granularity measure that depicts the size of the features. Based on the previous description, we formalise 3D surfling features as $\mathbf{\Pi}^{\sigma} = \{SE(3)\}$ (see Fig. 6a). σ depicts the granularity level for the feature. The granularity is measured in the number of sub-features that a 3D surfling feature relies on and hence is a measure of the surface area it covers. $SE(3)$ depicts the 6D pose of the feature described in the Special Euclidean Group, $SE(3)$, hence the name.

With the description of the basis 3D surfling feature on the perception side, we introduce the concept of feature relations. Feature relations are essentially a combination of multiple features (3D surflings) described through their spatial and/or perceptual relationship, that allows for a set of higher level features.

One motivation for introducing the concept of feature relations is to compensate for the ambiguity in the 3D surfling feature pose, because the pose is derived from a principal component analysis of the underlying sub features (see Figs. 6a and 6b). The result is an unambiguous surface normal, but the other components in the orientation are ill defined. Hence we need other means to define a stable orientation of a 3D surfling feature.

By introducing feature relations, we add information through the spatial relationships between features, which theoretically will compensate for the uncertainties in the original pose. Moreover, we gain local structure information when we combine multiple features and hence achieve a more expressive visual representation. By means of feature relations, we create a representa-

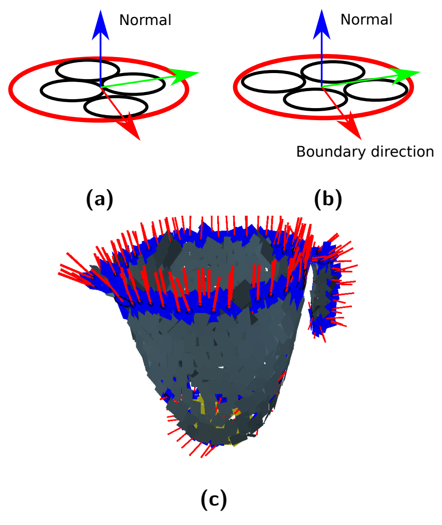


Fig. 6. Visualisation of the two basic building block. (a) a 3D surfling, Π^{σ}, where a principal component analysis is performed on the sub-features (black) to decide the orientation. (b) a boundary corrected 3D surfling, $\Pi^{\sigma,\beta}$, where the orientation is decided by the direction of a boundary. In (c), we see both boundary 3D surflings, blue with a red arrow, and standard 3D surflings.

tion where we can derive robust structures for predicting action affordances despite the simplicity of the basic building blocks. A complementary approach to tackle the issue of pose ambiguity in the basic building block is to introduce a more elaborated or expressive feature by additional levels of semantic. A boundary feature is introduced, where the pose is decided by the direction towards a given boundary. The boundary surfling is described by $\mathbf{\Pi}^{\sigma,\beta} = \{SE(3)\}$, where β denotes it is a boundary surfling and by definition, the first axis of the pose-frame is directed towards the boundary, see Figs. 6b and 6c.

Based on these basic 3D surfling features, we introduce a notation used for feature relations in equation 1,

$$\mathbf{\Upsilon}^{\sigma}_{N} = f(\mathbf{\Pi}^{\sigma}_{0}, \mathbf{\Pi}^{\sigma}_{1}, ..., \mathbf{\Pi}^{\sigma}_{N-1}) \qquad (1)$$

where N denotes the number of combined features, also referred to as the order of the relation, and σ denotes the granularity of the features it relies on. The function f transfers a combination of features into a parametrisation depending on the order and abstraction. To exemplify the transfer, we will describe a feature relation of second order based on generic 3D surflings (an illustration of such feature relations is shown in Fig. 7) which is parametrised as described in equation 2. The angles α_1 to α_3 and distance d_1 are defined as depicted in Fig. 7, whereas the feature relation coordinate system

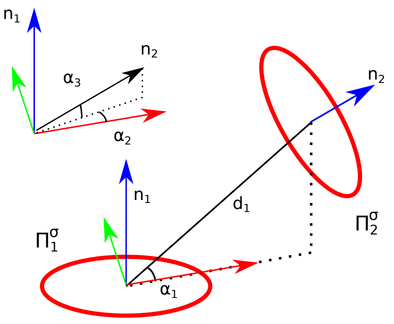


Fig. 7. Example of a feature relations of order two. It should be noted how the angles α_2 and α_3 describe the normal of the second feature Π^{σ}_{2} in terms of the coordinate system of the first feature, Π^{σ}_{1}.

is described in world coordinates.

$$\mathbf{\Upsilon}^{\sigma}_{2} = f(\mathbf{\Pi}^{\sigma}_{0}, \mathbf{\Pi}^{\sigma}_{1}\} = \{SE(3)^{P}_{W}, \alpha_1, \alpha_2, \alpha_3, d_1\} \qquad (2)$$

3.2 Action representation

Until now, we have not covered the action side of the perception × action space that we want to investigate. For this, we introduce grasping actions as an example. We define a minimalistic grasping action as follows:

$$\text{Action}_{\text{Grasp}} = \{SE(3)^{A}_{W}, E\} \qquad (3)$$

which essentially describes a target action pose in world coordinates $(SE(3)^{A}_{W})$ and an evaluation of the grasp outcome (E). The evaluation can theoretically take any value, but for the grasping case in this paper, we utilise a binary description. Other parameters such as preshape joint angles of the gripper could also be added to get a more elaborated action description.

3.3 Linking perception and action

In the final step, we link the perception part with the action part. Instances of the combined representation will be referred to as *particles* and denoted ρ as depicted in equation 4 and described in a condensed form using ρ's with superscript A (for action) and P (for perception) respectively.

$$\rho_i = \{\rho^{P}_{i} \times \rho^{A}_{i}\} \qquad (4)$$

A linked particle based on the previous examples of perception, equation 2, and action, equation 3, is presented in equations 5 to 6, where $SE(3)^{A}_{P}$ is a condensation of the poses from the different domains into a single pose, where the action is described in terms of the coordinate system of the perception side. In Fig. 8, an illustration of a particle is shown for two different levels of perception.

$$\rho = \{SE(3)^P_W, \alpha_1, \alpha_2, \alpha_3, d_1, SE(3)^A_W, E\} \quad (5)$$

$$\rho = \{SE(3)^A_P, \alpha_1, \alpha_2, \alpha_3, d_1, E\} \quad (6)$$

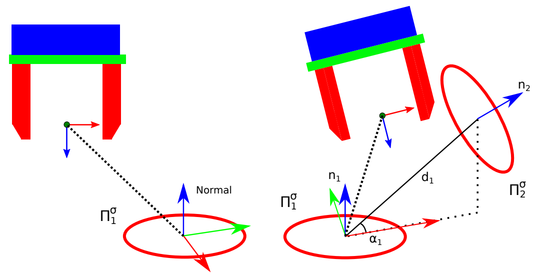


Fig. 8. Illustration of the linkage between action and perception for the first order case (left) and the second order case (right), essentially being a linkage (the dotted line) between the frame of the perception descriptor and the frame of the action.

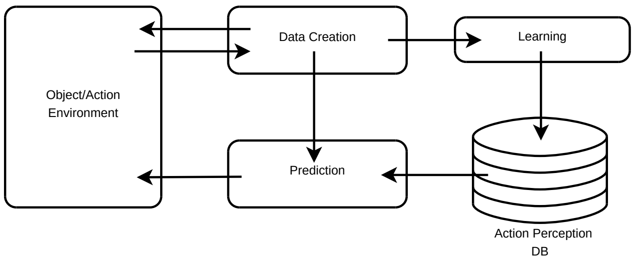


Fig. 9. Overview diagram of the data creation, learning, storing and prediction of action affordances. The data creation and learning process is addressed in Figs. 10 and 11, whereas the data creation and prediction process is covered in sections 4.3.1 and 4.3.2.

4 Learning algorithm

In this section, the algorithm for learning and applying the visually predicted action affordances will be explained. An overview of the process is shown in Fig. 9. The figure covers the steps from the Object/Action environment through a data-creation process, a learning process of which the results are stored in an Action Perception database, and finally a prediction step where the knowledge is used to predict actions to be performed in the Object/Action environment. In the following subsections, the different components shown in the overview diagram will be covered. First we describe the data creation process, (section 4.1), next the learning phase will be explained (section 4.2) and finally the utilisation of the learned knowledge for predicting actions will be described in (section 4.3).

4.1 Data creation

The data creation process is relying on the formalism defined in section 3.1, where the two domains, action and perception, are combined. From the Object/Action environment, we acquire evaluated action information as well as visual information in terms of extracted 3D surfling features, for training set objects. From features, we compute feature relations and then link the two domains together such that the action is defined with respect to the feature combination (see equation 6).

The procedure for doing the linking process is explained in algorithm 1. Note, that for every particle, ρ, a random action and feature relation is chosen and combined into a particle. The random selection is introduced due to the intractability of exhaustively combining feature relations and actions. In the combination step, additional constraints such as, e.g., locality (the action target pose should be close to the feature relation pose) could be added. A fundamental part of the

Alg. 1: Combining feature relations with actions.

```
**Input**: FeatureRelations $\rho^{\mathbf{P}}$, Actions $\rho^{\mathbf{A}}$
**Output**: Particles, $\rho$
1 N ; // Number of particles we use
2 i = 0;
3 while $i < N$ do
4     $\rho^A_j$ = random $\rho^{\mathbf{A}}$;
5     $\rho^P_k$ = random $\rho^{\mathbf{P}}$;
6     $\rho_i = \{\rho^P_k \times \rho^A_j\}$;
7     $\rho$.push_back($\rho_i$);
8     i++;
```

data creation process is the input actions. Such actions could be provided from various sources, e.g., real world experiments, simulation, hand labelled data or through human demonstration. The desirable properties of the input actions are that they provide a reasonable coverage and success rate for a given situation. In this work, we approach the data creation with a simulated environment that allows for a more explorative approach as compared to real world experiments. We utilise visually extracted surfling features as a bias for proposing a input action set. In Fig. 3, a number of examples are shown of how features can act as a bias for proposing candidate actions for the grasping case. That said, the action candidate creation is likely to be very dependent on the type of action. The input actions are then evaluated in simulation. Hereby we retain some control over the amount of input actions while we also can guide the rate of success.

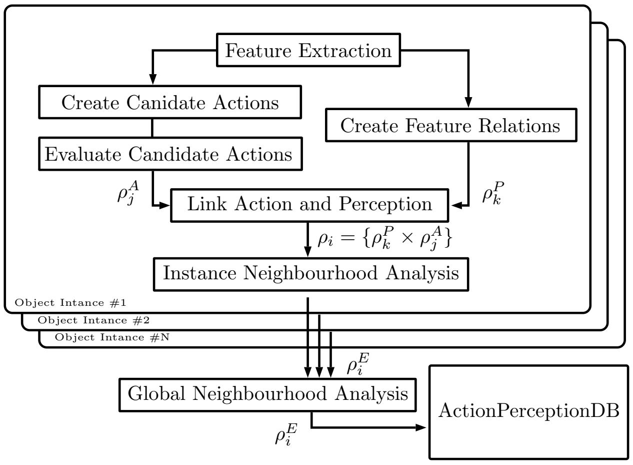


Fig. 10. Overview of the learning process, note the two-stage neighbourhood analysis, initially on instance level and finally on the combined set. On the instance level, the diagram explains how the visual features are first extracted, then utilised as a bias for the candidate grasp creation and finally evaluated. Secondly, the features are used to compute feature relations. Given the computed feature relations, ρ^P, and evaluated grasps, ρ^A, these two are linked to form particles ρ. Then the instance level neighbourhood analysis is performed, before the global neighbourhood analysis is used to merge the acquired knowledge from the instances. Finally the result are stored in the *ActionPerceptionDB*.

4.2 Neighbourhood analysis

In this section, the foundation for learning will be described in terms of the different components. First the learning approach is presented, next a two-stage extension is introduced and finally an optimisation of the learning outcome is considered.

4.2.1 Algorithm outline

The overall outline of the learning process is depicted in Fig. 10. This illustration encapsulates the steps from the feature extraction, action creation to the establishment of an action perception database, in terms of particles ρ.

The core of the learning process is a neighbourhood analysis, which is illustrated in Fig. 11. The first step is to find the set of supporting particles in the neighbourhood, which is formally described by, $\mathcal{A}_\mathbf{k}$, in equation 7. Based on the set of particles, the two measures probability and support are computed. The support, s_k, is given as the size of the set inside the neighbourhood (equation 8) and the probability, P_k, is defined as the average success probability within the neighbourhood (equation 9).

As we will show in the result section, both variables are essential for the efficient prediction of affordances.

$$\mathcal{A}_\mathbf{k} = \{\rho_i | Dist(\rho_i, \rho_k) < \mathbf{t}\} \tag{7}$$

$$s_k = |\mathcal{A}_\mathbf{k}| \tag{8}$$

$$P_k = \frac{1}{|\mathcal{A}_\mathbf{k}|} \sum_{\rho_i \in \mathcal{A}_\mathbf{k}} E_i \tag{9}$$

Given these two measures, we have a description of the action perception space in terms of success-outcome likelihood and the support for this likelihood. The latter

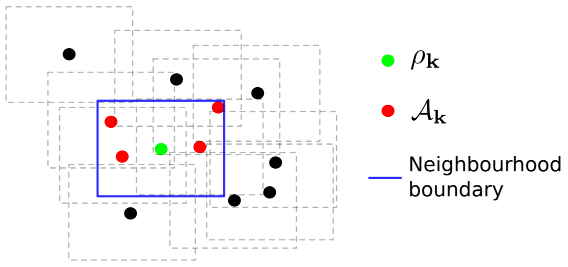


Fig. 11. 2D illustration of the neighbourhood analysis around a particle, highlighted in green.

can also be seen as the particle density in the neighbourhood. From a formal point of view, we go from particles in the form of equation 5 to *evaluated particles* of the form expressed in equation 10.

$$\rho_i^E = \{\rho_i^P \times \rho_i^A, P_i, s_i\} \tag{10}$$

The elementwise *Dist* function in equation 7, is used to decide whether the particle, ρ_k, is in the neighbourhood of ρ_i. For the distance computation, we split $SE(3)_P^A$, from equation 6, into a rotational part described by a quaternion $\mathbf{q}$ and a positional part (x, y, z) described by three components:

$$SE(3)_P^A = \{x, y, z, \mathbf{q}\} \tag{11}$$

The distance is computed in the individual dimensions of the parametrisation, with the exception of the orientation part of the $SE(3)_P^A$ pose, which is computed as the shortest angular distance between the orientation of ρ_k and ρ_i. Using a quaternion representation, the computation can be done with the formula in equation 12, where $\langle \mathbf{q_1}, \mathbf{q_2} \rangle$ depicts the inner product of the two quaternions $\mathbf{q_1}$ and $\mathbf{q_2}$. This approach ensures a well defined neighbourhood for the rotational part of the pose, as compared to a distance measure performed on the full quaternion parametrisation, which resembles the angular distance in a sub-optimal way. For the other parameters in the space this is not a problem. Hence for these a direct subtraction is used.

$$dist(\mathbf{q_1}, \mathbf{q_2}) = 2\arccos(\langle \mathbf{q_1}, \mathbf{q_2} \rangle) \tag{12}$$

In equation 13, the distance computation is expressed between two particles of the type described in equation 6.

$$\begin{aligned} Dist(\rho_i, \rho_k) = \{&x_i - x_k, y_i - y_k, z_i - z_k, dist(\mathbf{q_i}, \mathbf{q_k}), \\ &\alpha_{1,i} - \alpha_{1,k}, \alpha_{2,i} - \alpha_{2,k}, \alpha_{3,i} - \alpha_{3,k}, d_{1,i} - d_{2,k}\} \end{aligned} \tag{13}$$

It should be noted that the comparison operator (<) in equation 7 is an element wise comparison of the distance vector (see equation 13) and the threshold vector ($\mathbf{t}$). For it to be true, all the elementwise comparisons should be true.

The basic process for performing a neighbourhood analysis is captured by algorithm 2.

```
ALG. 2: Neighbourhood analysis.
  Input: Particles $\rho$
  Output: ActionPerceptionDB, $\rho_{\mathbf{DB}}$
1 $\mathbf{t}$ =Compute threshold;
2 for $\rho_k$ in $\rho$ do
3     $\mathcal{A}_\mathbf{k} = \{\rho_i | Dist(\rho_k, \rho_i) < \mathbf{t}\}, \rho_i \in \rho$;
4     $P_k = \frac{1}{|\mathcal{A}_\mathbf{k}|} \sum_{\mathcal{A}_\mathbf{k}} E_i$;
5     $s_k = |\mathcal{A}_\mathbf{k}|$;
6     $\rho_k^E = \{\rho_k, P_k, s_k\}$;
7     $\rho_{\mathbf{DB}}.push\_back(\rho_k^E)$;
```

The decisive parameter when doing a neighbourhood analysis is the choice of "neighbourhood" or vicinity, expressed as the threshold vector $\mathbf{t}$ in equation 7. We propose two options for setting the threshold, $\mathbf{t}$, a manual choice and an automatic choice. Using a manual approach to set the parameters involves setting a fixed threshold of each individual dimension based on common sense and then enable a scaling of the fixed parameter vector $\mathbf{t}$ by a scalar multiplier, M_m (see equation 14).

$$\mathbf{t}_{manual,M} = M_m \mathbf{t}_{manual} \tag{14}$$

The automatic setting is based on a rule of thumb from Kernel Density Estimation. Scott [27] proposed such a rule (see equation 15). The estimated threshold or bandwidth, t_{scott} is depending on the number of instances in the data, n, the dimensionality of the space, d, and the estimated standard deviation of the data-points within the dataset, $\hat{\sigma}$. It should be noted that the dimension of the vector $\mathbf{t}$ and $\hat{\sigma}$ depend on the parametrisation used for the particles ρ.

$$\mathbf{t}_{scott} = n^{\frac{-1}{d+4}} \hat{\sigma} \tag{15}$$

We can then use Scott's rule as a guideline for the ratio between the distances in the different dimensions. To adjust the neighbourhood-distance, we introduce an additional scaling parameter, M_s, similar to the multiplier mentioned for the manual defined threshold.

$$\mathbf{t}_{scott,M} = M_s \mathbf{t}_{scott} \tag{16}$$

In the Appendix, a comparison of an automatic- versus a manually set threshold is carried out. Here it it becomes apparent, that there might be a gain in prediction performance by choosing an appropriate manual threshold. Although there is a little gain, it is unlikely that the effort is worth it, especially when considering even more advanced visual representations of higher dimension.

4.2.2 Two-stage neighbourhood analysis

As displayed in the overview diagram (see Fig. 10), the neighbourhood analysis is performed in a two-stage process. This is motivated by the urge to decrease the computation time. The cost for performing the neighbourhood analysis is related to the number of particles (see equation 4), n, due to reliance on the KD-tree data structure. The computation cost for performing a search query in a KD-tree is $O(\log n)$, where n is the number of nodes in the tree, and when we take into account that we need to perform a search for every n particles to find the neighbours, the computational cost adds up to $O(n \cdot \log n)$. We can reduce the computational complexity by decreasing the amount of particles on which we are performing the neighbourhood analysis.

In an initial stage, we perform a neighbourhood analysis on the particles from the individual objects in the full dataset. This partitioning provides us with a set of significantly smaller neighbourhood problems, instead of a single large problem. Having a set of smaller problems, that are independent, we also facilitate a parallelisation of the first stage. By filtering the output particles of the first-stage before performing the second neighbourhood analysis on the combined problem, we can drastically improve the computational time. One way of filtering away “un-promising” particles, is to set up a criteria for the minimum support that a particle should have for it to be taken into account. Such a filter could be expressed in absolute, average or median values of the support in the dataset. There are however some pitfalls when using support as a filtering parameter, namely the risk for filtering away the diversity in the particles. This aspect of the learning is addressed in the results (section 6.4), where different levels of support filtering has been applied to verify the effect on the prediction outcome. In practice, an introduction of support filtering in the neighbourhood analysis includes a small extension that removes particles below a certain support threshold for the final dataset.

4.3 Prediction

In order to apply the learned data in novel situations, two different methods have been applied. One method where we look for similarities on the perception side and use these as direct cues for proposing actions denoted as “direct action proposition” and secondly a method, denoted as “voting scheme”, where we suggest a candidate list of actions from the *ActionPerceptionDB* to vote for the actions. The two approaches will be explained in the following subsections.

4.3.1 Direct action propositions

The direct action proposition approach is based on the assumptions, that our learned high probability and high support action perception particles are descriptive enough for predicting actions. Initially we extract feature relations, the ρ^P part of the particles, from the novel object and search for similar ρ^P parts in the *ActionPerceptionDB*. If we find a similar perception part with a high probability for success and high level of support, we take its action part, ρ^A, and attach to our ρ^P part, resulting in a proposed action.

Given the simplicity of the direct action proposition approach, it has some limitations. The main problem is, that the approach relies heavily on a discriminative perceptual representation in order to make reliable predictions. The potential problem arises when we use a too simple perceptual representation, namely that a particular simple relation can predict very different actions depending on the object it was learned from. This problem should eventually disappear if we utilise a more descriptive perception representation. Therefore we introduce a second approach, the voting scheme. For comparison, experiments have been carried out with the direct action proposition method (see Appendix A), where the prediction performance and limitation in the method are presented.

4.3.2 Voting scheme

The principle behind the voting scheme is that we want to utilise our learned *ActionPerceptionDB* as a means to vote for a set of candidate actions. Hereby we utilise multiple perception descriptors to predict the action outcome of a single candidate action, and by that improve the robustness of the prediction. The voting procedure has been formalised in algorithm 3. The process is very similar to the actual learning phase, however where we in the learning phase “forget” the origin actions when we combine them with the perception part, ρ_P, we remember them in the voting scheme. This allows for a final step in which we can project a prediction probability back to the origin candidate action, and thereby give a prediction based on multiple perception action particles. In Fig. 12, an example is presented, where we utilise multiple feature relations (Figs. 12d to 12g), to vote for a single candidate action (Fig. 12h).

