SEVENTH FRAMEWORK PROGRAMME

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

XPERIENCE.ORG

Deliverable Number:	D2.3.2
Deliverable Title:	Affordances and Categories (II): Report or scientific publication on generalization and creation of categories as well as grounding new categories through learning by imitation
Type (Internal, Restricted, Public):	PU
Authors:	J. Piater, H. Xiong, A. Ude, D. Kraft, N. Krüger
Contributing Partners:	UIBK, JSI, SDU

Contractual Date of Delivery to the EC:	31-01-2014
Actual Date of Delivery to the EC:	08-02-2014

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 two tasks that address the learning of categories:

Task 2.3.3 Exploration based learning of object categories

Task 2.3.4 Grounding new categories through learning by demonstration

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

Category formation is a fundamental ingredient of intelligence. Categories – including sensorimotor contingencies between an actor and objects acted upon (*affordances*) – provide abstractions essential for generalization across isolated experiences. Thus, categories are useful to the extent that they have predictive value for future experiences. Organizing a set of experiences into categories is generally difficult, as it requires the identification of predictive attribute sets, which is a combinatorial problem. Thus, while useful categorizations may be difficult to discover by pure exploration, this process can sometimes greatly benefit from demonstration and instruction.

This report presents recent work within the Xperience project on the acquisition and modeling of categories. Since all work presented here has already been published, we limit this report to concise overviews (Chapter 2) referring to the actual content in publications attached to the report.

Section 2.1 addresses the problem of learning shape features indicative of push affordances. The presented method produces features predictive of the result of various pushing actions (implicitly categorizing them into affordance categories), allowing a robot to make predictions of action results on objects it has never seen before.

Section 2.2 demonstrates how categories of spatial relations between objects can be learned that are predictive of how one object can be used to get hold of the other. The scenario is taken from a key stage in child development, where infants learn that they can get hold of an out-of-reach object using rake or by dragging the support on which that object rests.

In Section2.3 we address the problem of representing shapes within a category. For example, dogs come in various shapes and sizes, but all have four legs, a head and a tail. A useful category representation should represent this fact explicitly and represents objects in a way that identifies these common parts. We present a method that first aligns disparate shapes in a Reproducing Kernel Hilbert Space, and then identifies common parts using a novel latent-variable model that models spatial aspects using a Markov Random Field.

Chapter 2

Description of Results

2.1 Bootstrapping Object Categories By Push Affordances

Using an experimental platform that gathers 3-D data from the Kinect RGB-D sensor, as well as push action trajectories demonstrated by a human and acquired by a magnetic tracking system, we address the issues of learning new object affiordances using an action-grounded 3-D feature descriptor [RU13]. Rather than using pose-invariant visual features, as is often the case with object recognition, we ground the features of objects with respect to their manipulation, that is, by using shape features that describe the surface of an object relative to the push contact point and direction. Using this setup, object push affordance learning trials are performed by a human and both pre-push and post-push object features are gathered, as well as push action trajectories. A self-supervised multi-view online learning algorithm is employed to bootstrap both the discovery of affordance classes in the post-push view, as well as a discriminative model for predicting them in the pre-push view. Experimental results demonstrate the effectiveness of self-supervised class discovery, class prediction and feature relevance determination on a collection of unknown objects.

2.2 Formation of Spatial Relation Categories

In this section we describe our work on using visual data, from a simulated environment, to investigate the learning of spatial relations between objects and the categories found there (e.g., on-top, inside).

In previous work (D3.1.1, [3, 1, 2]) we have introduced a two track model representing infant development (see [3] and Figure 2.1) and we have explored some of the mechanism present on the behavioural track to learn means end schemas and to differentiate a generic schema (e.g., pull an object) generating a new means end schema (e.g., pull objectA to bring objectB, which is placed on top of objectA, into reach) and to adjust the corresponding pre-conditions [1, 2]. In [2] we also already investigated how to learn visual stimuli that predict if the differentiated schema is applicable (e.g., pull with object on top can be used if there is an objectB on top of objectA) or not using a very simple visual representation (using the objects' centre of gravity and orientation axis).

The work presented here [FAG$^+$13, FMM$^+$14] focuses on the representational track (see Figure 2.1(top)). We extended our previous work by (a) using a much more sophisticated visual representation and (b) by learning spatial relations (e.g., on-top) instead of when the differentiated schemas (e.g., pulling with object on-top) succeed. The new visual representation used is based on the previously presented multi-dimensional histograms (see [4] and Figure 2.2 for an example histogram used for spatial relation categories).

To explore the learning of spatial relation categories (inside, onto, rakeable, almost rakeable, not rakeable) contrary to our other work mentioned in this section, we used hand labeled data. We therefore placed objects in various relative positions and decided manually into which category this situation falls. Based on these situations we were able to generate multi-dimensional histograms that describe the relations between visual features belonging to the one object and features belonging to the other object (see Figure 2.2). We applied different histogram types and preprocessing techniques before feeding these

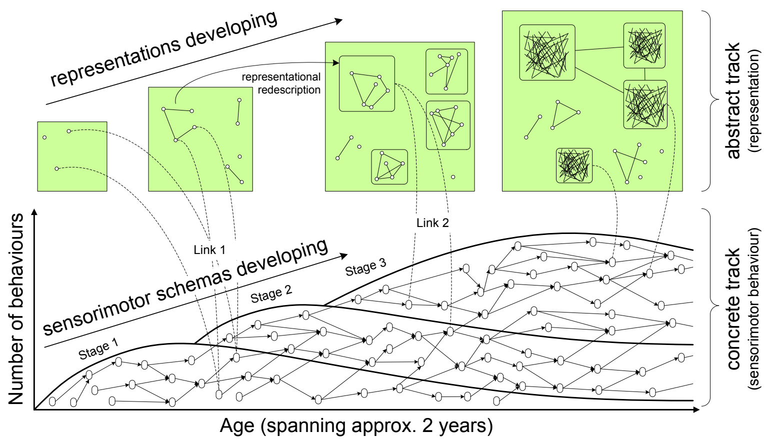

representations developing
representational redescription
abstract track (representation)
Link 1
Link 2
sensorimotor schemas developing
Stage 1
Stage 2
Stage 3
Number of behaviours
concrete track (sensorimotor behaviour)
Age (spanning approx. 2 years)

Figure 2.1: Conceptual diagram, overviewing infant developments leading to tool use (taken from [3]).