5 Setting

In this section, the settings for the experimental work will be explained. It involves the object data set (section 5.1), the simulation environment (section 5.2), the feature extraction (section 5.3), the visual biased action sampling (section 5.4) and details regarding action and perception parametrisation (section 5.5).

```
ALG. 3: Voting scheme.
    Input: ActionPerceptionDB $\rho_{\mathbf{DB}}$, Features
    Output: Candidate Actions with prediction $\rho^{\mathbf{A}}_{\mathbf{C,E}}$
1   $\rho^{\mathbf{A}}_{\mathbf{C}}$ = Create Candidate action through visual bias;
2   $\rho^{\mathbf{P}}_{\mathbf{C}}$ = Compute feature relations;
3   $\rho_{\mathbf{C}}$= Combine feature relations with candidate actions as in ALG. 1;
4   for $\rho_{C,k}$ in $\rho_{\mathbf{C}}$ do
5       $\mathcal{A}_{\mathbf{k}} = \{\rho_{C,i} | Dist(\rho_{C,i}, \rho_k) < \mathbf{t}\}, \rho_{C,i} \in \rho_{\mathbf{DB}}$;
6       $P_k = \frac{1}{|\mathcal{A}_{\mathbf{k}}|} \sum_{\mathcal{A}_{\mathbf{k}}} E_i$;
7       $s_k = |\mathcal{A}_{\mathbf{k}}|$;
8       $\rho^E_{C,k} = \{\rho_{C,k}, P_k, s_k\} = \{\rho^P_{C,k}, \rho^A_{C,k}, P_k, s_k\}$;
9   // Backproject probabilities to origin actions
10  for $\rho^A_{C,l}$ in $\rho^{\mathbf{A}}_{\mathbf{C}}$ do
11      $\mathcal{B}_{\mathbf{o}} = \{\rho^{A,E}_{C,i} | \rho^{A,E}_{C,i} == \rho^A_{C,l}\}, \rho^A_{C,i} \in \rho^{\mathbf{A}}_{\mathbf{C}}$;
12      $P_{avg} = \frac{1}{|B_o|} \sum_{\mathcal{B}_{\mathbf{o}}} P_i$;
13      $\rho^A_{C,l} = \{\rho^A_{C,l}, P_{avg}\}$;
14      $\rho^{\mathbf{A}}_{\mathbf{C,E}}.push\_back(\rho^A_{C,l})$
```

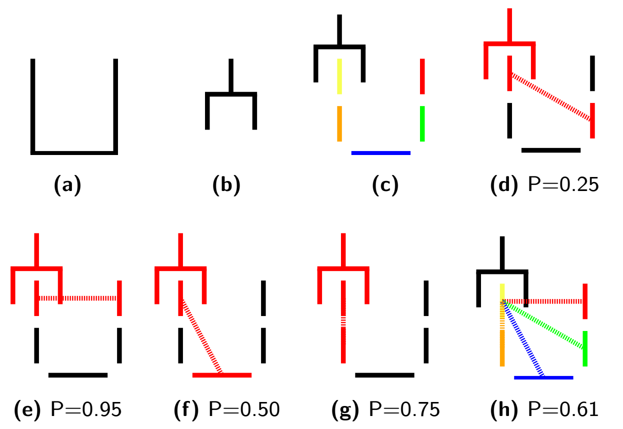


Fig. 12. A 2D idealised example illustration of the basic principle of the voting scheme given a candidate grasp. (a) An idealised cross-section of a 2D container, (b) a two-finger gripper, (c) a feature representation with a candidate grasp. Figures (d), (e), (f) and (g) show feature relations that are used to vote for the candidate action. Probabilities are shown below which would be the probabilities found in the database. Given the example probabilities, the combined probability for the candidate grasp is shown in (h).

5.1 Object set

In Fig. 13 an overview of the different objects used in the experiments is given. The objects are split into three different categories, namely box-like objects, curved/cylindrical objects and open/container objects. The objects in the set are partly taken from the KIT object database [28] and partly from the online database archive3D [29]. The KIT objects are digitalised real objects which potentially simplifies the transfer from a simulated environment to the real world. Furthermore they add realism to the feature extraction as the objects are textured based on the real objects. However due to the lack of open/container objects in the KIT set, we needed to extend the object set with objects from other sources, which are not digitalised real objects.

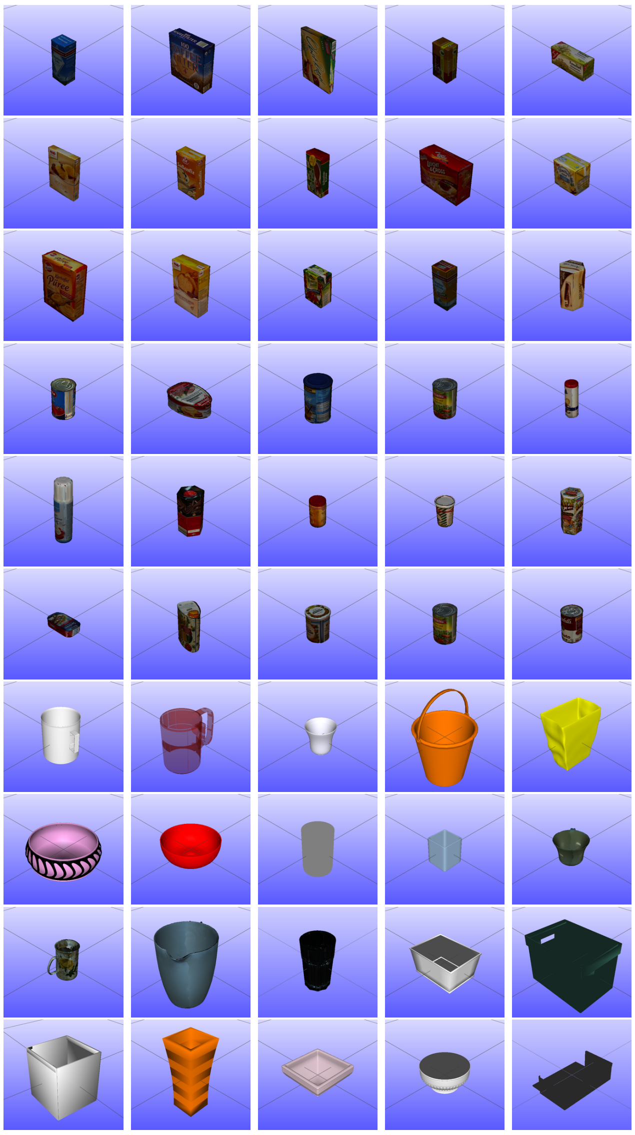


Fig. 13. Visualisation of the three different categories of objects. (Top), box objects, (middle), round objects and (bottom) open objects.

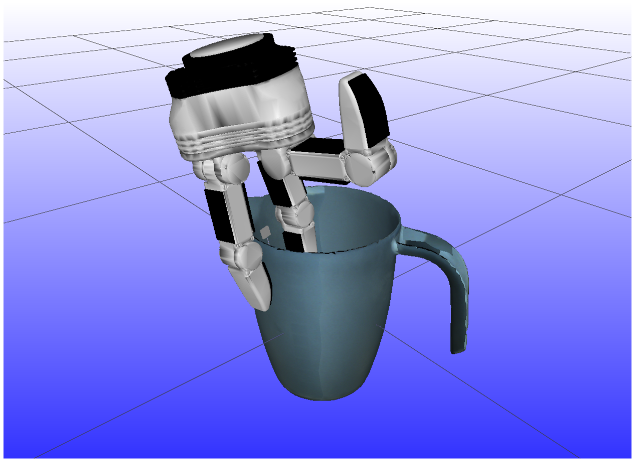


Fig. 14. Visualisation from RobWork showing a grasping action with the Schunk SDH-2 hand.

5.2 Simulation environment

The experiments in this paper are all performed in a simulated environment utilising the robotic library RobWork [30]. RobWork is used to create a realistic environment, that facilitates simulated sensors (such as RGB-D sensors and Stereo cameras) as well as simulation of dynamics. The dynamics simulation is carried out using the associated simulation environment, RobWorkSim [31]. Fig. 14 shows a view of a dynamic grasp simulation, with the Schunk SDH-2 hand and a pitcher. The grasping simulations are performed in a free-floating world where gravity is not taken into account since it facilitates grasping from every direction. It should be mentioned that although gravity is not taken into account other forces acting between the gripper and the object are simulated. The masses of the objects used in the simulations variy between 0.2 kg and 0.6 kg, estimated based on their size. The friction coefficient between the gripper fingers and the objects are set to $0.2\,\mu$ corresponding to the friction between rubber and plastic.

In Appendix A.3, a set of additional experiments — performed in a table scenario — are presented where gravity is taken into account. These results show a high degree of similarity to the results achieved in the free-floating scenario. Due to this similarity and since the focus of this work is on exploring the perceptual representations, we use the free-floating scenario in our experiments.

5.3 Feature extraction

An essential part of the setting is the feature extraction from the simulated environment. In Fig. 15, our setup of RGB-D sensors is displayed. Having a setup of three sensors surrounding the object and an additional sensor from below gives an approximated full view of the objects in the centre.

Although the simulated recording situation is not very common, there exist some robot set-ups with multiple cameras providing a rather complete scene representation (see, e.g., [32]). Our recording situation is somehow in-between dealing with full and perfect CAD models and the common recording context with one camera. Since humans have mechanisms to extract rather complete 3D representation in an observation context by either merging different views or by matching a complete representation existing in memory to a given scene context, one can assume that affordance reasoning in humans can make use of information beyond what is directly visible from one viewing direction. For example, the detection of complex features such as 'wall features' as second order relations of basic surface features is only possible when the inside and outside of an object is taken into account.

We are aware that this is a compromise between different possible options for a set-up. It allows for affordance reasoning on complete representations in which however still controlled amounts of sensory noise are present by the simulated RGB-D sensor as well as by the fact that the scene is observed by only a small set of viewing angles.

Based on the simulated setup in RobWork, we are able to extract the 3D surfling features at different granularities and with added semantic. An example of the feature extraction of surflings at four different granularity levels is visualised in Fig. 16. Furthermore the extracted features are shown both with and without the added semantic for boundary features. The boundary features are shown with an additional vector depicting the direction of the boundary.

5.4 Action sampling

The action sampling biased through the visually extracted features is a prerequisite for learning the grasp affordances in an automatic way since it ensures a reasonable chance of success as well as a limit to the amount of considered actions, see Fig. 3 for an overview of potential biases. We propose two template grasp types for

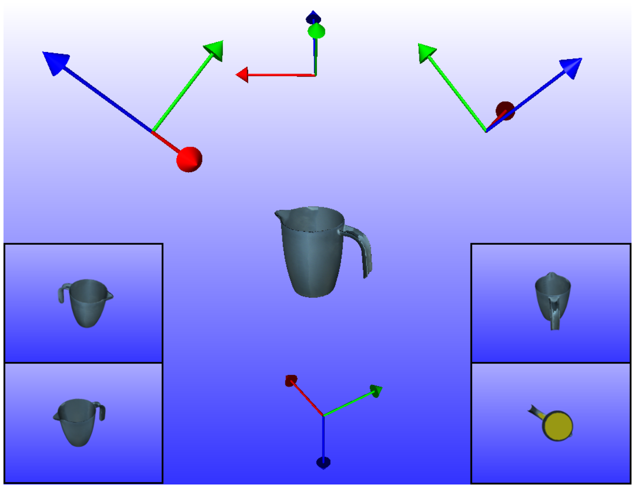
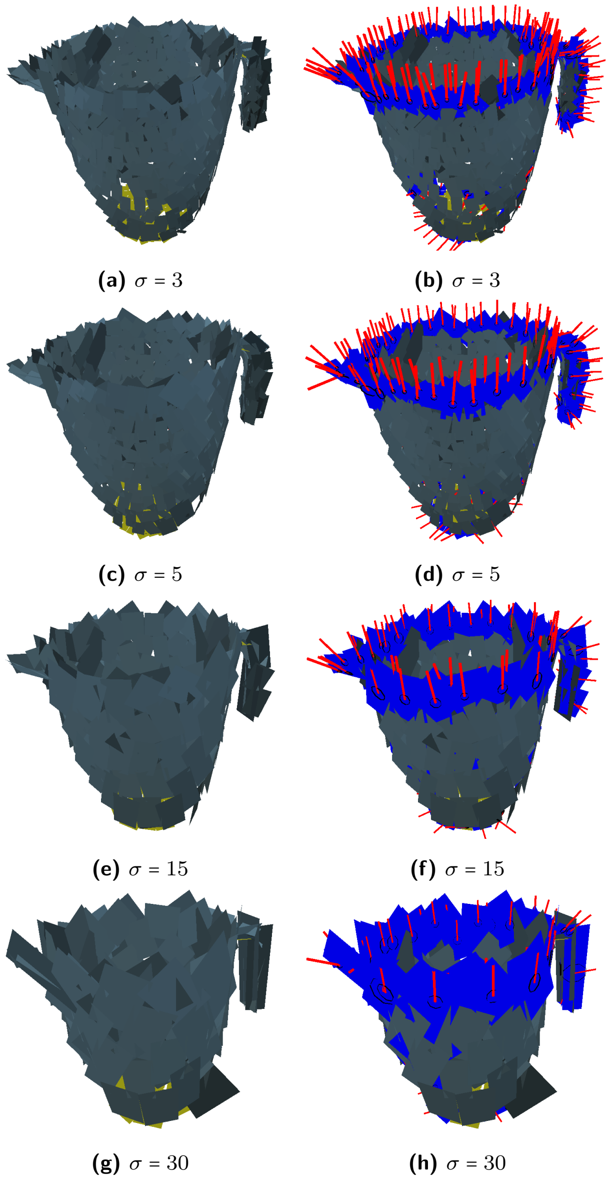


Fig. 15. Visualisation of the four simulated RGB-D sensor views, illustrated with the four coloured frames, and the object of interest in the centre. The frames depict the position and the camera-view (along the negative z-axis, coloured blue). The views from the four cameras are shown in the small images.



(a) $\sigma = 3$ (b) $\sigma = 3$ (c) $\sigma = 5$ (d) $\sigma = 5$ (e) $\sigma = 15$ (f) $\sigma = 15$ (g) $\sigma = 30$ (h) $\sigma = 30$

Fig. 16. Visualisation of extracted features at four different granularities with (right column) and without (left column) boundary semantic.

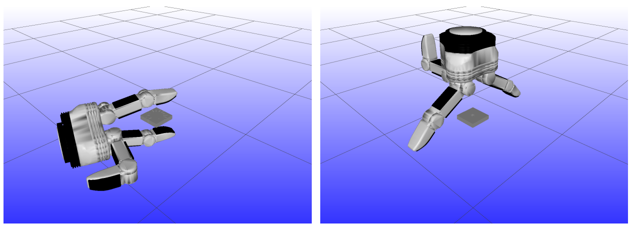


Fig. 17. Visualisation the two different basic grasp types, SidePinchGrasp (left) and TopGrasp (right)

the sampling. The two types are visualised in Fig. 17, one is denoted the SidePinchGrasp and the other is denoted TopGrasp. The SidePinchGrasp has a rather narrow opening between the two fingers such that it can grasp within a container and the TopGrasp has wide open fingers to make an encompassing grasp of larger objects. We create a set of candidate grasps by means of extracted 3D surfling features with a small feature size such that we can achieve a reasonable coverage of the objects. Based on the features, we propose a set of template grasps by rotating them in 32 steps around the feature normal. From this sampling we achieve an average success-rate between 10% and 50% depending on the object set (see the random chance as dashed horizontal lines in the results plots Figs. 21, 22 and 23).

5.5 Parametrisation of feature relations

Throughout the experiments, we will rely on a limited set of different feature relation types, namely of first and second order relation with different levels of boundary semantics. In equations 17 to 22 the different parametrisations are presented. The reason for limiting ourselves to first and second order combinations is partly due to the exponential combination explosion that our approach exhibit due to its simplicity. When utilising higher order combinations, three or more, the amount of possible combinations of visual feature and actions during the learning phase become intractable to cover exhaustively. This relates directly to the parameter spaces, that also increases and if we are not able to direct or limit the space, by for instance heuristics like the boundary feature, the sparsity becomes a problem.

$$\boldsymbol{\Upsilon}_1^{\sigma} = f(\Pi^{\sigma}) = \{SE(3)\} \tag{17}$$
$$\boldsymbol{\Upsilon}_1^{\sigma,\hat{\beta}} = f(\Pi_1^{\sigma,\hat{\beta}}) = \{SE(3)\} \tag{18}$$
$$\boldsymbol{\Upsilon}_1^{\sigma,\beta} = f(\Pi_1^{\sigma,\beta}) = \{SE(3)\} \tag{19}$$
$$\boldsymbol{\Upsilon}_2^{\sigma} = f(\Pi_1^{\sigma}, \Pi_2^{\sigma}) = \{SE(3), \alpha_1, \alpha_2, \alpha_3, d_1\} \tag{20}$$
$$\boldsymbol{\Upsilon}_2^{\sigma,\hat{\beta}} = f(\Pi_1^{\sigma,\hat{\beta}}, \Pi_2^{\sigma}) = \{SE(3), \alpha_1, \alpha_2, \alpha_3, d_1\} \tag{21}$$
$$\boldsymbol{\Upsilon}_2^{\sigma,\beta} = f(\Pi_1^{\sigma,\beta}, \Pi_2^{\sigma}) = \{SE(3), \alpha_1, \alpha_2, \alpha_3, \alpha_4, d_1\} \tag{22}$$

In Fig. 18 visualisations are shown of the different types of feature relations used in the experiments. Note that only four different feature relations are visualised. The reason is that the parameters for equations 17 and 18 are similar with the only difference being that we know the feature in equation 18 is a boundary feature. The same holds for the two cases in equation 20 and 21. The parametrisation covers three first order cases: one plain feature ($\boldsymbol{\Upsilon}_1^{\sigma}$), one where we know the feature is a boundary feature ($\boldsymbol{\Upsilon}_1^{\sigma,\hat{\beta}}$) and one were we utilise the boundary semantic with direction ($\boldsymbol{\Upsilon}_1^{\sigma,\beta}$). As for first

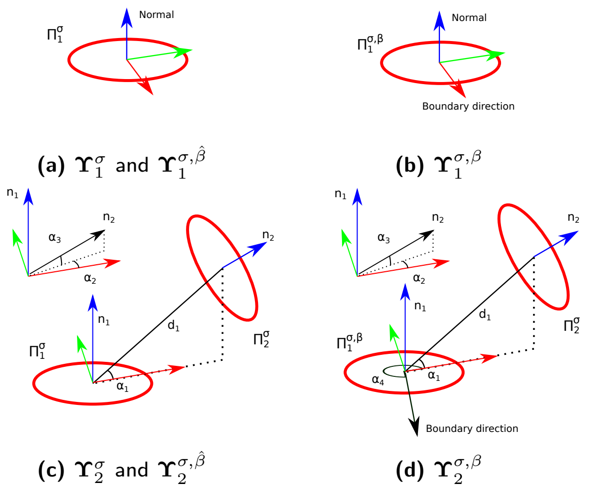


Fig. 18. Visualisation of the utilised feature relations and the associated parameters.

order, we introduce a parametrisation for three second order cases: One without semantic ($\boldsymbol{\Upsilon}_2^{\sigma}$), one with the knowledge of a boundary but not the direction ($\boldsymbol{\Upsilon}_2^{\sigma,\hat{\beta}}$) and finally one with boundary semantic and direction ($\boldsymbol{\Upsilon}_2^{\sigma,\beta}$).

6 Results

The result section is divided into four subsections. In section 6.1, we will present the outcome of the learning phase in terms of associated support and probability of the evaluated particles. In section 6.2, we will present the core results comparing the prediction performance when features at different granularities, different levels of abstraction and different semantics are input to the voting scheme. Subsequently (section 6.3), a qualitative analysis is presented of the results. Finally (section 6.4), we will present results regarding the impact of support filtering. In the experimental work, the different object sets have been split into two classes such that the learning from the first class and is applied on the second and vice versa. In the Appendix, a number of additional results are presented primarily focussing on methodology aspects such as automatic vs. manual threshold, the direct prediction method and table vs. free-floating simulations.

6.1 Learning outcome

In order to examine the learning outcome before it is used for prediction, we visualise the frequency of occurrence of the evaluated particles (see equation 6) in terms of support and probability. Fig. 19 shows the distributions in 2D histogram for the different parametrisations described in equations 17 to 22, where the colour depicts the frequency. The colouring is based on the log_{10} transform of the actual frequency in the area to allow for a visible distinction. A histogram corresponding to Fig. 19a but without performing a log_{10} transformation of the frequency is shown in Fig. 20 as a comparison. In this plot, we only see that the majority of the particles have low support and probability.

When assessing the 2D histograms in Fig. 19, we can acquire indications about the predictive power of the different visual representations. We see a shift towards the higher probability areas when the order is raised or semantic is added to the feature relation, e.g., compare Fig. 19a towards Fig. 19f. This change is reflected in the later presented prediction results (see Fig. 23).

6.2 Core experiments

The outcome of the voting method (section 4.3.2) is a set of candidate actions with associated predicted probability. To discretise these outcomes, which allows for

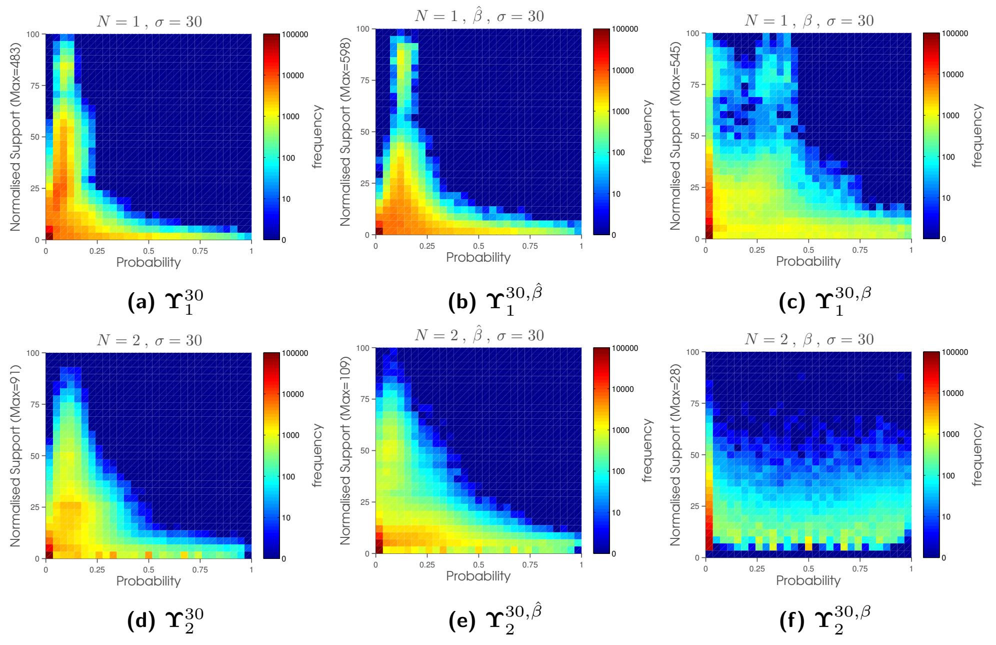


Fig. 19. Visualisation of the particle distribution for the open object set in terms of their support and probability for the learned *ActionPerceptionDB*'s. Note that the areas with high support and high probability in the action perception space increases when a more elaborated perceptual representation is used, e.g., compare (a) and (f). This change indicates that the latter representation has more prediction potential. The number of particles in the databases ranges from ~ 250,000 to ~ 400,000.

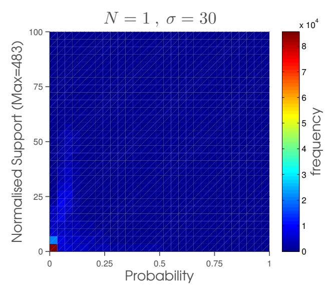


Fig. 20. Visualisation of the particle distribution in terms of support and probability for a learned *ActionPerceptionDB*, where the particle frequency is shown without any modifications.

a comparison to the binary grasp outcome from simulation and hence to quantify the performance, we introduce a probability selection threshold. We vary the actual value of the threshold between the extremes. This results in the plots in Figs. 21–23.

In order to assess the prediction results, we present two different average measures of the prediction success over the object set. In addition we present a measure of the percentage of grasped objects from the set

– **Avg-1** - An average computed over all the objects in the set, independent of whether feature combinations leading to any grasp prediction were found for a certain object. If no predictions was found the object contribute to the average with a success rate of zero. This average type is plotted with a full line.
– **Avg-2** - An average computed over the average success prediction for only the set of the object instances, where a prediction was found. This average type is plotted with a dashed line.
– **random** - The average chance on the object set for randomly getting a successful outcome given the candidate actions. This measure is plotted with a dashed black line.
– **Coverage** - A measure of the percentage of objects from the object set that have been grasped for a given selection threshold. This measure is shown with a dashed-dotted line.

When assessing the result plots, there are multiple aspects that one need to consider when we want to identify a good result. One aspect is the difference between random chance and the top point of the predictions, another is how well a change in the moving threshold to a higher value is reflected as a higher rate of success prediction. The result plots show in general a drop in the success-predictions after a top-point. The reason is that the amount of predicted grasps after the top-point drops drastically, because no or very few grasps are found with a prediction rate higher than the selection threshold at the top-point, enabling outliers to have a strong impact.

Finally one should note the ability to predict grasps for the full object set, which is covered by the percentage of objects that have been grasped by a certain selection threshold.

Box objects: The results for the box object set are presented in Fig. 21. The plots show results where the two dimensions "order" (denoted N, equation 1) and "feature granularity" (denoted σ equation 1), were varied.

From the results we derive: (1) When the order is increased from N=1 to N=2, we see a clear improvement of the prediction rates, by comparing the top-points of the six lines in the plot. This is explained by the added knowledge introduced by a more complex visual feature, and (2), when the feature size is changed, small changes in the performance are observed. For the first order case, we see the best performance with a medium sized feature whereas there is no or little difference when we compare the second order cases at different granularities. Variations based on the used feature granularity is related to the ability of a given feature size to represent the object with adequate accuracy. (3) The object set seem well covered as there are predictions for all objects until a selection threshold of 0.90, where a drop is seen for N=1, at granularity 15.

Round objects: The experimental results acquired for the round object set are shown in Fig. 22. As above, the plots show results where the two dimensions "order" and "feature granularity" where varied. We see: (1) When the order is increased from N=1 to N=2, a clear improvement is seen in the prediction rates specifically when observing the top-points of the plots. This is explained by the information gain from a second visual feature, and (2), when feature size is varied, we see small changes in the performance for the first order case, whereas we see a clear distinction in performance when we use the largest feature size for the second order case (brown line in Fig. 22). The last result is in line with the expected result, namely that a large surfling patch is a bad reflection of a round object and hence should be less descriptive as compared to a feature of smaller size. (3) The object coverage is in general good, as predictions are found for all the objects until a selection threshold of 0.8, where a drop is seen.

Open objects: The experimental results for the open object set are displayed slightly differently compared to the round and box object sets, since we observed that for open objects the semantic information in terms of boundary information is crucial. The introduction of boundary features allows for all the parametrisations described in section 5.5. The results are presented in Fig. 23 for three different granularities, respectively 5,

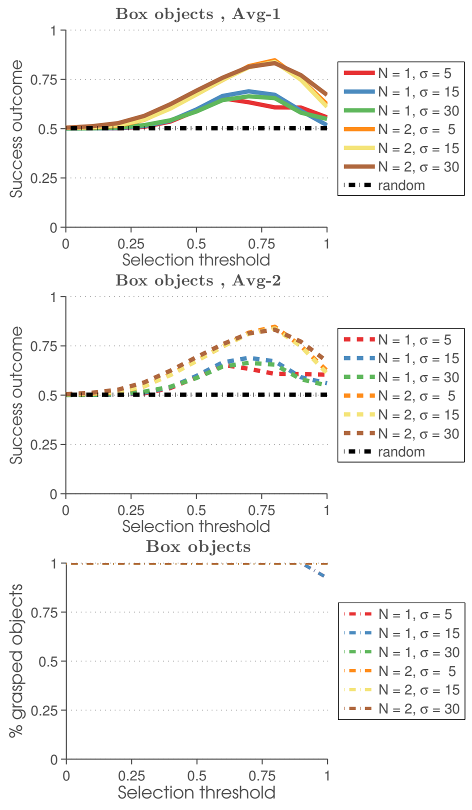


Fig. 21. Box objects prediction results. See equations 17 and 20 for the utilised parametrisation and see text for further details.

15 and 30. In each of the figures, results for the order and level of abstraction through semantic are shown. We see, that the higher order we use and the more semantic we add, the prediction results improve. An improvement is observed when we go to second order relations as compared to first order (see orange line (N=2) compared to the red line (N=1) in Fig. 23), however we do not see any significant improvement in the prediction power when we add the semantic of a boundary without direction (comparing red and blue lines, and comparing orange and yellow lines), although we have a better object set coverage as the full line is resulting in a higher success probability. A significant improvement of success prediction rating is however achieved for second order relations with boundary and direction (brown line). We see however a small drop when we reach the higher end of the selection filter. This can be explained with the fact that the voting method act as a smoothing operator hence high prediction areas will be in general occurring rarely. When we compare the results acquired for the different granularities, we see a similar outcome as in Figs. 21 and 22. The results for the percentage of grasped objects show some interesting patterns. In general the parametrisations with N=2, show a rather

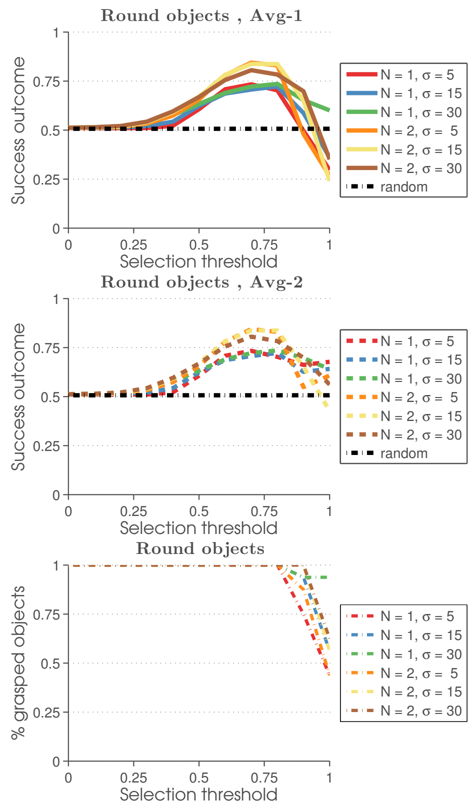


Fig. 22. Round objects prediction results. See equations 17 and 20 for the used parametrisation, and see text for further details.

good coverage with a percentage between 0.6 and 1.0. In particular the most elaborated representation (brown line) shows close to full coverage. The parametrisations with N=1 show in general less coverage as the generalisation is worse. For the parametrisation of N=1 with boundary and direction it is seen that even a the lowest selection threshold, only around half the objects are covered which tells that although we have a good prediction (dotted green line) the generalisation over the object is not convincing.

6.3 Qualitative analysis of the power of semantic information

In order to illustrate the performance gain we get when we introduce the boundary semantic, we present a visualisation of the *ActionPerceptionDB* for the three first order cases. The visualisations are shown in Fig. 24. In the centre, a surfling feature is placed and the coloured area around the feature represents how the actions are distributed with respect to the pose of the feature. The colour coding of the actions depicts the likelihood of success for that particular particle.

For $\boldsymbol{\Upsilon}_1^5$ we see a uniform distribution of success probability, whereas for $\boldsymbol{\Upsilon}_1^{5,\hat{\beta}}$ we see two rather uniformly coloured areas. Noticeable is an inner part with a higher success likelihood as compared to the outer part. This is explained with the added knowledge of the boundary, specifically by the fact that, at the boundary, a successful action will be closer to the feature, hence the inner circle captures both the successful boundary grasp as well as unsuccessful, whereas the outer part mostly capture the non-boundary action.

When assessing $\boldsymbol{\Upsilon}_1^{5,\beta}$, it becomes obvious what we gain by introducing the direction towards the boundary. The visualisation shows a high likelihood of success along the direction of the boundary and the further the grasp are located orientational wise from the boundary direction a lower success likelihood is observed.

To visualise how the power of semantic constitute itself when applied for predicting actions, a visualisation of the distribution of predicted grasps for an object is shown in Fig. 25. The figure shows the prediction result for a pitcher, where the order and level of semantic are varied. One can easily notice how the introduction of boundary and direction information for both first and second order cases allow for high success areas at the boundary of the pitcher.

6.4 Support filtering

In order to investigate the impact of the support filter, a series of experiments based on the open object set have been performed, in which the amount of particles used from the first stage of the neighbourhood analysis is varied. We filter by choosing the zeroth to the tenth decile of the particles based on their support, e.g., split the first decile lowest supported particles from the highest supported particles and then utilise the highest supported part. Hereby we cover the extreme situations, from using every particle to using very few. The acquired results are presented in Fig. 26. Note the support level is described as a measure between zero and 1.0.

From the results, three main points are derived:

(1) When assessing the results for **Avg-1** for the four cases, Υ_1, $\Upsilon_1^{\hat{\beta}}$, Υ_2 and $\Upsilon_2^{\hat{\beta}}$, the observed pattern shows, that a lower support filter results in higher success rate, although only at lower selection threshold. When comparing the results of **Avg-1** with **Avg-2** for the same four cases, it is noticed that a larger support level result in a higher success rate for the instances that are found. This is in particular seen for Υ_1 and Υ_2, as the selection threshold increases towards 1.0. This re-

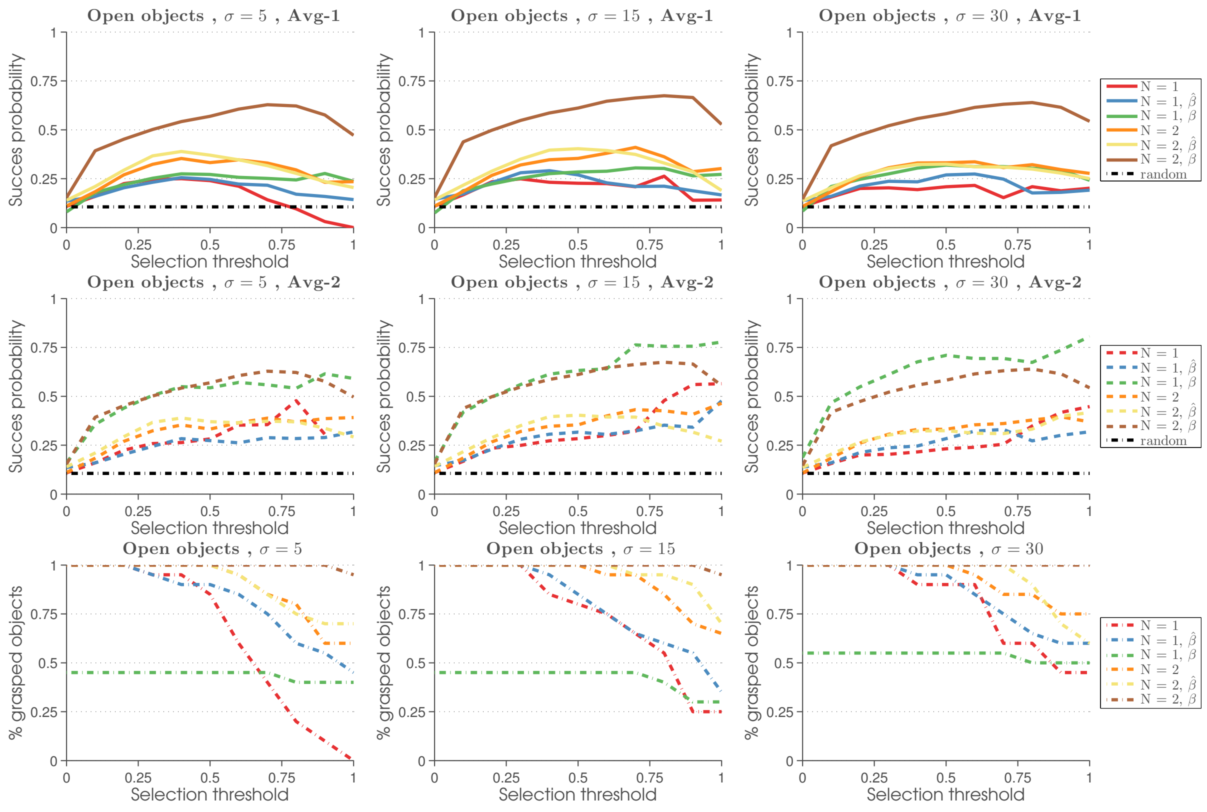


Fig. 23. Prediction result for open objects of granularity 5, 15 and 30. See equations 17–22 for the used parametrisation, and see text for further details.

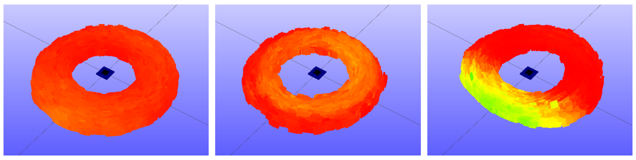


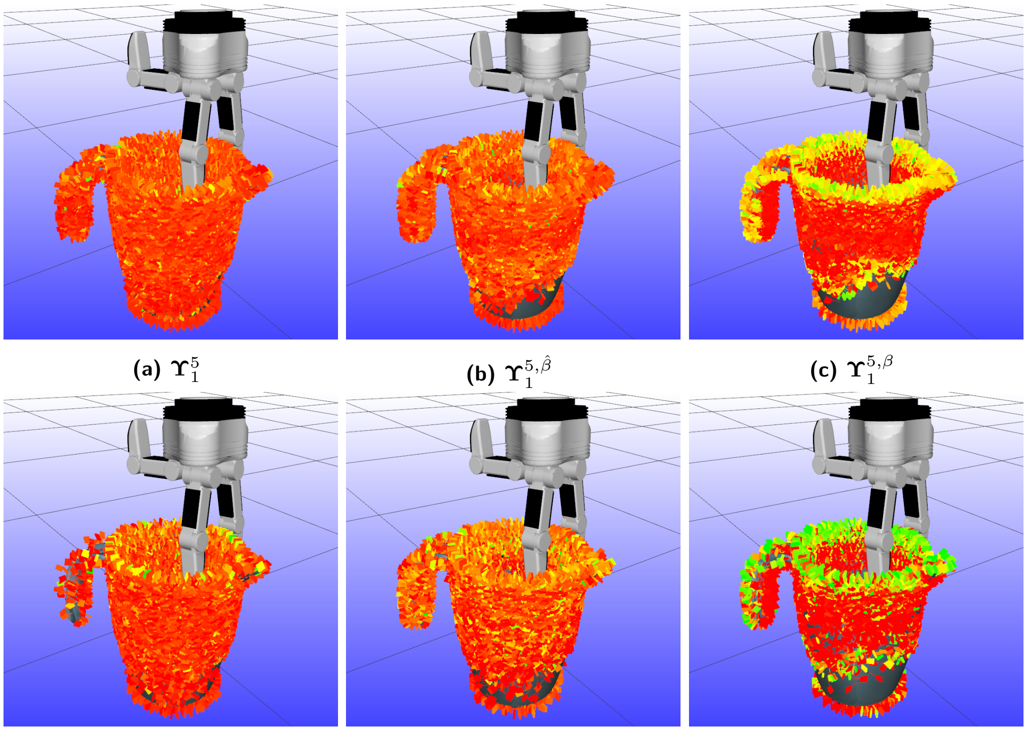
Fig. 24. The three visualisations show how the learned particles are distributed, when the feature part of the particles is positioned in the centre. The three cases are Υ_1^5 (left), $\Upsilon_1^{5,\hat{\beta}}$ (middle) and $\Upsilon_1^{5,\beta}$ (right). Red colour depict a success likelihood of 0.0 and green a success likelihood of 1.0.

sult indicates, that with a higher support level very good prediction for a subset of the objects can be derived.

(2) When assessing the Υ_1^{β} results the pattern is significantly different. For **Avg-1** the prediction results show similar performance independent of the applied support level, with the only exception being the highest support level, where the performance is degrading at a low selection threshold. The results for **Avg-2** show that if a prediction is found, then a higher success rate is achieved when a high support level is used.

(3) When assessing the results for Υ_2^{β} the recognised pattern for both the averages, **Avg-1** and **Avg-2**, show similar performance with a small advantage at the higher support levels. Especially at the two highest support levels, an improved performance is noticed. The reason for the improvement is related to the predictive power of the representation: If we are able to find particles of high support and high success probability, then when filtering for a high support level, we would keep these “good” particles.

To summarise the outcome of the support filter experiment, it can be observed that for the less elaborated feature representations, good predictions can be found for individual instances of objects at a high support level, whereas generalisation is in general not observed when utilising a lot of instances (a low support level). For the more elaborated visual representations, it becomes evident, that we are able to achieve an improved performance and still retain the generalisation when using a higher support level. This result indicates, that there indeed exists particular feature relations, which are predictive for grasping in the provided visual representation.



(a) Υ_1^5 (b) $\Upsilon_1^{5,\hat{\beta}}$ (c) $\Upsilon_1^{5,\beta}$

(d) Υ_2^5 (e) $\Upsilon_2^{5,\hat{\beta}}$ (f) $\Upsilon_2^{5,\beta}$

Fig. 25. Visualisation of the grasp predictions for a pitcher object with feature relations of different order and with different semantic. The colour depict the predicted likelihood for success. Green meaning a success likelihood of 1.0 and red meaning a success likelihood of 0.0.

7 Summary and conclusion

In this paper, we have introduced a method for finding combinations of visual features that are predictive for actions. The method has been exemplified for the problem of learning grasping actions. We have performed an analysis of the cross space of perceptual features and grasping actions with special focus on how an enrichment of the perception side leads to improvements of the derived prediction.

Through the performed investigations, we have been able to learn actions with a high likelihood of success for three different object classes, namely box like, round and open objects. For the box and round object set we were able to reach a grasp prediction success of up to 0.90 and 0.80 respectively, when utilising a second order feature constellation as a perceptual descriptor. This high success rate should be seen in the context that grasping of those objects is a rather simple task. For the more difficult open object set, we investigated in addition to granularity and order of feature combination also the impact of additional semantic information attached to the features through boundary information. From these results, we were able to achieve a success-rate of up to 0.75, when second order features with added semantic where utilised on the perception side.

By that we have replaced manual design of affordances as done in [2] by learning. We could confirm that relatively high success rates for action feature associations built by means of rather basic features is possible. Moreover and most importantly, we have shown how the structure of the feature space influences the results of the algorithm. For that we investigated three important dimensions of a feature space motivated by the visual hierarchy of the human visual system: granularity, order of features and semantic abstraction. Since our approach is not restricted to grasping, in future work we plan to apply our algorithm to other action affordances.

A Learning methodology experiments

In the following subsections, two aspects of the learning approach will be investigated and one aspect of the

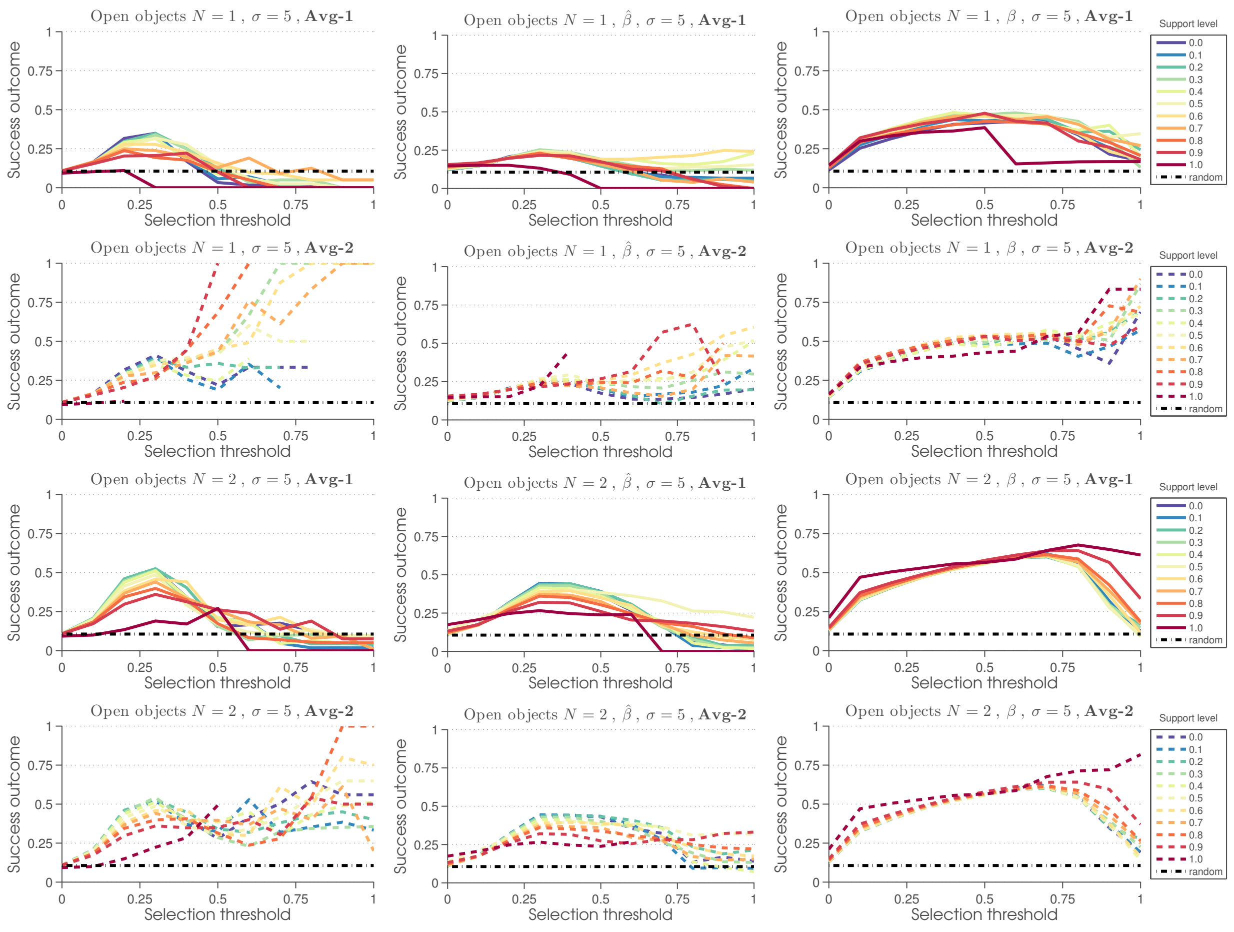


Fig. 26. Prediction results for the open object set, with a feature size of 5 and different support filters, see equations 17–22 for the used parametrisations, and see text for further details.

simulation scenario: (1) The prediction results when the direct action proposition approach (see section 4.3.1) is applied, (2) the difference between an automatically- and a manually set threshold (see section 4.2.1) and (3) a comparison between a free-floating environment and a table environment with gravity acting for grasp simulation (see section 5.2).

A.1 Direct action proposition approach

As a comparison to the voting scheme (see section 4.3.2), a number of experiments were performed using the direct action proposition method (see section 4.3.1). The experimental results are presented in table 1. Compared to the results presented when utilising the voting method (see section 6.1), these results are evaluated with a single measure depicting the success prediction. In the experiments, the order and granularity were varied for the box- and round object classes, whereas the level of semantic in addition were varied for the open object class. For the box- and round objects, two things can be observed: (1) A larger feature size improves the success rate for the first order cases, whereas it degrades for the second order cases, and (2) the success rate is in general higher for the second order cases. The improvement due to a larger feature is explained by the increased object knowledge. This information gain however seems to be counteracted by the addition of another feature, resulting in a degradation of prediction performance for the larger feature. For the open objects, three things can be observed. (1) The performance when utilising the representations without any semantic is very low, however an improvement is noticed when going from the first order cases to second order cases. (2) For the first order cases, a larger feature results in a better prediction rate. This is not the case for the second order cases, where the highest prediction rate is achieved at a feature size of 15. (3) The highest overall prediction rate is achieved at a representation based

FeatureSize	ObjectSet	Order + Abstraction					
σ		N=1	N=1, $\hat{\beta}$	N=1, β	N=2	N=2, $\hat{\beta}$	N=2, β
5	Box objects	0.51	-	-	0.65	-	-
	Round objects	0.57	-	-	0.66	-	-
	Open objects	0.05	0.10	0.44	0.12	0.14	0.45
15	Box objects	0.54	-	-	0.61	-	-
	Round objects	0.62	-	-	0.63	-	-
	Open objects	0.06	0.08	0.52	0.12	0.18	0.49
30	Box objects	0.54	-	-	0.60	-	-
	Round objects	0.67	-	-	0.56	-	-
	Open objects	0.08	0.08	0.53	0.11	0.13	0.45

Table 1. Prediction results when utilising the direct action proposition method. The used parametrizations are found in equations 17–22, and see text for further details.

on Υ_1^{30}. This essentially tells us, that the information gain from a larger feature is superior to adding another feature when used in connection with the direct action proposition method.

Finally, when comparing the results with the voting method, the direct action approach show lower performance, which is explained by the direct attachment of an action to a perceptual representation. This compares to the multiple particles, that are used to vote for a single action in the voting method.

A.2 Automatic vs. manual threshold

In this experiment we show the impact of an automatically chosen threshold as compared to a manually chosen threshold (see equations 14, 15 and 16 in section 4.2.1). In Fig. 27, the outcome of the experiments are shown for the three different object classes. We focus on the results with highest abstraction and order, meaning Υ_2^{β} for the open objects and Υ_2 for the box and round objects. For the open objects, we see an improved performance when the manual threshold is used. Both the top point of the curve and the consistency between the selection threshold and the prediction rate on the high end of the selection threshold show superior performance compared to an automatic thresholding. For the box- and round objects, the automatic threshold results show slightly better performance as the top point has a higher success rate, although the curve drops earlier than the manually selected threshold.

From these results, it can be derived, that an automatically chosen threshold shows a tendency to smooth the data more. Hence, the correspondence between the selection threshold and the actually success outcome is suboptimal close to the selection threshold of 1.0. However, although the manual chosen threshold shows better consistency between the selection threshold and the actual prediction, it comes with the cost of a lower top point and the need to manually define the threshold for the individual dimensions of the parametrisation.

A.3 Table vs. Free-floating scenario

In this experiment a comparison between a dynamic grasp simulation performed in free-floating and in a gravitation field is performed. By means of this comparison, we justify the usage of the "simpler" free-floating environment for the experiments performed in this work. To make this comparison, an experiment is performed on one set of objects, the open object set (see Fig. 13) and based on the results, we discuss why we believe the simpler scenario is preferable in this context. Initially the two different scenarios are explained.

The free-floating scenario, as explained in section 5.2, is a simulation performed with the object floating in space, enabling grasping from every direction without any gravity working in the simulation. However, the dynamic forces between the manipulator and object are still simulated. In contrast, the scenario with gravity, denoted the "table scenario", is limited by the fact that the object has to stand on a table perpendicular to the direction of gravity, hereby enabling grasping with gravity. In the free-floating environment there is no limitation on the ability to execute the grasps, however when presented with a table scenario a lot of potential grasps will initially be in collision with table and therefore do not make sense to execute. In the next section the setting for the experiments will be presented, then

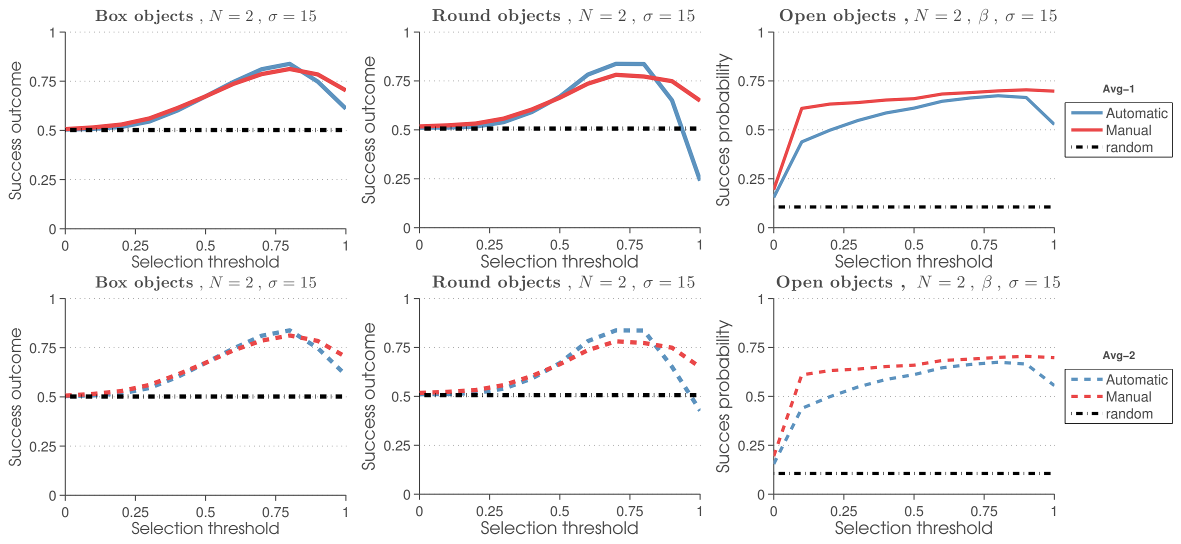


Fig. 27. Automatic vs. manual set threshold for the three different object set, see equations 17–22 for the used parametrisations, and see text for further details.

the results are presented and then finally the results will be summarised and interpreted.

A.3.1 Setting

The general setting for the experiments of the free-floating scenario is as explained in section 5. The setting for the table scenario is slightly different. All the open objects are placed on a table in a stable position with the opening away from the table. The gravity is set to $-9.82\frac{m}{s^2}$ along the normal of the table.

Given that the open objects primarily are graspable by the rim, this object setting allows for the objects still to be grasped. Secondly this could be seen as the "natural" pose of the objects. However when we then do a filtering based on gripper collision with the table, we remove a lot potential grasps from the equation, and due to the pose of the object this means that the probability to pick a successful grasps by chance is increased significantly. This can be seen when comparing the random line (see dotted black line) in the two result plots of Figs. 29 and 30. In Fig. 28 the distribution of grasps is visualised for the pitcher object with and without the filtering. As the intuition suggest, all the grasps at the bottom of the objects are filtered away as they collide with the table. Although not obvious from the visualisation, approximately half the grasps are filtered away for this object.

A.3.2 Results

In order to justify the use of the free-floating environment, we present results obtained in both environments, free-floating and table, and discuss the differences. Initially, we assess the results in Fig. 28. Qualitatively

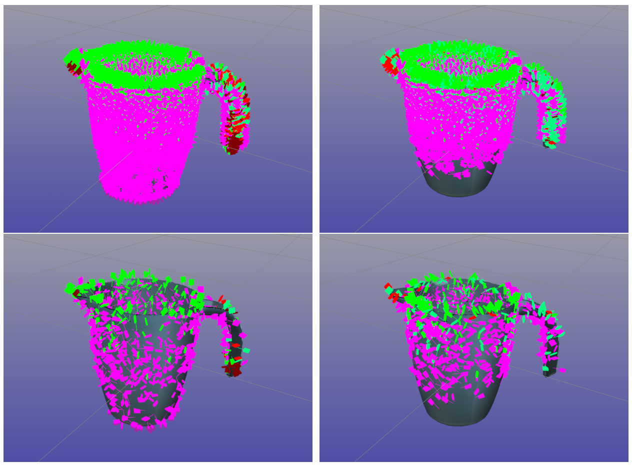


Fig. 28. Visualisation of the consequences of filtering due to collision in a table scenario. The pitcher object is used as example. The left column shows a visualisation of the grasp distribution (shown with small stick figures, the pink and red ones are failed grasps and the green ones are successful grasps) for the free-floating scenario and the right columns shows a visualisation of the grasp distribution after filtering for the table scenario. The top row shows the full grasping set and the bottom row shows a randomly chosen subset of 1000 grasps. In free-floating scenario 22,000 grasps, in table scenario — due to filtering — only 12,794 grasps.

the grasp outcome of the two scenarios are very similar. The distribution of successful grasps around the rim of the object seem similar, whereas there seem to be differences around the handle of the pitcher, suggesting that the addition of gravity makes it easier to grasp the handle. Otherwise the most obvious observation is the lack of grasps from the bottom of the object, due to collisions with the table. This result indicates that although there are some differences between the two scenarios due to the required filtering, the similarity between them is significant. The second results are presented in Figs. 29 and 30. These results show the prediction performance, similar to the core results presented in section 6.2, on the open objects for the two different scenarios. A number of conclusions can be derived from the results. First, the random chance of successfully picking a successful grasp is significantly increased for the table scenario (see dotted black line). This increase is caused by the filtering process and the fact that the successful grasps primarily are at the top rim of the objects, hence the vast majority of grasps filtered away are unsuccessful. Regarding the performance of the prediction, a certain pattern emerges. The parametrisation with, $N = 2$, β performs to a significant extent similarly when comparing the two scenarios. The other parametrisations show however a slightly different outcome. They all have increased performance in the table scenario, which is explained by the increased chance to be successful by random choice. Other than that, the most significant change is the increased performance in the table scenario for the parametrisation where boundary information is used. This can be in particular seen for the case $N = 1$, β, which shows similar performance as the $N = 2$, β case. The reason for this drastic improvement is that by filtering away the grasps at the bottom of the objects, a significant amount of boundary features found are not considered in learning and prediction, making a single boundary an even more significant feature in terms of successful grasps. See Fig. 16d and notice the boundary semantic features at the bottom of the object. The results for the percentage of objects that have been grasped show a similar pattern for the two different experiments. The percentages of grasped objects for the table scenario seem to drop a bit faster as compared to the free-floating scenario. A significant change is seen for N=1 β, where in the table scenario a higher coverage can be noted. This is explained with the same argument as before, namely that the table scenario filtering limits the perceptual space and therefore allows for better generalisation over the objects.

A.3.3 Summary

Based on the previously presented results the following points summarise the results: (1) Qualitatively a scenario with gravity exhibit similar performance as one in a free-floating environment, exemplified by the visualisation in Fig. 28. (2) In terms of the ability to predict grasp affordances with the most elaborated perceptual representation ($N = 2$, β), prediction performance exhibit comparable characteristics with success prediction rate around 0.70. (3) There is a significant improvement in the table scenario of the less elaborated feature representations, this however is explained by the bias that the filtering criteria needed for a table scenario produces, allowing for the simpler representation to expose this bias.

Given these results, we derive the following conclusion. From a strict grasping point of view, it would be preferable to perform the grasp simulations in an environment with gravity. However, in these experiments it has been demonstrated that the free-floating environment simulations deliver comparable performance, both exposed from a qualitative point of view and in the use for learning and predicting grasp affordances. What the experiments also exposed was the strong context specific bias, that a table scenario induces into the system, making the exploration of the perceptual spaces for grasp affordance learning and prediction performed in this paper diffuse. Based on these conclusion we find it justified to use the free-floating environment for our investigations of the perceptual action space.

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. 270273, Xperience.

References

[1] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008. [Online]. Available: http://ijr.sagepub.com/content/27/2/157.abstract

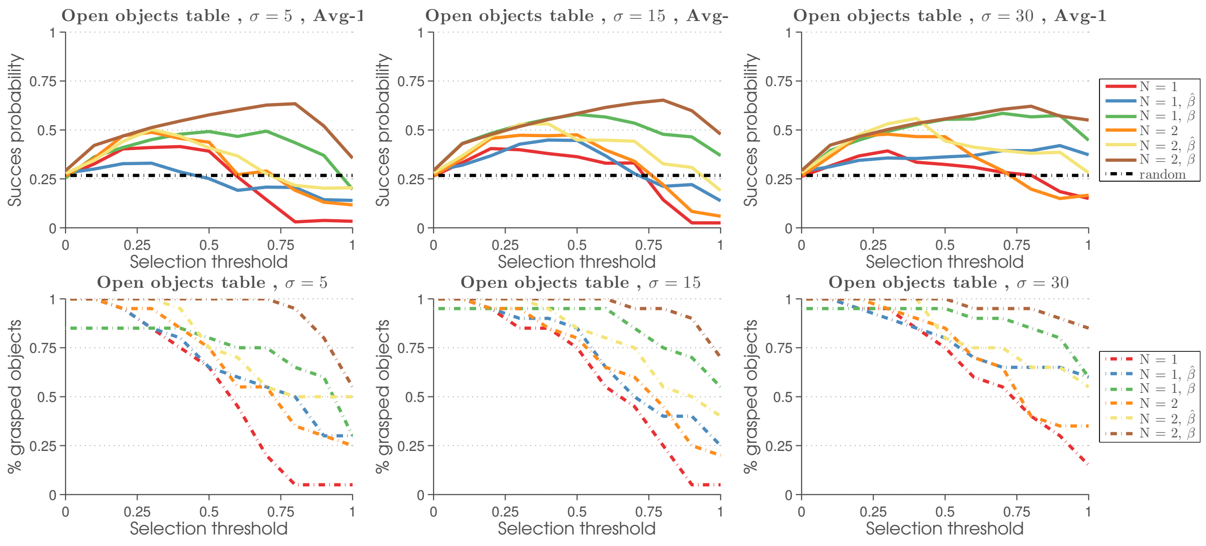


Fig. 29. Prediction result for open objects of granularity 5, 15 and 30 in a table scenario. See equations 17–22 for the used parametrisation, and see text for further details.

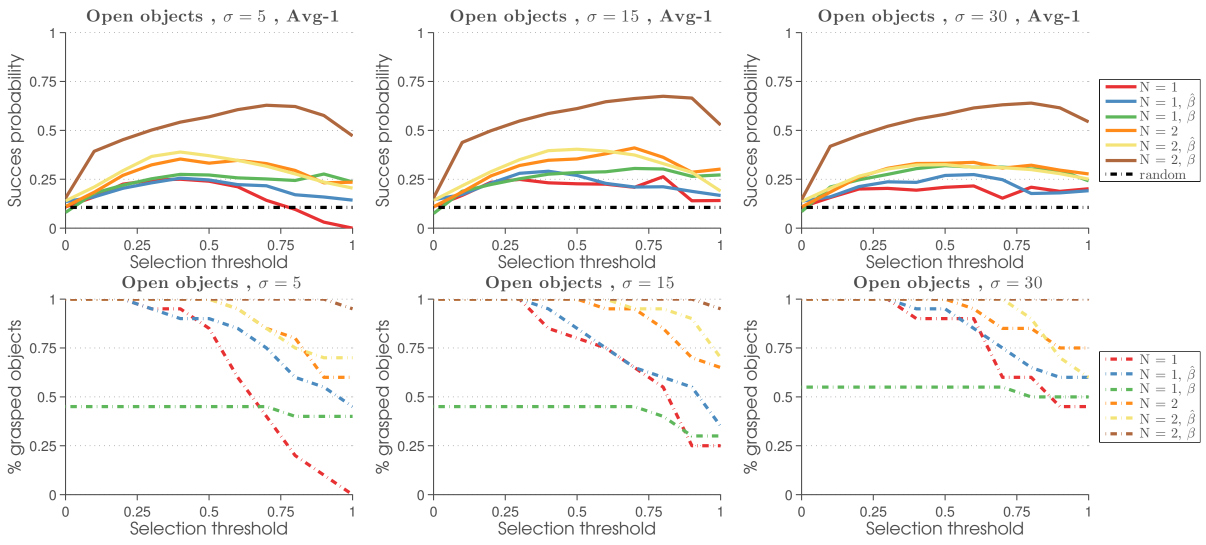


Fig. 30. Prediction result for open objects of granularity 5, 15 and 30 in a free-floating scenario. See equations 17–22 for the used parametrisation, and see text for further details.

[2] G. Kootstra, M. Popovic, J. Jørgensen, K. Kuklinski, K. Miatliuk, D. Kragic, and N. Kruger, "Enabling grasping of unknown objects through a synergistic use of edge and surface information," *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1190–1213, 2012. [Online]. Available: http://ijr.sagepub.com/content/31/10/1190.abstract

[3] A. Kleinhans, S. Thill, B. Rosman, R. Detry, and B. Tripp, "Modelling primate control of grasping for robotics applications," in *Second Workshop on Affordances: Visual Perception of Affordances and Functional Visual Primitives for Scene Analysis (in conjunction with ECCV 2014)*, 2014.

[4] G. Granlund, "The complexity of vision," *Signal Processing*, vol. 74, 1999.

[5] H. Zhou, H. S. Friedman, and R. von der Heydt, "Coding of border ownership in monkey visual cortex," *The Journal of Neuroscience*, vol. 20, no. 17, pp. 6594–6611, 2000. [Online]. Available: http://www.jneurosci.org/content/20/17/6594.abstract

[6] N. Krüger, P. Janssen, S. Kalkan, M. Lappe, A. Leonardis, J. H. Piater, A. J. Rodríguez-Sánchez, and L. Wiskott, "Deep hierarchies in the primate visual cortex: What can we learn for computer vision?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1847–1871, 2013.

[7] A. Treisman and G. Gelade, "A feature integration theory of attention," *Cognitive Psychology*, vol. 12, pp. 97–136, 1980.

[8] L. Montesano and M. Lopes, "Learning grasping affordances from local visual descriptors," in *Proceedings of the 2009 IEEE 8th International Conference on Development and Learning*, ser. DEVLRN '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–6. [Online]. Available: http://dx.doi.org/10.1109/DEVLRN.2009.5175529

[9] M. Stark, P. Lies, M. Zillich, J. Wyatt, and B. Schiele, "Functional object class detection based on learned affordance cues," in *Computer Vision Systems*, ser. Lecture Notes in Computer Science, A. Gasteratos, M. Vincze, and J. Tsotsos, Eds. Springer Berlin Heidelberg, 2008, vol. 5008, pp. 435–444. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-79547-6_42

[10] G. Fritz, L. Paletta, M. Kumar, G. Dorffner, R. Breithaupt, and E. Rome, "Visual learning of affordance based cues," in *Proceedings of the 9th International Conference on From Animals to Animats: Simulation of Adaptive Behavior*, ser. SAB'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 52–64. [Online]. Available: http://dx.doi.org/10.1007/11840541_5

[11] D. Rao, Q. V. Le, T. Phoka, M. Quigley, A. Sudsang, and A. Y. Ng, "Grasping novel objects with depth segmentation," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 2578–2585.

[12] M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. A. Şucan, "Towards reliable grasping and manipulation in household environments," in *Experimental Robotics*. Springer, 2014, pp. 241–252.

[13] J. Stückler, R. Steffens, D. Holz, and S. Behnke, "Efficient 3d object perception and grasp planning for mobile manipulation in domestic environments," *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1106–1115, 2013.

[14] M. Richtsfeld and M. Zillich, "Grasping unknown objects based on 21/2d range data," in *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*. IEEE, 2008, pp. 691–696.

[15] K. Huebner, S. Ruthotto, and D. Kragic, "Minimum volume bounding box decomposition for shape approximation in robot grasping," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1628–1633.

[16] N. Curtis, J. Xiao, and S. Member, "Efficient and effective grasping of novel objects through learning and adapting a knowledge base," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 2252–2257.

[17] R. Detry, C. H. Ek, M. Madry, and D. Kragic, "Learning a dictionary of prototypical grasp-predicting parts from grasping experience," in *IEEE International Conference on Robotics and Automation*, 2013.

[18] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp planning via decomposition trees," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 4679–4684.

[19] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, and S. Schaal, "Template-based learning of grasp selection," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2379–2384.

[20] J. Laaksonen, E. Nikandrova, and V. Kyrki, "Probabilistic sensor-based grasping," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 2019–2026.

[21] Y. Bekiroglu, R. Detry, and D. Kragic, "Learning tactile characterizations of object-and pose-specific grasps," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 1554–1560.

[22] A. Bierbaum and M. Rambow, "Grasp affordances from multi-fingered tactile exploration using dynamic potential fields," in *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, Dec 2009, pp. 168–174.

[23] T. A. J. Bohg, A. Morales and D. Kragic, "Data-driven grasp synthesis – a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[24] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *CoRR*, pp. –1–1, 2013.

[25] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from rgbd images: Learning using a new rectangle representation," in *ICRA'11*, 2011, pp. 3304–3311.

[26] S. Fidler, M. Boben, and A. Leonardis, "Learning hierarchical compositional representations of object structure," in *Object Categorization: Computer and Human Vision Perspectives*, S. Dickinson, A. Leonardis, B. Schiele, and M. Tarr, Eds. Cambridge University Press, 2009, pp. 196–215.

[27] D. W. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*, 1st ed. Wiley, Sept. 1992. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0471547700

[28] R. Becher, P. Steinhaus, R. Zöllner, and R. Dillmann, "Design and implementation of an interactive object modelling system," in *Proceedings Conference Robotik/ISR 2006, München*, May 2006.

[29] archvied3D, "Archive3d free online cad model database," http://www.archive3d.net.

[30] L.-P. Ellekilde and J. A. Jørgensen, "Robwork: A flexible toolbox for robotics research and education," *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pp. 1 –7, june 2010.

[31] J. A. Jørgensen, L.-P. Ellekilde, and H. G. Petersen, "RobWorkSim - an Open Simulator for Sensor based Grasping," *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pp. 1 –8, june 2010.

[32] M. Fischer and D. Henrich, "3d collision detection for industrial robots and unknown obstacles using multiple depth images," in *Advances in Robotics Research*, T. Kröger and F. Wahl, Eds. Springer Berlin Heidelberg, 2009, pp. 111–122. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01213-6_11

Comparing Binary Hamiltonian Monte Carlo and Gibbs Sampling for Training Discrete MRFs with Stochastic Approximation

Hanchen Xiong Sandor Szedmak Justus Piater
Institute of Computer Science, University of Innsbruck, Technikerstr.21a A-6020, Innsbruck, Austria

Learning discrete Markov random fields (MRFs) has been an important yet challenging machine learning task. In general, learning Markov random fields (MRF) is intractable due to the presence of partition function. Persistent contrastive divergence (PCD), known as a state-of-the-art learning algorithm for Markov random fields (MRFs), is a Robbins-Monro's stochastic approximation procedure (SAP) with Gibbs sampling as transitions (Salakhutdinov, 2010). We conduct an empirical study on a SAP with an alternative transition: binary Hamiltonian Monte Carlo. Ususally sampling-based method (*e.g.* Markov chain Monte Carlo) is employed for approximation. However, when the disibution exhibts multiple modes, the standard Metropplois algorithm will lead to low mxing rate of Markov chains. Hamiltonian Monte Carlo (HMC) is a Metropolis algorithm with a proposal distribution analogous to Hamiltonian dynamics. Compared to random walk in the standard Metropolis algorithm, HMC can propose a distant jump while still preserving a high acceptance rate. Suppose that we are interested in sampling from $p(\mathbf{x})$ (where $\mathbf{x} \in \mathbb{R}^D$). An auxiliary variable $\mathbf{q} \in \mathbb{R}^D$ with $\mathbf{q} \sim \mathcal{N}(\mathbf{q}; \mathbf{0}, \mathbf{M})$ is introduced (usually $\mathbf{M} = c \cdot \mathbf{I}_D$). A Hamiltonian function can be constructed as:

$$\mathcal{H}(\mathbf{x}, \mathbf{q}) = U(\mathbf{x}) + K(\mathbf{q}) \tag{1}$$

where $U(\mathbf{x})$, $K(\mathbf{q})$ are negative logarithms of $p(\mathbf{x})$ and $p(\mathbf{q})$. The changes of $\mathbf{x}$ and $\mathbf{q}$ over time ν are:

$$\dot{\mathbf{x}}(\nu) = \frac{\partial \mathcal{H}}{\partial \mathbf{q}(\nu)} = \mathbf{M}^{-1}\mathbf{q}(\nu) \quad \dot{\mathbf{q}}(\nu) = -\frac{\partial \mathcal{H}}{\partial \mathbf{x}(\nu)} = -\frac{dU(\mathbf{x})}{d\mathbf{x}(\nu)} \tag{2}$$

HMC can yield more effective sampling by making use of gradient information of target distribution's density function. We can also see, from (2), that HMC can only be applied on continuous distributions of which the partial derivatives of the log density function can be computed. Therefore, applying HMC to sample from discrete MRFs is not straightforward. However, the random variables are discrete in many applications (e.g. computer vision, natural language processing), Luckily, Zhang et al. (2012) pointed out that all discrete MRFs can be generally converted to Boltzmann machines (BMs):

$$p(\mathbf{x}; \boldsymbol{\theta}) = \frac{\exp\left(-E(\mathbf{x}; \boldsymbol{\theta})\right)}{\mathbf{Z}(\boldsymbol{\theta})}, \quad E(\mathbf{x}; \boldsymbol{\theta}) = -\sum_{i<j} x_i x_j W_{ij} \tag{3}$$

where $\mathbf{x} \in \{-1, +1\}^D$, $\boldsymbol{\theta} = \{W_{ij}\}$. In addition, Pakman and Paninski (2013) developed an extension of HMC for binary distribution to learn Bayesian regression with spike-and-slab prior. Here we attempt to apply this extended HMC, which we refer to as binary HMC (bHMC), for our purpose of learning Boltzmann machines (so also discrete MRFs). Assume that we are interested in sampling from a Boltzmann machine $p(\mathbf{x} \in \{-1, +1\}^D)$. An auxiliary, continuous variable $\mathbf{y} \in \mathbb{R}^D$ can be added with its conditional probability on $\mathbf{x}$ as a truncated Gaussian:

$$p(\mathbf{y}|\mathbf{x}) = \begin{cases} c \cdot \exp(-\frac{\mathbf{y}^\top \mathbf{y}}{2}) & \forall d \in [1, D], \text{sign}(y_d) = x_d \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Hence, instead of sampling $\mathbf{x}$ directly, we can sample $\mathbf{y}$ first and take their signs as our desired samples. By making use of orthant consistency constraint in (4), we can have:

$$p(\mathbf{y}) = \sum_{\mathbf{x}} p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}; \boldsymbol{\theta}) = p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}; \boldsymbol{\theta}) \tag{5}$$

Since $\mathbf{y}$ is continuous, we can employ HMC to sample them from $p(\mathbf{y})$. By substituting (5) into (1) and (2), we can have:

$$y_d(\nu) = u_d \sin(\omega_d + \nu) \quad q_d(\nu) = u_d \cos(\omega_d + \nu) \tag{6}$$

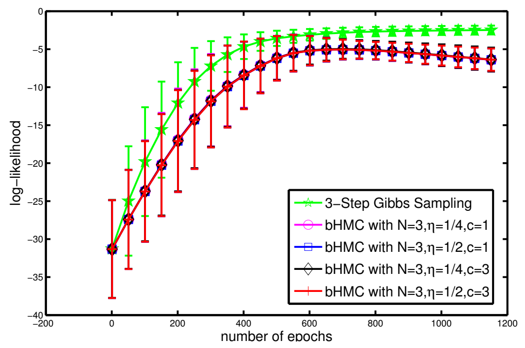
where $u_d = \sqrt{y_d(0)^2 + q_d(0)^2}$, and $\omega_d = \tan^{-1}\left(\frac{y_d(0)}{q_d(0)}\right)$. In addition, since (6) keeps Hamiltonian function (1) invariant, change of $\mathbf{y}, \mathbf{q}$ according to (6) are always accepted. It can be seen in (6) that (q_d, y_d) actually moves counterclockwisely along a circle with radius u_d. However, one issue arising from discontinuity of $p(\mathbf{y}|\mathbf{x})$ is that when y_d hits 0 at time ν_*, whether it will be reflected from the $y_d = 0$ or cross it depends on the sign of:

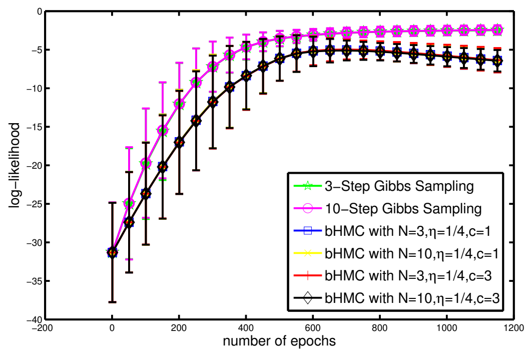
$$\frac{q_d^2(\nu_*^-)}{2} - \left(E(-x_d, \mathbf{x}_{\neg d}; \boldsymbol{\theta}) - E(x_d, \mathbf{x}_{\neg d}; \boldsymbol{\theta})\right) \tag{7}$$

where $q_d(\nu_*^-)$ is the q_d immediately before time ν_*, so it equlals to u_d. (7) can be considered as a pseudo Gibbs sampling. When $E(-x_d, \mathbf{x}_{\neg d}; \boldsymbol{\theta}) - E(x_d, \mathbf{x}_{\neg d}; \boldsymbol{\theta}) > 0$, the probability of switching sign of x_d is lower than not. According to (7), as long as the energy raise is smaller than a threshold $u_d^2/2$, the switch still can take place. In addition, with different initializations of $(y_d(0), q_d(0))$, different y_d will hit 0 at different time ν_*^d, so bHMC is a randomly scheduled sampling. Finally, since $y_d(N\pi) = y(0)$ if y_d always gets reflected, traveling time is recommended as $T = (N + \eta)\pi$ so as to avoid degeneracy of samples ($\eta \in (0, 1)$). In conclusion, bHMC somehow resembles Gibbs sampling but with a different acceptance criterion. To verify its practical applicabilities, we compared it against Gibbs sampling in SAP (with different c, N, η) on training a toy Boltzmann machine ($D = 10$), the results are presented as follows. Our empirical results suggest that the SAP with bHMC is inferior to the one with Gibbs sampling for learning discrete MRFs.

Acknowledgment

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

Ari Pakman and Liam Paninski. Auxiliary-variable exact hamiltonian monte carlo sampler for binary distributions. In *NIPS*, 2013.

Ruslan Salakhutdinov. Learning in markov random fields using tempered transitions. In *NIPS*, 2010.

Yichuan Zhang, Charles Sutton, Amos Storkey, and Zoubin Ghahramani. Continuous relaxations for discrete hamiltonian monte carlo. In *NIPS*. 2012.

JMLR: Workshop and Conference Proceedings 29:1–16, 2014 ACML 2014

Towards Maximum Likelihood: Learning Undirected Graphical Models using Persistent Sequential Monte Carlo

Hanchen Xiong HANCHEN.XIONG@UIBK.AC.AT
Sandor Szedmak SANDOR.SZEDMAK@UIBK.AC.AT
Justus Piater JUSTUS.PIATER@UIBK.AC.AT
Institute of Computer Science, University of Innsbruck
Technikerstr. 21a, A-6020 Innsbruck, Austria

Abstract

Along with the emergence of algorithms such as persistent contrastive divergence (PCD), tempered transition and parallel tempering, the past decade has witnessed a revival of learning undirected graphical models (UGMs) with sampling-based approximations. In this paper, based upon the analogy between Robbins-Monro's stochastic approximation procedure and sequential Monte Carlo (SMC), we analyze the strengths and limitations of state-of-the-art learning algorithms from an SMC point of view. Moreover, we apply the rationale further in sampling at each iteration, and propose to learn UGMs using *persistent sequential Monte Carlo* (PSMC). The whole learning procedure is based on the samples from a long, persistent sequence of distributions which are actively constructed. Compared to the above-mentioned algorithms, one critical strength of PSMC-based learning is that it can explore the sampling space more effectively. In particular, it is robust when learning rates are large or model distributions are high-dimensional and thus multi-modal, which often causes other algorithms to deteriorate. We tested PSMC learning, also with other related methods, on carefully-designed experiments with both synthetic and real-world data, and our empirical results demonstrate that PSMC compares favorably with the state of the art.

Keywords: Sequential Monte Carlo, maximum likelihood learning, undirected graphical models.

1. Introduction

Learning undirected graphical models (UGMs), or Markov random fields (MRF), has been an important yet challenging machine learning task. On the one hand, thanks to its flexible and powerful capability in modeling complicated dependencies, UGMs are prevalently used in many domains such as computer vision, natural language processing and social analysis. Undoubtedly, it is of great significance to enable UGMs' parameters to be automatically adjusted to fit empiric data, *e.g.* maximum likelihood (ML) learning. A fortunate property of the likelihood function is that it is concave with respect to its parameters (Koller and Friedman, 2009), and therefore gradient ascent can be applied to find the unique maximum. On the other hand, learning UGMs via ML in general remains intractable due to the presence of the partition function. Monte Carlo estimation is a principal solution to the problem. For example, one can employ Markov chain Monte Carlo (MCMC) to obtain samples from the model distribution, and approximate the partition function with the samples. However, the sampling procedure of MCMC is very inefficient because it usually requires a large number of steps for the Markov chain to reach equilibrium. Even though in some cases where efficiency can be ignored, another weakness of MCMC estimation is that it yields large estimation variances. A more practically feasible alternative is MCMC maximum likelihood (MCMCML;

Geyer 1991); see section 2.1. MCMCML approximates the gradient of the partition function with importance sampling, in which a proposal distribution is initialized to generate a fixed set of MCMC samples. Although MCMCML increases efficiency by avoiding MCMC sampling at every iteration, it also suffers from high variances (with different initial proposal distributions). Hinton (2002) studied *contrastive divergence* (CD) to replace the objective function of ML learning. This turned out to be an efficient approximation of the likelihood gradient by running only a few steps of Gibbs sampling, which greatly reduces variance as well as the computational burden. However, it was pointed out that CD is a biased estimation of ML (Carreira-Perpinan and Hinton, 2005), which prevents it from being widely employed (Tieleman, 2008; Tieleman and Hinton, 2009; Desjardins et al., 2010). Later, a *persistent* version of CD (PCD) was put forward as a closer approximation of the likelihood gradient (Tieleman, 2008). Instead of running a few steps of Gibbs sampling from training data in CD, PCD maintains an almost persistent Markov chain throughout iterations by preserving samples from the previous iteration, and using them as the initializations of Gibbs samplers in the current iteration. When the learning rate is sufficiently small, samples can be roughly considered as being generated from the stationary state of the Markov chain. However, one critical drawback in PCD is that Gibbs sampling will generate highly correlated samples between consecutive weight updates, so mixing will be poor before the model distribution gets updated at each iteration. The limitations of PCD sparked many recent studies of more sophisticated sampling strategies for effective exploration within data space (section 3). For instance, Salakhutdinov (2010) studied *tempered transition* (Neal, 1994) for learning UGMs. The strength of tempered transition is that it can make potentially big transitions by going through a trajectory of intermediary Gibbs samplers which are smoothed with different temperatures. At the same time, *parallel tempering*, which can be considered a parallel version of tempered transition, was developed by Desjardins et al. (2010) for training restricted Boltzmann machines (RBMs). Contrary to a single Markov chain in PCD and tempered transition, parallel tempering maintains a pool of Markov chains governed by different temperatures. Multiple tempered chains progress in parallel and are mixed at each iteration by randomly swapping the states of neighbouring chains.

The contributions of this paper are twofold. The first is theoretic. By linking Robbins-Monro's stochastic approximation procedure (SAP; Robbins and Monro 1951) and sequential Monte Carlo (SMC), we cast PCD and other state-of-the-art learning algorithms into a SMC-based interpretation framework. Moreover, within the SMC-based interpretation, two key factors which affect the performance of learning algorithms are disclosed: *learning rate* and *model complexity* (section 4). Based on this rationale, the strengths and limitations of different learning algorithms can be analyzed and understood in a new light. The second contribution is practical. Inspired by the understanding of learning UGMs from a SMC perspective, and the successes of global tempering used in parallel tempering and tempered transition, we put forward a novel approximation-based algorithm, *persistent SMC* (PSMC), to approach the ML solution in learning UGMs. The basic idea is to construct a long, persistent distribution sequence by inserting many tempered intermediary distributions between two successively updated distributions (section 5). According to our empirical results on learning two discrete UGMs (section 6), the proposed PSMC outperforms other learning algorithms in challenging circumstances, *i.e.* large learning rates or large-scale models.

2. Learning Undirected Graphical Models

In general, we can define undirected graphical models (UGMs) in an energy-based form:

$$\begin{aligned} p(\mathbf{x};\boldsymbol{\theta}) &= \frac{\exp\left(-E(\mathbf{x};\boldsymbol{\theta})\right)}{\mathbf{Z}(\boldsymbol{\theta})} && (1)\\ \textit{Energy function:}\quad E(\mathbf{x};\boldsymbol{\theta}) &= -\boldsymbol{\theta}^\top \phi(\mathbf{x}) && (2)\end{aligned}$$

with random variables $\mathbf{x} = [x_1, x_2, \ldots, x_D] \in \mathcal{X}^D$ where x_d can take N_d discrete values, $\phi(\mathbf{x})$ is a K-dimensional vector of sufficient statistics, and parameter $\boldsymbol{\theta} \in \mathbb{R}^K$. $\mathbf{Z}(\boldsymbol{\theta}) = \sum_{\mathbf{x}} \exp(\boldsymbol{\theta}^\top \phi(\mathbf{x}))$ is the partition function for global normalization. Learning UGMs is usually done via maximum likelihood (ML). A critical observation of UGMs' likelihood functions is that they are concave with respect to $\boldsymbol{\theta}$, therefore any local maximum is also global maximum (Koller and Friedman, 2009), and gradient ascent can be employed to find the optimal $\boldsymbol{\theta}^*$. Given training data $\mathcal{D} = \{\mathbf{x}^{(m)}\}_{m=1}^M$, we can compute the derivative of average log-likelihood $\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{M}\sum_{m=1}^M \log p(\mathbf{x}^{(m)};\boldsymbol{\theta})$ as

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}} = \underbrace{\mathbb{E}_{\mathcal{D}}(\phi(\mathbf{x}))}_{\psi^+} - \underbrace{\mathbb{E}_{\boldsymbol{\theta}}(\phi(\mathbf{x}))}_{\psi^-}, \tag{3}$$

where $\mathbb{E}_{\mathcal{D}}(\xi)$ is the expectation of ξ under the empirical data distribution $p_{\mathcal{D}} = \frac{1}{M}\sum_{m=1}^M \delta(\mathbf{x}^{(m)})$, while $\mathbb{E}_{\boldsymbol{\theta}}(\xi)$ is the expectation of ξ under the model probability with parameter $\boldsymbol{\theta}$. The first term in (3), which is often referred to as *positive phase* ψ^+, can be easily computed as the average of $\phi(\mathbf{x}^{(m)}), \mathbf{x}^{(m)} \in \mathcal{D}$. The second term in (3), also known as *negative phase* ψ^-, however, is not trivial because it is a sum of $\prod_{d=1}^D N_d$ terms, which is only computationally feasible for UGMs of very small size. Markov chain Monte Carlo (MCMC) can be employed to approximate ψ^-, although it is usually expensive and leads to large estimation variances. The underlying procedure of ML learning with gradient ascent, according to (3), can be envisioned as a behavior that iteratively pulls down the energy of the data space occupied by $\mathcal{D}$ (positive phase), but raises the energy over all data space $\mathcal{X}^D$ (negative phase), until it reaches a balance ($\psi^+ = \psi^-$).

2.1. Markov Chain Monte Carlo Maximum Likelihood

A practically feasible approximation of (3) is Markov chain Monte Carlo maximum likelihood (MCMCML; Geyer 1991). In MCMCML, a proposal distribution $p(\mathbf{x};\boldsymbol{\theta}_0)$ is set up in the same form as (1) and (2), and we have

$$\begin{aligned} \frac{\mathbf{Z}(\boldsymbol{\theta})}{\mathbf{Z}(\boldsymbol{\theta}_0)} &= \frac{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}^\top \phi(\mathbf{x}))}{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}_0^\top \phi(\mathbf{x}))} && (4)\\ &= \frac{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}^\top \phi(\mathbf{x}))}{\exp(\boldsymbol{\theta}_0^\top \phi(\mathbf{x}))} \times \frac{\exp(\boldsymbol{\theta}_0^\top \phi(\mathbf{x}))}{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}_0^\top \phi(\mathbf{x}))} && (5)\\ &= \sum_{\mathbf{x}} \exp\left((\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \phi(\mathbf{x})\right) p(\mathbf{x};\boldsymbol{\theta}_0) && (6)\\ &\approx \frac{1}{S}\sum_{s=1}^S w^{(s)} && (7)\end{aligned}$$

Algorithm 1 MCMCML Learning Algorithm

```
Input: training data $\mathcal{D} = \{\mathbf{x}^{(m)}\}_{m=1}^{M}$; learning rate $\eta$; gap $L$ between two successive proposal
    distribution resets
 1: $t \leftarrow 0$, initialize the proposal distribution $p(\mathbf{x}; \boldsymbol{\theta}_0)$
 2: while ! stop criterion do
 3:     if $(t \mod L) == 0$ then
 4:         (Re)set the proposal distribution as $p(\mathbf{x}; \boldsymbol{\theta}_t)$
 5:         Sample $\{\bar{\mathbf{x}}^{(s)}\}$ from $p(\mathbf{x}; \boldsymbol{\theta}_t)$
 6:     end if
 7:     Calculate $w^{(s)}$ using (8)
 8:     Calculate gradient $\frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}}$ using (9)
 9:     update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}}$
10:     $t \leftarrow t + 1$
11: end while
Output: estimated parameters $\boldsymbol{\theta}^* = \boldsymbol{\theta}_t$
```

where $w^{(s)}$ is

$$w^{(s)} = \exp\left((\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \phi(\bar{\mathbf{x}}^{(s)}) \right), \tag{8}$$

and the $\bar{\mathbf{x}}^{(s)}$ are sampled from the proposal distribution $p(\mathbf{x}; \boldsymbol{\theta}_0)$. By substituting $\mathbf{Z}(\boldsymbol{\theta}) = \mathbf{Z}(\boldsymbol{\theta}_0)\frac{1}{S}\sum_{s=1}^{S} w^{(s)}$ into (1) and average log-likelihood, we can compute corresponding gradient as (note $\mathbf{Z}(\boldsymbol{\theta}_0)$ will be eliminated since it corresponds to a constant in the logarithm)

$$\frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}} = \mathbb{E}_{\mathcal{D}}(\phi(\mathbf{x})) - \mathbb{E}_{\boldsymbol{\theta}_0}(\phi(\mathbf{x})), \tag{9}$$

where $\mathbb{E}_{\boldsymbol{\theta}_0}(\xi)$ is the expectation of ξ under a *weighted* empirical data distribution $p_{\boldsymbol{\theta}_0} = \sum_{s=1}^{S} w^{(s)} \delta(\bar{\mathbf{x}}^{(s)}) / \sum_{s=1}^{S} w^{(s)}$ with data sampled from $p(\mathbf{x}; \boldsymbol{\theta}_0)$. From (9), it can be seen that MCMCML does nothing more than an importance sampling estimation of ψ^- in (3). MCMCML has the nice asymptotic convergence property (Salakhutdinov, 2010) that it will converge to the exact ML solution when the number of samples S goes to infinity. However, as an inherent weakness of importance sampling, the performance of MCMCML in practice highly depends on the choice of the proposal distribution, which results in large estimation variances. The phenomenon gets worse when it scales up to high-dimensional models. One engineering trick to alleviate this pain is to reset the proposal distribution, after a certain number of iterations, to the recently updated estimation $p(\mathbf{x}; \boldsymbol{\theta}^{estim})$ (Handcock et al., 2007). Pseudocode of the MCMCML learning algorithm is presented in Algorithm 1.

3. State-of-the-art Learning Algorithms

Contrastive Divergence (CD) is an alternative objective function of likelihood (Hinton, 2002), and turned out to be de facto a cheap and low-variance approximation of the maximum likelihood (ML) solution. CD tries to minimize the discrepancy between two Kullback-Leibler (KL) divergences, $KL(p^0|p^\infty_{\boldsymbol{\theta}})$ and $KL(p^n_{\boldsymbol{\theta}}|p^\infty_{\boldsymbol{\theta}})$, where $p^0 = p(\mathcal{D}; \boldsymbol{\theta})$, $p^n_{\boldsymbol{\theta}} = p(\bar{\mathcal{D}}_n; \boldsymbol{\theta})$ with $\bar{\mathcal{D}}_n$ denoting the data sampled after n steps of Gibbs sampling with parameter $\boldsymbol{\theta}$, and $p^\infty_{\boldsymbol{\theta}} = p(\bar{\mathcal{D}}_\infty; \boldsymbol{\theta})$ with $\bar{\mathcal{D}}_\infty$ denoting the data sampled from the equilibrium of a Markov chain. Usually $n = 1$ is used, and correspondingly it is referred to as the CD-1 algorithm. The negative gradient of CD-1 is

$$-\frac{\partial\big(CD_1(\mathcal{D};\boldsymbol{\theta})\big)}{\partial\boldsymbol{\theta}} = \mathbb{E}_{\mathcal{D}}(\phi(\mathbf{x})) - \mathbb{E}_{\bar{\mathcal{D}}_1}(\phi(\mathbf{x})) \tag{10}$$

where $\mathbb{E}_{\bar{\mathcal{D}}_1}(\xi)$ is the expectation of ξ under the distribution $p^1_{\boldsymbol{\theta}}$. The key advantage of CD-1 is that it efficiently approximates ψ^- in the likelihood gradient (3) by running only one step Gibbs sampling. While this local exploration of sampling space can avoid large variances, CD-1 was theoretically (Carreira-Perpinan and Hinton, 2005) and empirically (Tieleman, 2008; Tieleman and Hinton, 2009; Desjardins et al., 2010) proved to be a biased estimation of ML .

Persistent Contrastive Divergence (PCD) is an extension of CD by running a nearly persistent Markov chain. For approximating ψ^- in likelihood gradient (3), the samples at each iteration are retained as the initialization of Gibbs sampling in the next iteration. The mechanism of PCD was usually interpreted as a case of Robbins-Monro's stochastic approximation procedure (SAP; Robbins and Monro 1951) with Gibbs sampling as transitions. In general SAP, if the learning rate η is sufficiently small compared to the mixing rate of the Markov chain, the chain can be roughly considered as staying close to the equilibrium distribution (i.e. PCD$\rightarrow$ML when $\eta \rightarrow 0$). Nevertheless, Gibbs sampling as used in PCD heavily hinders the exploration of data space by generating highly correlated samples along successive model updates. This hindrance becomes more severe when the model distribution is highly multi-modal. Although multiple chains (mini-batch learning) used in PCD can mitigate the problem, we cannot generally expect the number of chains to exceed the number of modes. Therefore, at the late stage of learning, PCD usually gets stuck in a local optimum, and in practice, small and linearly-decayed learning rates can improve the performance (Tieleman, 2008).

Tempered Transition was originally developed by Neal (1994) to generate relatively big jumps in Markov chains while keeping reasonably high acceptance rates. Instead of standard Gibbs sampling used in PCD, tempered transition constructs a sequence of Gibbs samplers based on the model distribution specified with different temperatures:

$$p_h(\mathbf{x};\boldsymbol{\theta}) = \frac{\exp(-E(\mathbf{x};\boldsymbol{\theta})\beta_h)}{\mathbf{Z}(h)} \tag{11}$$

where h indexes temperatures $h \in [0, H]$ and β_H are inverse temperatures $0 \leq \beta_H < \beta_{H-1} < \cdots \beta_0 = 1$. In particular, β_0 corresponds to the original complex distribution. When h increases, the distribution gets more flat, where Gibbs samplers can more adequately explore. In tempered transition, a sample is generated with a Gibbs sampler starting from the original distribution. It then goes through a trajectory of Gibbs sampling through sequentially tempered distributions (11). A backward trajectory is then run until the sample reaches the original distribution. The acceptance of the final sample is determined by the probability of the whole forward-and-backward trajectory. If the trajectory is rejected, the sample does not move at all, which is even worse than local movements of Gibbs sampling, so β_H is set relatively high (0.9 in Salakhutdinov 2010) to ensure high acceptance rates.

Parallel Tempering, on the other hand, is a "parallel" version of Tempered Transition, in which smoothed distributions (11) are run with one step of Gibbs sampling in parallel at each iteration. Thus, samples native to more uniform chains will move with larger transitions, while samples native to the original distribution still move locally. All chains are mixed by swapping samples of randomly selected neighbouring chains. The probability of the swap is

$$r = \exp\big((\beta_h - \beta_{h+1})(E(\mathbf{x}_h) - E(\mathbf{x}_{h+1}))\big) \tag{12}$$

Although multiple Markov chains are maintained, only samples at the original distribution are used. In the worst case (there is no swap between β_0 and β_1), parallel tempering degrades to PCD-1. β_H can be set arbitrarily low (0 was used by Desjardins et al. 2010).

4. Learning as Sequential Monte Carlo

Before we delve into the analysis of the underlying mechanism in different learning algorithms, it is better to find a unified interpretation framework, within which the behaviors of all algorithms can be more apparently viewed and compared in a consistent way. In most previous work, PCD, tempered transition and parallel tempering were studied as special cases of Robbins-Monro's stochastic approximation procedure (SAP; Tieleman and Hinton 2009; Desjardins et al. 2010; Salakhutdinov 2010). These studies focus on the interactions between the mixing of Markov chains and distribution updates. However, we found that, since the model changes at each iteration, the Markov chain is actually not subject to an invariant distribution, the concept of the mixing of Markov chains is fairly subtle and difficult to capture based on SAP.

Alternatively, Asuncion et al. (2010) exposed that PCD can be interpreted as a sequential Monte Carlo procedure by extending MCMCML to a particle filtered version. To have an quick overview of sequential Monte Carlo More and how it is related to learning UGMs, we first go back to Markov chain Monte Carlo maximum likelihood (MCMCML; section 2.1) and examine it in an extreme case. When the proposal distribution in MCMCML is reset at every iteration as the previously updated estimation, *i.e.* $L = 1$ in Algorithm 1 and the proposal distribution is left as $p(\mathbf{x};\boldsymbol{\theta}_{t-1})$ at the tth iteration, the weights will be computed as $w^{(s)} = \exp(\boldsymbol{\theta_t} - \boldsymbol{\theta_{t-1}})^\top \phi(\bar{\mathbf{x}}^{(s)})$. Since the parameters $\boldsymbol{\theta}$ do not change very much along iterations, it is not necessary to generate particles[1] from proposal distributions at each iteration. Instead, a set of particles are initially generated and reweighted sequentially for approximating the negative phase. However, if the gap between two successive $\boldsymbol{\theta}$ is relatively large, particles will degenerate. Usually, the effective sampling size (ESS) can be computed to measure the degeneracy of particles, so if ESS is smaller than a pre-defined threshold, resampling and MCMC transition are necessary to recover from it. The description above notably leads to *particle filtered MCMCML* (Asuncion et al., 2010), which greatly outperforms MCMCML with small amount of extra computation.

More interestingly, it was pointed out that PCD also fits the above sequential Monte Carlo procedure (*i.e.* importance reweighting + resampling + MCMC transition) with uniform weighting for all particles and Gibbs sampling as MCMC transition. Here we extend this analogy further to general Robbins-Monro's SAP, into which tempered transition and parallel tempering are also categorized, and write out a uniform interpretation framework of all learning algorithms from SMC perspective (see Algorithm 2). Note that all particle weights are uniformly assigned; resampling has no effect and can be ignored. In addition, the MCMC transition step is forced to take place at every iteration, believing that the particle set is always degenerated.

It is also worth noting that when we are applying algorithms in Algorithm 2, we are not interested in particles from any individual target distribution (which is usually the purpose of SMC).

1. From now on, we use "particles" to fit SMC terminology, it is equivalent to "samples" unless mentioned otherwise.

Algorithm 2 Interpreting Learning as SMC

```
Input:   training data $\mathcal{D} = \{\mathbf{x}^{(m)}\}_{m=1}^{M}$; learning rate $\eta$
 1: Initialize $p(\mathbf{x};\boldsymbol{\theta}_0), t \leftarrow 0$
 2: Sample particles $\{\bar{\mathbf{x}}_0^{(s)}\}_{s=1}^{S} \sim p(\mathbf{x};\boldsymbol{\theta}_0)$
 3: while ! stop criterion do
 4:   // importance reweighting
      Assign $w^{(s)} \leftarrow \frac{1}{S}, \forall s \in S$
 5:   // resampling is ignored because it has no effect
 6:   // MCMC transition
 7:   switch (algorithmic choice)
 8:   case CD:
 9:     generate a brand new particle set $\{\bar{\mathbf{x}}_{t+1}^{(s)}\}_{s=1}^{S}$ with one step Gibbs sampling from $\mathcal{D}$
10:   case PCD:
11:     evolve particle set $\{\bar{\mathbf{x}}_t^{(s)}\}_{s=1}^{S}$ to $\{\bar{\mathbf{x}}_{t+1}^{(s)}\}_{s=1}^{S}$ with one step Gibbs sampling
12:   case Tempered Transition:
13:     evolve particle set $\{\bar{\mathbf{x}}_t^{(s)}\}_{s=1}^{S}$ to $\{\bar{\mathbf{x}}_{t+1}^{(s)}\}_{s=1}^{S}$ with tempered transition
14:   case Parallel Tempering:
15:     evolve particle set $\{\bar{\mathbf{x}}_t^{(s)}\}_{s=1}^{S}$ to $\{\bar{\mathbf{x}}_{t+1}^{(s)}\}_{s=1}^{S}$ with parallel tempering
16:   end switch
17:   // update distribution
      Compute the gradient $\Delta\boldsymbol{\theta}_t$ according to (3)
18:   $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta\Delta\boldsymbol{\theta}_t$
19:   $t \leftarrow t + 1$
20: end while
Output:   estimated parameters $\boldsymbol{\theta}^* = \boldsymbol{\theta}_t$
```

Instead, we want to obtain particles faithfully sampled from all sequence distributions. It can be easily imagined that one badly sampled particle set at tth iteration will lead to a biased incremental update $\Delta\boldsymbol{\theta}_t$. Consequently, the learning will go to a wrong direction even though the later sampling is perfectly good. In other words, we are considering all sequentially updated distributions $p(\mathbf{x};\boldsymbol{\theta}_t)$ as our target distributions.

In practice, the performance of SMC highly depends on the construction of sequential distributions. In our learning case, sequential distributions are learned by iterative updates, therefore, learning and sampling are somehow entangled. As we mentioned earlier, particles will degenerate when the gap between successive sequential distributions is large. Checking ESS followed by resampling and MCMC transition can help to some extent. However, in many practical cases where real-world distributions are extremely complex, more consideration on MCMC transition is due. In our case of learning UGMs, the gap can be intuitively understood as the product of *learning rate* η and *model complexity* $\mathcal{O}(\boldsymbol{\theta})$. Therefore, we believe that learning rate and model complexity[2] are two key factors to challenge learning algorithms.

2. Here we consider the multimodality of a distribution as its complexity, *i.e.* smooth distributions are less complex than multi-modal distributions.

Within this SMC-based interpretation, we can see that four algorithms differ from each other at MCMC transitions, which is an important component in SMC (Schäfer and Chopin, 2013). In PCD, a one-step Gibbs sampler is used as MCMC transition. As for tempered transition, a Metropolis-Hasting (MH) move based on forward-and-backward sequence of Gibbs samplers of different temperatures is employed. Likewise, parallel tempering also uses a MH move. This move is generated by swapping particles native to the distributions of different temperatures. By contrast, in CD, a brand new particle set is generated by running one-step Gibbs sampling from training data, which is actually not a MCMC transition. When the learning rate is small and two successive distributions are smooth (*e.g.* at the early stage of learning or when the model is of low dimension), PCD, tempered transition and parallel tempering can traverse sampling space sufficiently well. However, when the learning rate is large or two sequential distributions exhibit multiple modes (*e.g.* at the late stage of learning or when the model is high-dimensional), highly correlated particles from the one-step Gibbs sampler's local movement cannot go through the gap between two distributions. Tempered transition and parallel tempering, instead, are more robust to the large gap since it moves closer to the later distribution by making use of many globally-tempered intermediary distributions. The worst case is CD, which always samples particles within the vicinity of training data $\mathcal{D}$. So it will eventually drop $\mathcal{D}$ down into an energy well surrounded by barriers set up by their proximities.

5. Persistent Sequential Monte Carlo

It was explained that learning UGMs can be interpreted as a SMC procedure. Here we propose to apply this rationale further in learning UGMs with a deeper construction of sequential distributions. The basic idea is very simple; given particles from $p(\mathbf{x};\boldsymbol{\theta}_t)$, many sub-sequential distributions are inserted to construct a sub-SMC for obtaining particles from $p(\mathbf{x};\boldsymbol{\theta}_{t+1})$. Inspired by global tempering used in parallel tempering and tempered transition, we build sub-sequential distributions $\{p_h(\mathbf{x};\boldsymbol{\theta}_{t+1})\}_{h=0}^{H}$ between $p(\mathbf{x};\boldsymbol{\theta}_t)$ and $p(\mathbf{x};\boldsymbol{\theta}_{t+1})$ as follows:

$$p_h(\mathbf{x};\boldsymbol{\theta}_{t+1}) \propto p(\mathbf{x};\boldsymbol{\theta}_t)^{1-\beta_h} p(\mathbf{x};\boldsymbol{\theta}_{t+1})^{\beta_h} \tag{13}$$

where $0 \leq \beta_H \leq \beta_{H-1} \leq \cdots \beta_0 = 1$. In this way, the length of the distribution sequence will be extended in SMC. In addition, obviously, $p_H(\mathbf{x};\boldsymbol{\theta}_{t+1}) = p(\mathbf{x};\boldsymbol{\theta}_t)$ while $p_0(\mathbf{x};\boldsymbol{\theta}_{t+1}) = p(\mathbf{x};\boldsymbol{\theta}_{t+1})$. Therefore, the whole learning can be considered to be based on a long, persistent sequence of distributions, and therefore the proposed algorithm is referred to as *persistent SMC* (PSMC). An alternative understanding of PSMC can be based on using standard SMC for sampling $p(\mathbf{x};\boldsymbol{\theta}_t)$ at each iteration. In standard SMC case, the sub-sequential distributions are:

$$p_h(\mathbf{x};\boldsymbol{\theta}_{t+1}) \propto p(\mathbf{x};\boldsymbol{\theta}_{t+1})^{\beta_h} \tag{14}$$

where $0 \leq \beta_H \leq \beta_{H-1} \leq \cdots \beta_0 = 1$. The schematic figures of standard SMC and PSMC are presented in Figure 1 where we can see a prominent difference between them, the continuity from $p_0(\mathbf{x};\boldsymbol{\theta}_t)$ to $p_H(\mathbf{x};\boldsymbol{\theta}_{t+1})$. Intuitively, PSMC can be seen as a linked version of SMC by connecting $p_0(\mathbf{x};\boldsymbol{\theta}_t)$ and $p_H(\mathbf{x};\boldsymbol{\theta}_{t+1})$. In addition, in our implementation of PSMC, to ensure adequate exploration, only half of the particles from $p_0(\mathbf{x};\boldsymbol{\theta}_t)$ are preserved to the next iteration; the other half particles are randomly initialized with a uniform distribution $\mathcal{U}^D$ (Figure 1(*b*)).

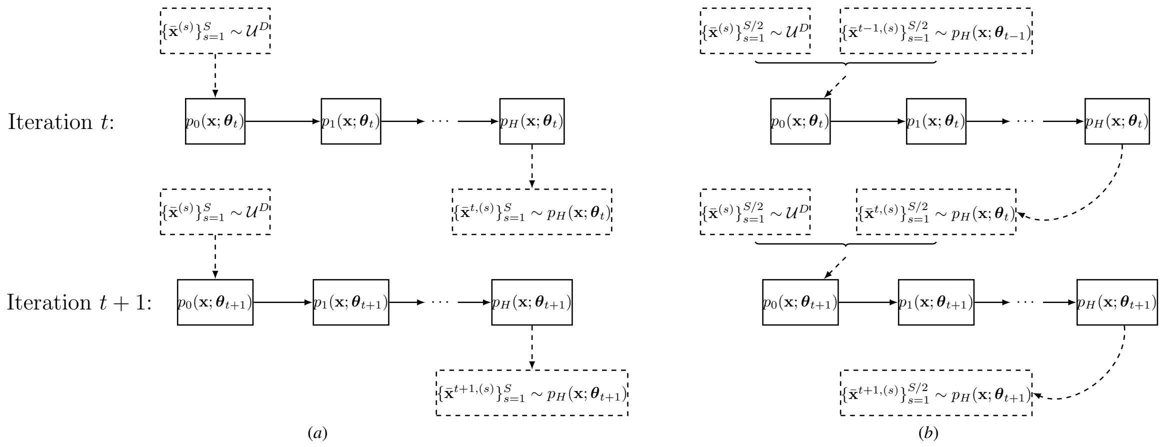

Iteration t:
Iteration $t+1$:
$\{\bar{\mathbf{x}}^{(s)}\}_{s=1}^{S} \sim \mathcal{U}^D$
$p_0(\mathbf{x};\boldsymbol{\theta}_t)$
$p_1(\mathbf{x};\boldsymbol{\theta}_t)$
$p_H(\mathbf{x};\boldsymbol{\theta}_t)$
$\{\bar{\mathbf{x}}^{t,(s)}\}_{s=1}^{S} \sim p_H(\mathbf{x};\boldsymbol{\theta}_t)$
$\{\bar{\mathbf{x}}^{t+1,(s)}\}_{s=1}^{S} \sim p_H(\mathbf{x};\boldsymbol{\theta}_{t+1})$
(a)
$\{\bar{\mathbf{x}}^{(s)}\}_{s=1}^{S/2} \sim \mathcal{U}^D$
$\{\bar{\mathbf{x}}^{t-1,(s)}\}_{s=1}^{S/2} \sim p_H(\mathbf{x};\boldsymbol{\theta}_{t-1})$
$\{\bar{\mathbf{x}}^{t,(s)}\}_{s=1}^{S/2} \sim p_H(\mathbf{x};\boldsymbol{\theta}_t)$
$\{\bar{\mathbf{x}}^{t+1,(s)}\}_{s=1}^{S/2} \sim p_H(\mathbf{x};\boldsymbol{\theta}_{t+1})$
(b)

Figure 1: The schematic figures of (a) standard sequential Monte Carlo and (b) persistent sequential Monte Carlo for learning UGMs. Solid boxes denote sequential distributions and solid arrows represent the move (resampling and MCMC transition) between successive distributions. Dashed boxes are particle sets and dashed arrows mean feeding particles into a SMC or sampling particles out of a distribution.

One issue arising in PSMC is the number of β_h, *i.e.* H, which is also a problem in parallel tempering and tempered transition[3]. Here, we employed the bidirectional searching method (Jasra et al., 2011). When we construct sub-sequential distributions as (13), the importance weighting for each particle is

$$\begin{aligned} w^{(s)} &= \frac{p_h(\bar{\mathbf{x}}^{(s)};\boldsymbol{\theta}_{t+1})}{p_{h-1}(\bar{\mathbf{x}}^{(s)};\boldsymbol{\theta}_{t+1})} \\ &= \exp\left(E(\bar{\mathbf{x}}^{(s)};\boldsymbol{\theta}_t)\right)^{\Delta\beta_h} \exp\left(E(\bar{\mathbf{x}}^{(s)};\boldsymbol{\theta}_{t+1})\right)^{-\Delta\beta_h} \qquad (15) \end{aligned}$$

where $\Delta\beta_h$ is the step length from β_{h-1} to β_h, *i.e.* $\Delta\beta_h = \beta_h - \beta_{h-1}$. We can also compute the ESS of a particle set as (Kong et al., 1994)

$$\sigma = \frac{(\sum_{s=1}^{S} w^{(s)})^2}{S\sum_{s=1}^{S} w^{(s)2}} \in \left[\frac{1}{S}, 1\right] \qquad (16)$$

Based on (15) and (16), we can see that, when a particle set is given, ESS σ is actually a function of $\Delta\beta_h$. Therefore, assuming that we set the threshold of ESS as σ^*, we can then find the biggest $\Delta\beta_h$ by using bidirectional search (see Algorithm 3) . Usually a small particle set is used in learning (mini-patch scheme), so it will be quick to compute ESS. Therefore, with a small amount of extra computation, the gap between two successive βs and the length of the distribution sequence in PSMC can be actively determined, which is a great advantage over the manual tunning in parallel tempering and tempered transition. By integrating all pieces together, we can write out a pseudo code of PSMC as in Algorithm 4.

3. Usually, there is no systematic way to determine the number of β_h in parallel tempering and tempered transition, and it is selected empirically.

Algorithm 3 Finding $\Delta\beta_h$

```
Input:   a particle set $\{\bar{\mathbf{x}}^{(s)}\}_{s=1}^{S}$, $\beta_{h-1}$
 1: $l \leftarrow 0$, $u \leftarrow \beta_{h-1}$, $\alpha \leftarrow 0.05$
 2: while $|u - l| \geq 0.005$ and $l \leq \beta_{h-1}$ do
 3:     compute ESS $\sigma$ by replacing $\Delta\beta_h$ with $-\alpha$ according to (16)
 4:     if $\sigma < \sigma*$ then
 5:         $u \leftarrow \alpha$, $\alpha \leftarrow (l + \alpha)/2$
 6:     else
 7:         $l \leftarrow \alpha$, $\alpha \leftarrow (\alpha + u)/2$
 8:     end if
 9: end while
Output:   Return $\Delta\beta_h = \max(-\alpha, -\beta_{h-1})$
```

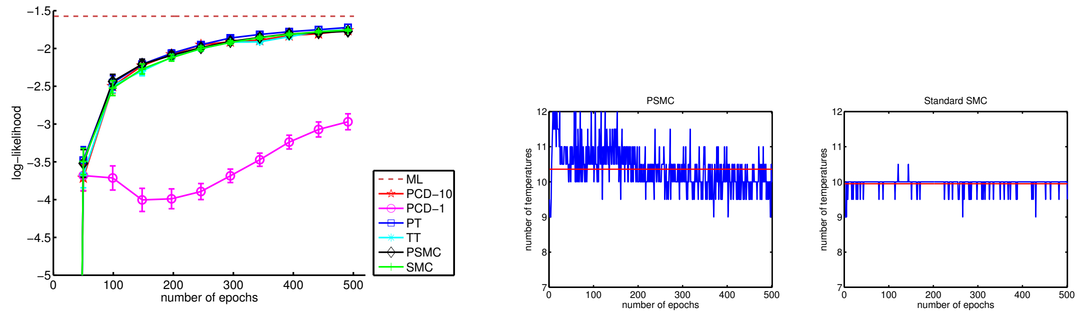
Algorithm 4 Learning with PSMC

```
Input:   a particle set $\{\mathbf{x}^{(m)}\}_{m=1}^{M}$, learning rate $\eta$
 1: Initialize $p(\mathbf{x}; \boldsymbol{\theta}_0), t \leftarrow 0$
 2: Sample particles $\{\bar{\mathbf{x}}_0^{(s)}\}_{s=1}^{S} \sim p(\mathbf{x}; \boldsymbol{\theta}_0)$
 3: while ! stop criterion do
 4:     $h \leftarrow 0$, $\beta_0 \leftarrow 1$
 5:     while $\beta_h < 1$ do
 6:         assign importance weights $\{w^{(s)}\}_{s=1}^{S}$ to particles according to (15)
 7:         resample particles based on $\{w^{(s)}\}_{s=1}^{S}$
 8:         compute the step length $\Delta\beta_h$ according to Algorithm 3
 9:         $\beta_{h+1} = \beta_h + \delta\beta$
10:         $h \leftarrow h + 1$
11:     end while
12:     Compute the gradient $\Delta\boldsymbol{\theta}_t$ according to (3)
13:     $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta\Delta\boldsymbol{\theta}_t$
14:     $t \leftarrow t + 1$
15: end while
Output:   estimated parameters $\boldsymbol{\theta}^* = \boldsymbol{\theta}_t$
```

6. Experiments

In our experiments, PCD, parallel tempering (PT), tempered transition (TT), standard SMC and PSCM were empirically compared on 2 different discrete UGMs, *i.e.* fully visible Boltzmann machines (VBMs) and restricted Boltzmann machines (RBMs). As we analyzed in section 4, large learning rate and high model complexity are two main challenges for learning UGMs. Therefore, two experiments were constructed to test the robustness of algorithms to different learning rates and model complexities separately. On one hand, one VBM was constructed with small size and tested with synthetic data. The purpose of the small-scale VBM is to reduce the effect of model complexity. In addition, the exact log-likelihood can be computed in this model. On the other hand, two RMBs were used in our second experiment, one is medium-scale and the other is large-scale.

learning rate $\eta_t = \frac{1}{100+t}$



(*a*) (*b*) (*c*)

Figure 2: The performance of algorithms with the first learning rate scheme. (a): log-likelihood *vs.* number of epochs; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

They were applied on a real-world database MNIST[4]. In this experiment, the learning rate was set to be small to avoid its effect. In both experiments, mini-patch of 200 data instances were used. When PSMC and SMC were run, $\sigma^* = 0.9$ was used as the threshold of ESS. We recorded the number of βs at each iteration in PSMC, and computed the average value H. In order to ensure the fairness of the comparison, we offset the computation of different algorithms. In PT, H βs were uniformly assigned between 0 and 1. In TT, similarly, H βs were uniformly distributed in the range $[0.9, 1]$[5]. Two PCD algorithms were implemented, one is with one-step Gibbs sampling (PCD-1) and the other is with H-step Gibbs sampling (PCD-H). In the second experiment, the computation of log-likelihoods is intractable, so here we employed an annealing importance sampling (AIS)-based estimation proposed by Salakhutdinov and Murray (2008). All methods were run on the same hardware and experimental conditions unless otherwise mentioned.

6.1. Experiments with Different Learning Rates

A Boltzmann machine is a kind of stochastic recurrent neural network with fully connected variables. Each variable takes binary value $\mathbf{x} \in \{-1, +1\}^D$. Using the energy representation (2), parameters $\boldsymbol{\theta}$ correspond to $\{\mathbf{W} \in \mathbb{R}^{D\times D}, \mathbf{b} \in \mathbb{R}^{D\times 1}\}$ and $\phi(\mathbf{x}) = \{\mathbf{x}\mathbf{x}^\top, \mathbf{x}\}$. Here we used a fully visible Boltzmann machine (VBM), and computed the log-likelihood to quantify performances. In this experiment, a small-size VBM with only 10 variables is used to avoid the effect of model complexity. For simplicity, $W_{ij_{i,j\in[1,10]}}$ were randomly generated from an identical distribution $\mathcal{N}(0, 1)$, and 200 training data instances were sampled. Here we tested all learning algorithms with 3 different learning rate schemes: (1) $\eta_t = \frac{1}{100+t}$, (2) $\eta_t = \frac{1}{20+0.5\times t}$, (3) $\eta_t = \frac{1}{10+0.1\times t}$. The learning rates in the three schemes were at different magnitude levels. The first one is smallest, the second is intermediate and the last one is relative large. For the first scheme, 500 epochs were run, and the log-likelihood *vs.* number of epochs plots of different learning algorithms are presented in

4. http://yann.lecun.com/exdb/mnist/index.html
5. In our experiment, we used a TT similar to that used by Salakhutdinov (2010) by alternating between one Gibbs sampling and one tempered transition.

learning rate $\eta_t = \frac{1}{20+0.5\times t}$

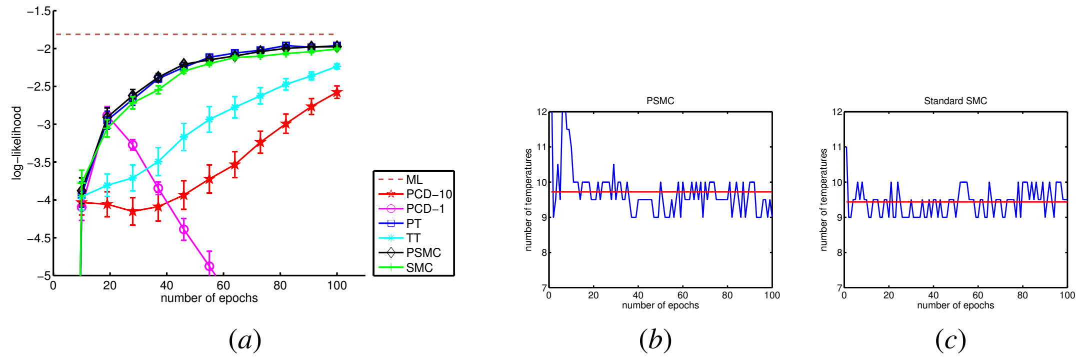


Figure 3: The performance of algorithms with the second learning rate scheme. (a): log-likelihood *vs.* number of epochs; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

learning rate $\eta_t = \frac{1}{10+0.1\times t}$

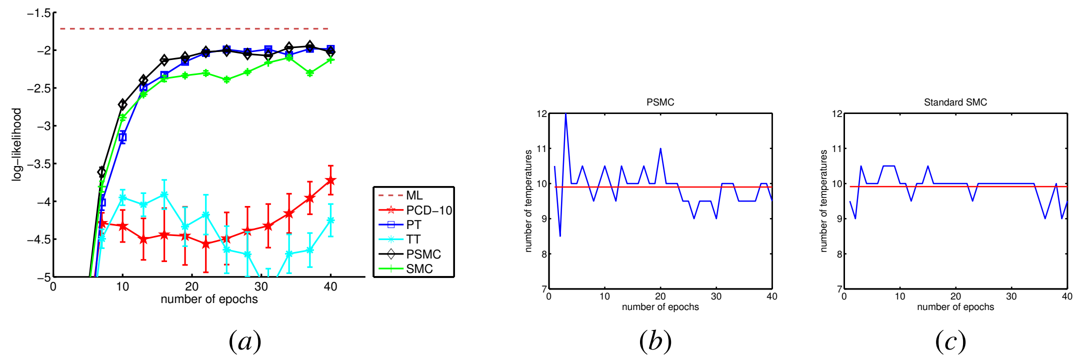
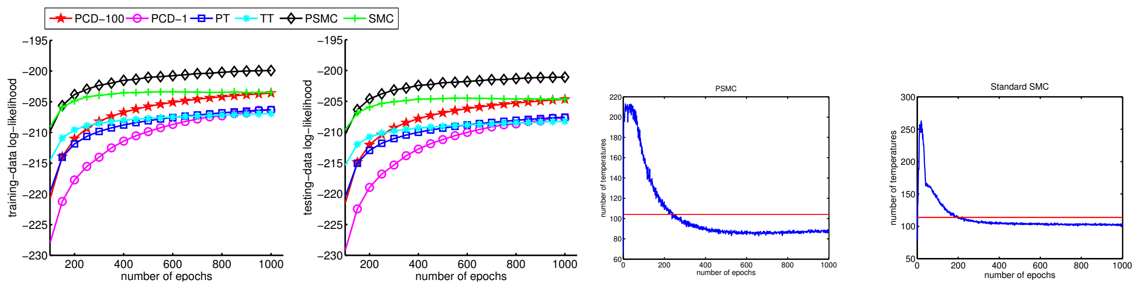


Figure 4: The performance of algorithms with the third learning rate scheme. (a): log-likelihood *vs.* number of epochs; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

Figure 2(*a*). The number of βs in PSMC and SMC are also plotted in Figures 2(*b*) and 2(*c*) respectively. We can see that the mean value H in PSMC is around 10, which is slightly higher than the one in SMC. For the second and third learning rate schemes, we ran 100 and 40 epochs respectively. All algorithms' performances are shown in Figure 3(*a*) and 4(*a*). We found that the number of βs in PSMC and SMC are very similar to those of the first scheme (Figures 3(*b*), 3(*c*), 4(*b*) and 4(*c*)). For all three schemes, 5 trials were run with different initial parameters, and the results are presented with mean values (curves) and standard deviations (error bars). In addition, maximum likelihood (ML) solutions were obtained by computing exact gradients (3). For better quantitative comparison, the average log-likelihoods based on the parameters learned from six algorithms and three learn-

Models		(Avg.) Log-Likelihoods					
(Size)	Learning rate schemes	PCD-1	PCD-H	PT	TT	SMC	PSMC
VBM	$\eta_t = \frac{1}{100+t}$	-1.693	-1.691	$\mathbf{-1.689}$	-1.692	-1.692	-1.691
(15)	$\eta_t = \frac{1}{20+0.5\times t}$	-7.046	-2.612	-1.995	-2.227	-2.069	$\mathbf{-1.891}$
	$\eta_t = \frac{1}{10+0.1\times t}$	-25.179	-3.714	-2.118	-4.329	-2.224	$\mathbf{-1.976}$
	MNIST						
RBM	training data	-206.3846	-203.5884	206.2819	-206.9033	-203.3672	$\mathbf{-199.9089}$
(784 × 10)	testing data	-207.7464	-204.6717	206.2819	-208.2452	-204.4852	$\mathbf{-201.0794}$
RBM	training data	-176.3767	-173.0064	-165.2149	-170.9312	-678.6464	$\mathbf{-161.6231}$
(784 × 500)	testing data	-177.0584	-173.4998	-166.1645	-171.6008	-678.7835	$\mathbf{-162.1705}$

Table 1: Comparison of Avg.log-likelihoods with parameters learned from different learning algorithms and conditions.



(a) (b) (c)

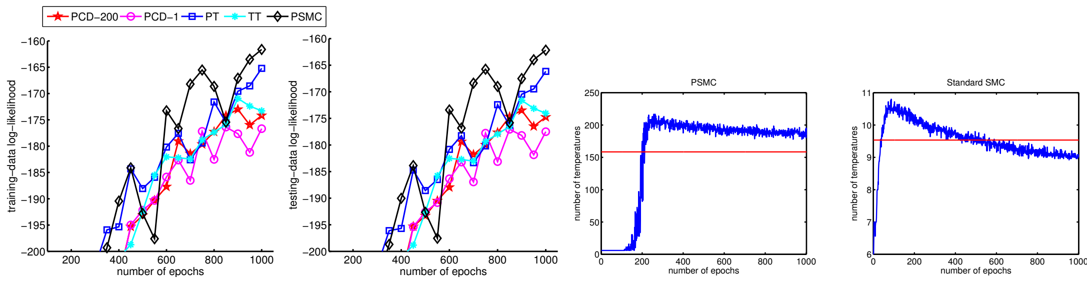
Figure 5: The performance of algorithms on the medium-scale RBM. (a): log-likelihood *vs.* number of epochs for both training images (left) and testing images (right) in the MNIST database; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

ing rate schemes are listed in the upper part of Table 1. The results of the first experiment can be summarized as follows:

1. When the learning rate was small, PT, TT, SMC, PSMC and PCD-10 worked similarly well, outperforming PCD-1 by a large margin.

2. When the learning rate was intermediate, PT and PSMC still worked successfully, which were closely followed by SMC. TT and PCD-10 deteriorated, while PCD-1 absolutely failed.

3. When the learning rate went to relatively large, the fluctuation patterns were obvious in all algorithms. Meanwhile, the performance gaps between PSMC and other algorithms was larger. In particular, TT and PCD-10 deteriorated very much. Since PCD-1 failed even worse in this case, its results are not plotted in Figure 4(a).



(*a*) (*b*) (*c*)

Figure 6: The performance of algorithms on the large-scale RBM. (a): log-likelihood *vs.* number of epochs for both training images (left) and testing images (right) in the MNIST database; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

6.2. Experiments with Models of Different Complexities

In our second experiment, we used the popular restricted Boltzmann machine to model handwritten digit images (with the MNIST database). RBM is a bipartite Markov network consisting of a visible layer and a hidden layer, it is a "restricted" version of Boltzmann machine with only interconnections between the hidden layer and the visible layer. Assuming that the input data are binary and N_v-dimensional, each data point is fed into the N_v units of the visible layer $\mathbf{v}$, and N_h units in hidden layer $\mathbf{h}$ are also stochastically binary variables (latent features). Usually, $\{0, 1\}$ is used to represent binary values in RBMs to indicate the activations of units.The energy function $E(\mathbf{v}, \mathbf{h})$ is defined as $E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^\top \mathbf{W}\mathbf{h} - \mathbf{h}^\top \mathbf{b} - \mathbf{v}^\top \mathbf{c}$, where $\mathbf{W} \in \mathbb{R}^{N_v \times N_h}$, $\mathbf{b} \in \mathbb{R}^{N_v \times 1}$ and $\mathbf{c} \in \mathbb{R}^{N_h \times 1}$. Although there are hidden variables in the energy function, the gradient of likelihood function can be written out in a form similar to (3) (Hinton, 2002). Images in the MNIST database are 28×28 handwritten digits, *i.e.* N_v=784. To avoid the effect of learning rate, in this experiment, a small learning rate scheme $\eta_t = \frac{1}{100+t}$ was used and 1000 epochs were run in all learning algorithms. Two RBMs were constructed for testing the robustness of learning algorithms to model complexity, one medium-scale with 10 hidden variables (*i.e.* $\mathbf{W} \in \mathbb{R}^{784\times10}$), the other large-scale with 500 hidden variables (*i.e.* $\mathbf{W} \in \mathbb{R}^{784\times500}$)[6]. Similarly to the first experiment, we first ran PSMC and SMC, and recorded the number of triggered βs at each iteration and their mean values (Figure 5(*b*), 5(*c*), 6(*b*) and 6(*c*)). For the medium-scale model, the number of βs in PSMC and SMC are similar (around 100). However, for the large-scale model, the mean value of $|\{\beta_0, \beta_1, \cdots\}|$ is 9.6 in SMC while 159 in PSMC. The reason for this dramatic change in SMC is that all 200 particles initialized from the uniform distribution were depleted when the distribution gets extremely complex. For other learning algorithms, H was set 100 and 200 in the medium- and large-scale cases, respectively. Since there are 60000 training images and 10000 testing images in the MNIST database, we plotted both training-data log-likelihoods and testing-data log-likelihoods as learning progressed (see Figure 5(*a*) and 6(*a*)). More detailed quantitative comparison can be seen in the lower part of Table 1. Similarly, we conclude the results of the second experiments as follows:

6. Since a small-scale model was already tested in the first experiment, we did not repeat it here.

1. When the scale of RBM was medium, PSMC worked best by reaching the highest training-data and testing-data log-likelihoods. SMC and PCD-100 arrived the second highest log-likelihoods, although SMC converged much faster than PCD-100. PT, TT and PCD-1 led to the lowest log-likelihoods although PT and TT raised log-likelihoods more quickly than PCD-1.

2. When the scale of RBM was large, all algorithms displayed fluctuation patterns. Meanwhile, PSMC still worked better than others by obtaining the highest log-likelihoods. PT ranked second, and TT ranked third, which was slightly better than PCD-200. PCD-1 ranked last. SMC failed in learning the large-scale RBM, so its results are not presented in Figure 6(a).

7. Conclusion

A SMC interpretation framework of learning UGMs was presented, within which two main challenges of the learning task were disclosed as well. Then, a persistent SMC (PSMC) learning algorithm was developed by applying SMC more deeply in learning. According to our experimental results, the proposed PSMC algorithm demonstrates promising stability and robustness in various challenging circumstances with comparison to state-of-the-art methods. Meanwhile, there still exist much room for improvement of PSMC, *e.g.* using adaptive MCMC transition (Schäfer and Chopin, 2013; Jasra et al., 2011), which suggests many possible directions for future work. Besides, although PSMC is expected to approach the maximum likelihood solution in learning UGMs, sometimes maximizing the posterior function is more desirable (*e.g.* when the prior is available), so it is also interesting to extend PSMC for maximum a posteriori learning.

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. 270273, Xperience.

References

Arthur U. Asuncion, Qiang Liu, Alexander T. Ihler, and Padhraic Smyth. Particle filtered MCMC-MLE with connections to contrastive divergence. In *ICML*, 2010.

Miguel A. Carreira-Perpinan and Geoffrey E. Hinton. On Contrastive Divergence Learning. In *AISTATS*, 2005.

Guillaume Desjardins, Aaron Courville, Yoshua Bengio, Pascal Vincent, and Olivier Delalleau. Tempered Markov Chain Monte Carlo for training of restricted Boltzmann machines. In *AISTATS*, 2010.

Charles J. Geyer. *Markov Chain Monte Carlo Maximum Likelihood.* Compuuting Science and Statistics:Proceedings of the 23rd Symposium on the Interface, 1991.

Mark S. Handcock, David R. Hunter, Carter T. Butts, Steven M. Goodreau, and Martina Morris. statnet: Software tools for the representation, visualization, analysis and simulation of network data. *Journal of Statistical Software*, pages 1548–7660, 2007.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

Ajay Jasra, David A. Stephens, Arnaud Doucet, and Theodoros Tsagaris. Inference for lévy-driven stochastic volatility models via adaptive sequential monte carlo. *Scandinavian Journal of Statistics*, 38:1–22, 2011.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Augustine Kong, Jun S. Liu, and Wing H. Wong. Sequential Imputations and Bayesian Missing Data Problems. *Journal of the American Statistical Association*, 89(425):278–288, March 1994.

Radford Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6:353–366, 1994.

H. Robbins and S. Monro. A Stochastic Approximation Method. *Ann.Math.Stat.*, 22:400–407, 1951.

Ruslan Salakhutdinov. Learning in markov random fields using tempered transitions. In *NIPS*, 2010.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *ICML*, 2008.

Christian Schäfer and Nicolas Chopin. Sequential monte carlo on large binary sampling spaces. *Statistics and Computing*, 23(2):163–184, 2013.

T. Tieleman. Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient. In *ICML*, pages 1064–1071, 2008.

T. Tieleman and G.E. Hinton. Using Fast Weights to Improve Persistent Contrastive Divergence. In *ICML*, pages 1033–1040. ACM New York, NY, USA, 2009.

Towards Sparsity and Selectivity: Bayesian Learning of Restricted Boltzmann Machine for Early Visual Features[*]

Hanchen Xiong, Sandor Szedmak,
Antonio Rodríguez-Sánchez, and Justus Piater

Institute of Computer Science, University of Innsbruck
{hanchen.xiong, sandor.szedmak, antonio.rodriguez-sanchez,
justus.piater}@uibk.ac.at

Abstract. This paper exploits how Bayesian learning of restricted Boltzmann machine (RBM) can discover more biologically-resembled early visual features. The study is mainly motivated by the sparsity and selectivity of visual neurons' activations in V1 area. Most previous work of computational modeling emphasize selectivity and sparsity independently, which neglects the underlying connections between them. In this paper, a prior on parameters is defined to simultaneously enhance these two properties, and a Bayesian learning framework of RBM is introduced to infer the maximum posterior of the parameters. The proposed prior performs as the lateral inhibition between neurons. According to our empirical results, the visual features learned from the proposed Bayesian framework yield better discriminative and generalization capability than the ones learned with maximum likelihood, or other state-of-the-art training strategies.

1 Introduction

Over the past decades, there have been a large volume of literature dedicated to model the statistics of natural images in biologically plausible ways. Especially, the primary visual cortex (V1) has been intensively studied and various computational models were proposed to reproduce its functionalities [10,6,12]. It has been well documented that mainly V1 simple cells perform an early stage processing of the visual input signal from the retina and the lateral geniculate nucleus (LGN). One important property of V1 simple cells is that their receptive fields are selective in terms of locations, orientations and frequencies, which can be modelled as Gabor filters. Another characteristic on V1 simple cells is that their activations are sparse. To be more clear, selectivity means that one neuron

[*] The authors would like to thank Dr. George Azzopardi for his helpful comments. 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.

only strongly responds to a small number types of stimuli while rarely responding to other types. Sparsity means that the population size of activated neurons should be small given a stimulus, i.e. only a tiny fraction of neurons are activated by a stimulus. Since selectivity and sparsity are interpreted as rareness in lifetime and population domain, sometimes they are also called "lifetime sparseness" and "population sparseness" respectively [12]. It has been hypothesized that the selective and sparse responses of visual neurons are due to certain redundancy reduction mechanism, with which the visual cortex is evolved to encode visual information as efficiently as possible [1]. Based on this hypothesis, a sparse coding strategy was proposed to enhance the coding efficiency, and has led to Gabor-like representations [10]. Although sparse coding has shown success in producing receptive fields similar to those of simple cells, yet it was pointed out that selectivity does not have to be correlated with sparseness in practice [12]. Moreover, it was even suspected that sparse activations of simple cells is only an epiphenomenon or side effect of selectivity [2]. (see section 3 for a detailed analysis). Recently, as another stream of feature learning, restricted Boltzmann machines (RBMs) have attracted increasingly more attention thanks to its success in many application domains [7]. However, the capability of RBMs is rather limited when learning receptive fields of V1 simple cells. To make inference and learning easier, there is no connection between hidden units in RBMs. Consequently, given visible data, all hidden units are conditionally independent to each other (see section 2). It can be easily envisioned that when RBMs are trained on natural images, many learned features will be rather distributed, unlocalized and repeated, which is far from the (selective and sparse) nature of the learning task.

Prior work have exploited different strategies to adapt RBMs towards learning selective and/or sparsely activated neurons [8,9,5] on visual input. However, most of them focus only on one property and does not ensure sparsity and selectivity simultaneously in reproduced neurons. Usually, these strategies are to impose certain regularization to bias learning. To overcome this deficiency, in this paper, we propose to encode an inductive bias about the task as prior probability on parameters. Then, the parameter estimation can be done within a consistent Bayesian learning framework, *i.e.* maximum a posterior (MAP). In particular, the prior probability on parameters encourages the diversity of neurons' receptive fields, which performs equivalently to the lateral inhibition between neurons. The MAP learning is achieved via a Markov chain Monte Carlo (MCMC)-based simulated annealing. In addition, due to the fact that the parameter space is high-dimensional and multi-modal, annealing importance sampling (AIS) and parallel tempering are employed in subroutines to avoid local maxima (see section 4). In section 5, we verify our Bayesian learning of RBMs on a benchmark database of natural images, comparing to maximum likelihood learning and other state-of-the-art learning strategies. Our empirical results demonstrate that neurons in our model display better sparsity and selectivity than in others; in addition, the features encoded by our neurons via Bayesian learning show better generalization capabilities than the ones from other learning methods.

2 Restricted Boltzmann Machine

The restricted Boltzmann machine (RBM) is a two-layer, bipartite neural network, it is a "restricted version" of the Boltzmann machine with only interconnections between hidden layers and visible layers. Input data is binary and N_v dimensional, they are fed into N_v units in the visible layer $\mathbf{v}$, N_h units in the hidden layer $\mathbf{h}$ are stochastic binary variables, *i.e.* $\mathbf{v} \in \{0,1\}^{N_v}$, $\mathbf{h} \in \{0,1\}^{N_h}$, the joint probability of $\{\mathbf{v}, \mathbf{h}\}$ is[1]:

$$p(\mathbf{v},\mathbf{h}) = \frac{1}{\mathbf{Z}} \exp(-E(\mathbf{v},\mathbf{h})) \qquad E(\mathbf{v},\mathbf{h}) = -\mathbf{v}^\top \mathbf{W}\mathbf{h} \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{N_v \times N_h}$ is the matrix of symmetry weights, $\mathbf{Z} = \sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}))$ is the partition function for normalization. Because of the restricted connections in RBMs, hidden units h_j are independent of each other conditioned on the visible data $\mathbf{v}$, and similarly, visible units v_i are conditionally independent of each other given $\mathbf{h}$. Given training data $\mathcal{D} = \{\mathbf{v}^{(l)}\}_{l=1}^{L}$, RBM can be learned by maximizing the average log-likelihood of $\mathcal{D}$:

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} \mathcal{L}(\mathcal{D}) = \arg\max_{\mathbf{W}} \frac{1}{L}\sum_{l=1}^{L} \big(\log \sum_{\mathbf{h}} p(\mathbf{v}^l, \mathbf{h})\big) \tag{2}$$

based on(1), the gradient of $\mathcal{L}(\mathcal{D})$ is computed as:

$$\nabla \mathcal{L}(\mathcal{D}) = \frac{1}{L}\sum_{l=1}^{L} [\mathbb{E}_{\mathbf{v}^{(l)} \in \mathbf{V}, \mathbf{h} \sim p(\mathbf{h}|\mathbf{v}^{(l)})}(\mathbf{v}^{(l)}\mathbf{h}^\top) - \mathbb{E}_{\mathbf{v},\mathbf{h} \sim p(\mathbf{v},\mathbf{h})}(\mathbf{v}\mathbf{h}^\top)] \tag{3}$$

where $\mathbb{E}_p(\cdot)$ denotes the expected values with respect to p. Obviously, the sampling $\mathbf{v},\mathbf{h} \sim p(\mathbf{v},\mathbf{h})$ makes learning practically infeasible because it requires a large number of Markov chain Monte Carlo (MCMC) iterations to reach equilibrium. Fortunately, we can compute an efficient approximation to the exact gradient: contrastive divergence (CD), which works well in practice [7]. By using CD$_k$, only a small number k steps are run in block Gibbs sampling (usually $k = 1$), and (3) can be approximated as:

$$\nabla \hat{\mathcal{L}}(\mathcal{D}) = \frac{1}{L}\sum_{l=1}^{L} [\mathbf{v}^{(l)} p(\mathbf{h}^{(l)+}|\mathbf{v}^{(l)})^\top - p(\mathbf{v}^{(l)-}|\mathbf{h}^{(l)+}) p(\mathbf{h}^{(l)-}|\mathbf{v}^{(l)-})^\top] \tag{4}$$

3 Bias Learning with Selectivity and Sparsity

Simple cells in V1 area are well known to be selective to locations, orientations and frequencies, and their activations are sparse [10] to visual stimuli. The concepts of selectivity and sparsity are illustrated in Figure 1(a), where each row (red) represent how one neuron selectively respond to different visual stimuli while each column (blue) describes how many neurons are activated by one stimulus. Although selectivity and sparsity are related at their average values,

[1] Bias vectors on visible and hidden units are omitted them for notation simplicity, but we would like to note that we use such biases in our experiments.

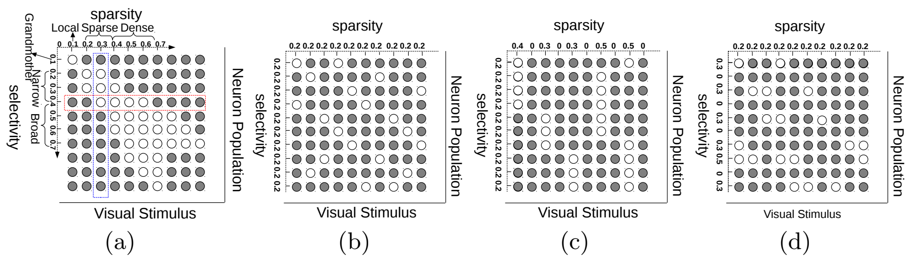


Fig. 1. Understanding sparsity and selectivity: white circles mean activations while gray circles denote inactivations. See text for more description.

they are not necessarily correlated [12]. Selective neurons cannot ensure sparse neuron coding (Figure 1(c)); sparsely activated neurons can also be not narrowly selective (Figure 1(d)).

Sparse group restricted Boltzmann machine (SGRBM) [9] is a RBM trained with the CD algorithm plus a $l1/l2$ norm regularization on the activations of neuron population. Although $l1/l2$ norm regularization can ensure sparsity, yet it can also lead to many "dead" (never respond) and "potential over-tolerant" (always respond) neurons (see Figure 1(d) and section 5). On the other hand, a selectivity-induced regularization was used in [8] by suppressing the average activation probability of each neuron to all training data. One limitation of this strategy, as argued in [5], is that decreasing average activation probabilities can not guarantee selectivity, instead, it will result in many similar neurons with uniformly low activation probabilities to all types of visual stimuli, which prone to be "dead" as well (see section 5). One closely related work to ours is proposed in [5], of which the essence is to tune the activation matrix (Figure 1) towards a target one that is both selective and sparse while maximizing likelihood.

Based on the analysis above, we can see that the motivation of sparsity is to better differentiate neurons while the goal of selectivity is to avoid "over-tolerant" neurons. Assume that there exist N types of visual stimuli and K neurons, and usually $N \gg K$. Obviously, the ideal selectivity rates of neurons is N/K. At the same time, for sparsity, we also want to prevent the existence of any duplicate or similar activations in the neuron population. The best scenario is that there is no overlap among the activations of different neurons (rows in Figure 1), *i.e.* K neurons respond to non-overlapping N/K types of visual stimuli respectively. In the RBM case, the weights W, to some extent, can represent the activation matrix. For so, a natural choice of biasing parameters is to diversify the columns of $\mathbf{W}$ as much as possible. Here we approach diversification by minimizing absolute cosine similarities among columns of $\mathbf{W}$:

$$\arg\min_{\mathbf{W}} \sum_{j=1}^{N_h} \sum_{k\neq j}^{N_h} \left| \frac{\mathbf{W}_{\cdot,j}^{\top}\mathbf{W}_{\cdot,k}}{||\mathbf{W}_{\cdot,j}||||\mathbf{W}_{\cdot,k}||} \right| \qquad (5)$$

where $\mathbf{W}_{\cdot j}$ denotes the jth column of $\mathbf{W}$. Note that the denominator in (5) is necessary, because eliminating it will generate many "dead" or *principal component* *analysis* (PCA)-like neurons. An extreme case is that the activation probabilities of neurons are exclusive to each other. Despite selectivity is not so obvious in (5), it can be imagined that it can be better minimized when $|W_{\cdot,j:\forall j\in N_h}|$ are small. Therefore, sparsity and selectivity are enhanced simultaneously by using diversity-induced bias (5) (Figure 1(b)).

4 Bayesian Learning of Restricted Boltzmann Machines

In contrast to the incremental updates composed of CD approximation and regularization-based gradients [8,9,5], we propose to train RBMs in a consistent Bayesian framework. Based on the discussion in the previous section, we can define the prior probability on parameters $p(\mathbf{W})$ as:

$$p(\mathbf{W}) \propto \exp(-\lambda \cdot \sum_{j=1}^{N_h}\sum_{k\neq j}^{N_h}\left|\frac{\mathbf{W}_{\cdot,j}^{\top}\mathbf{W}_{\cdot,k}}{||\mathbf{W}_{\cdot,j}||||\mathbf{W}_{\cdot,k}||}\right|) \tag{6}$$

then the parameters can be estimated via maximum a posterior (MAP):

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} p(\mathbf{W}|\mathcal{D}) = \arg\max_{\mathbf{W}} p(\mathbf{W})\prod_{l=1}^{L}\sum_{\mathbf{h}} p(\mathbf{v}^{(l)},\mathbf{h}|\mathbf{W}) \tag{7}$$

Since the derivative of (7) *w.r.t.* $\mathbf{W}$ can not be analytically computed, and (7) in general is not concave, here a Markov chain Monte Carlo (MCMC)-based simulated annealing is employed to find the optimal solution. In the basic Metropolis algorithm, a sample $\mathbf{W}'$ is accepted with probability $\min(1, p(\mathbf{W}'|\mathcal{D})/p(\mathbf{W}|\mathcal{D}))$ where:

$$\frac{p(\mathbf{W}'|\mathcal{D})}{p(\mathbf{W}|\mathcal{D})} = \frac{p(\mathbf{W}')}{p(\mathbf{W})}\frac{p(\mathcal{D}|\mathbf{W}')}{p(\mathcal{D}|\mathbf{W})} = \frac{p(\mathbf{W}')}{p(\mathbf{W})}\frac{\prod_{l=1}^{L}\sum_{\mathbf{h}} p(\mathbf{v}^{(l)},\mathbf{h}|\mathbf{W}')}{\prod_{l=1}^{L}\sum_{\mathbf{h}} p(\mathbf{v}^{(l)},\mathbf{h}|\mathbf{W})} \tag{8}$$

Because of the special structure of RBM, the term $\sum_{\mathbf{h}} p(\mathbf{v},\mathbf{h}|\mathbf{W})$ can be written in a polynomial form as:

$$p(\mathbf{v}|\mathbf{W}) = \sum_{\mathbf{h}} p(\mathbf{v},\mathbf{h}|\mathbf{W}) = \frac{1}{\mathbf{Z}(\mathbf{W})}\prod_{j=1}^{N_h}\exp(1+\mathbf{v}^{\top}\mathbf{W}_{\cdot,j}) \tag{9}$$

Consequently, (8) can be further expanded as:

$$\frac{p(\mathbf{W}'|\mathcal{D})}{p(\mathbf{W}|\mathcal{D})} = \frac{p(\mathbf{W}')}{p(\mathbf{W})}\Big(\frac{\mathbf{Z}(\mathbf{W})}{\mathbf{Z}(\mathbf{W}')}\Big)^N \exp\Big\{\sum_{l=1}^{L}\sum_{j=1}^{N_h}\mathbf{v}^{(l)\top}(\mathbf{W}'_{\cdot,j}-\mathbf{W}_{\cdot,j})\Big\} \tag{10}$$

Since (10) is invariant to different scales of $\mathbf{W}$, without loss of generality, we constraint $\forall w_{ij}\in\mathbf{W}, w_{i,j}\in[-1,+1]$. One difficulty in computing (10) is the ratio of normalization terms $\frac{\mathbf{Z}(\mathbf{W})}{\mathbf{Z}(\mathbf{W}')}$. Instead of computing it analytically, we use a tractable approximation of it via *annealing importance sampling* (AIS) [11]. Basically, importance sampling can be used for estimating the ratio:

$$\frac{\mathbf{Z}(\mathbf{W})}{\mathbf{Z}(\mathbf{W}')} = \frac{\sum_{\mathbf{v}} p(\mathbf{v}|\mathbf{W})}{\sum_{\mathbf{v}} p(\mathbf{v}|\mathbf{W}')} = \sum_{\mathbf{v}}\frac{p(\mathbf{v}|\mathbf{W})}{p(\mathbf{v}|\mathbf{W}')}\frac{p(\mathbf{v}|\mathbf{W}')}{\sum_{\mathbf{v}} p(\mathbf{v}|\mathbf{W}')} = \mathbb{E}_{p(\mathbf{v}|\mathbf{W}')}(\frac{p(\mathbf{v}|\mathbf{W})}{p(\mathbf{v}|\mathbf{W}')}) \tag{11}$$

However, the estimation will be poor if $\mathbf{W}$ and $\mathbf{W}'$ are not close. By contrast, AIS constructs many intermediary distributions between $p(\mathbf{v}|\mathbf{W}')$ and $p(\mathbf{v}|\mathbf{W})$ as: $p_s(\mathbf{v}) \propto p(\mathbf{v}|\mathbf{W}')^{1-\alpha_s} p(\mathbf{v}|\mathbf{W})^{\alpha_s}$ with $0 < \alpha_0 < \alpha_1 < \cdots < \alpha_s < \cdots < \alpha_S = 1$. Then one AIS run is as follows:

```
1.Initialize $\mathbf{v}_0^{(m)} \sim p_0(\mathbf{v})$
2.for $s = 1 \to S$, sample $\mathbf{v}_s^{(m)}$ give $\mathbf{v}_{s-1}^{(m)}$ with one Gibbs sampling w.r.t. $p_s(\mathbf{v})$;
3.$w^{(m)} = \frac{p_1(\mathbf{v}_1^{(m)})}{p_0(\mathbf{v}_1^{(m)})} \frac{p_2(\mathbf{v}_2^{(m)})}{p_1(\mathbf{v}_2^{(m)})} \cdots \frac{p_S(\mathbf{v}_S^{(m)})}{p_{S-1}(\mathbf{v}_S^{(m)})}$.
```

When M runs of AIS are implemented, the ratio can be estimated as:

$$\frac{\mathbf{Z}(\mathbf{W})}{\mathbf{Z}(\mathbf{W}')} \approx \frac{1}{M} \sum_{m=1}^{M} w^{(m)} \tag{12}$$

In addition, to avoid being trapped in local maxima, we construct the state transition of a Markov chain as a mixture of a local Metropolis kernel (10) and an independent Metropolis-Hasting kernel. To better explore the sampling space, the uniform distribution $\mathcal{U}$ on $\mathbf{W}$ is set as the Metropolis-Hasting kernel. Therefore, the whole sampling is a weighted combination of local exploitation and global exploration, and here we use the mixture weight $\eta = 0.5$. At iteration n, the invariant distribution which the Markov chain is subject to is $p(\mathbf{W}|\mathcal{D})^{1/T_n}$, where T_n is a decreasing temperature schedule. When $T_n \to 0$, the Markov chain can hardly move and the still state will be used as the maximum. Usually $\mathbf{W}$ is high-dimensional (with large number of neurons and high-dimensional visual input), so the parameter space can be rather complicated, *e.g.* sharp with many isolated modes, and simulated annealing based on one single Markov chain is unreliable. One simple way is to run multiple Markov chains in parallel, and pick the states of one chain which lead to the best result. However, a better strategy is *parallel tempering* [3]. $R+1$ Markov chains are constructed under different initial temperatures $\{p_r(\mathbf{W}|\mathcal{D}) \propto p(\mathbf{W}|\mathcal{D})^{\beta_r}\}_{r=0}^{R}$, $0 \leq \beta_R < \cdots \beta_r < \cdots \beta_0 = 1$, β_0 is referred to the base distribution, and others correspond to more flat distributions smoothed with different temperatures. As the simulated annealing on differently tempered Markov chains progress, the states of neighbouring chains $\mathbf{W}^r$, $\mathbf{W}^{r+1}$ can be swapped with probability:

$$\min(1, \frac{p_r(\mathbf{W}^{r+1}|\mathcal{D}) p_{r+1}(\mathbf{W}^r|\mathcal{D})}{p_{r+1}(\mathbf{W}^{r+1}|\mathcal{D}) p(\mathbf{W}^r|\mathcal{D})}) = \min(1, \exp\{\sum_{l=1}^{L} \sum_{j=1}^{N_h} (\beta_r - \beta_{r+1}) \mathbf{v}^{(l)\top} (\mathbf{W}_{\cdot,j}^{r+1} - \mathbf{W}_{\cdot,j}^{r}\})) \tag{13}$$

5 Experiments

To evaluate the proposed learning strategy, a benchmark database [10] was used for training. 100000 small patches (size 14 × 14) were extracted from random positions of ten whitened images. A sigmoid function was applied on the pixel intensities to fit their values in the range $[0, 1]$; in addition, the patches with variances smaller than 0.1 were filtered out to accelerate training. For comparison, three additional RBMs were trained by using the CD algorithm, the CD

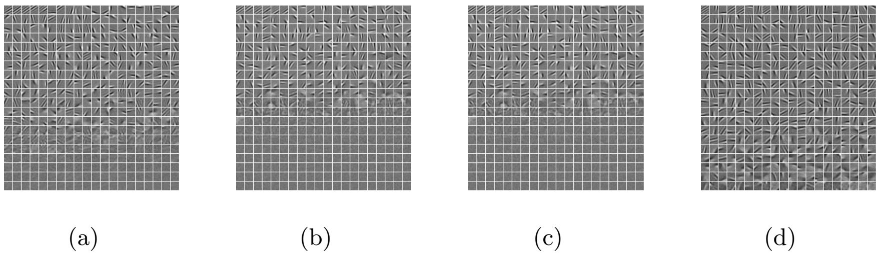


Fig. 2. The receptive fields of neurons learned from (a) CD algorithm, (b) sparse CD, (c) selective CD and (d)our Bayesian strategy. See text for more description.

	CD	Sparse CD [9]	Selective CD[8]	Bayesian Learning
# Dead Neurons	**0**	68	88	**0**
Ave. Selectivity	0.4582	0.3552	0.4090	**0.3221**
Ave. Sparsity	0.3749	0.1883	0.1939	**0.1671**
Error rate on MNIST[%]	27.21	17.54	19.32	**14.21**

Fig. 3. Performance of different learning methods.

algorithm with sparse regularization (sparse CD) [9] and the CD algorithm with selectivity regularization (selective CD) [8]. For each RBM, 200 hidden neurons were learned and their receptive fields are presented in Figure 2. We can see that many neurons' receptive fields learned from the CD algorithm (Figure 2(a)) are very vague and unlocalized, compared to which, the neurons' receptive fields learned from sparse CD and selective CD (Figure 2(b) and 2(c)) look "clearer" and "sharper". However, both sparse CD and selective CD led to many useless, "dead" neurons. The neurons obtained from our Bayesian learning strategy display rather diverse receptive fields and there seems no "dead" neuron (Figure 2(d)). We roughly obtained the number of "dead" neurons by counting the number of neurons whose maximal activation probabilities to all training visual stimuli is smaller than 0.1, and the results are in the first row of Figure 3.

Selectivity and sparsity are usually measured using activity ratio [4]. For a neuron, its selectivity is computed across all L input visual stimuli: $selectivity = \left(\sum_{l=1}^{L} r_l/L\right)^2 / \left(\sum_{l=1}^{L} r_l^2/L\right)$ where r_l is the activation rate of the neuron given the lth stimulus. The sparsity of activations by one stimulus is computed across all N_h neurons: $sparsity = \left(\sum_{j=1}^{N_h} r_j/N_h\right)^2 / \left(\sum_{j=1}^{N_h} r_j^2/N_h\right)$. We computed the selectivity and sparsity of 4 RBMs on the MNIST patch dataset[2], which contains digit images. Although natural images and digit images are two absolutely different visual domains, we believe that early features encoded in neurons should be able to successfully adapt from one domain to the other. The results were presented Figure 3. It can be seen that our Bayesian learning method yields bet-

[2] Available on http://yann.lecun.com/exdb/mnist.

ter selectivity and sparsity to other cases. Furthermore, to check the practical effectiveness of learned neurons, we use them as basis filters on the digit images for a multi-classification task. Given a digit image, the activations of hidden neurons are computed as input of a softmax function, and its corresponding label is output. The testing results with four sets of neurons are presented in the bottom part of Figure 3. We can see that the features from Bayesian learning yield lower average test error than others, which suggests superior discriminative and generalization capability.

6 Conclusion

A Bayesian learning framework for RBM was put forward based on many state-of-the-art approximation techniques. To mimic V1 simple cells, a diversity-induced prior was introduced on RBMs' parameters, and maximum a posterior learning yields better results than other learning strategies. In particular, the features encoded in learned neurons display nice discriminative and generalization property for domain adaption. As a possible future work direction, we are studying more sophisticated priors to approach other simple neurons' properties.

References

1. Barlow, H.B.: Unsupervised learning. Neural Computation 1, 295–311 (1989)
2. Berkes, P., White, B., Fiser, J.: No evidence for active sparsification in the visual cortex. In: NIPS. pp. 108–116 (2009)
3. Earl, D.J., Deem, M.W.: Parallel tempering: Theory, applications, and new perspectives. Phys. Chem. Chem. Phys. 7, 3910–3916 (2005)
4. Franco, L., Rolls, E.T., Aggelopoulos, N.C., Jerez, J.M.: Neuronal selectivity, population sparseness, and ergodicity in the inferior temporal visual cortex. Biological Cybernetics 96(6), 547–560 (2007)
5. Goh, H., Thome, N., Cord, M.: Biasing restricted Boltzmann machines to manipulate latent selectivity and sparsity. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2010)
6. van Hateren, J.H., van der Schaaf, A.: Independent component filters of natural images compared with simple cells in primary visual cortex. Proceedings of the Royal Society B: Biological Sciences 265(1394), 359C366 (1998), pMC1688904
7. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (Jul 2006)
8. Lee, H., Ekanadham, C., Ng, A.Y.: Sparse deep belief net model for visual area v2. In: NIPS (2007)
9. Luo, H., Shen, R., Niu, C., Ullrich, C.: Sparse group restricted boltzmann machines. In: AAAI (2011)
10. Olshausen, B., Field, D.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381, 607–609 (1996)
11. Salakhutdinov, R., Murray, I.: On the quantitative analysis of deep belief networks. In: ICML (2008)
12. Willmore, B., Tolhurst, D.: Characterising the sparseness of neural codes. Network:Comput.Neural Syst. 12, 255–270 (2001)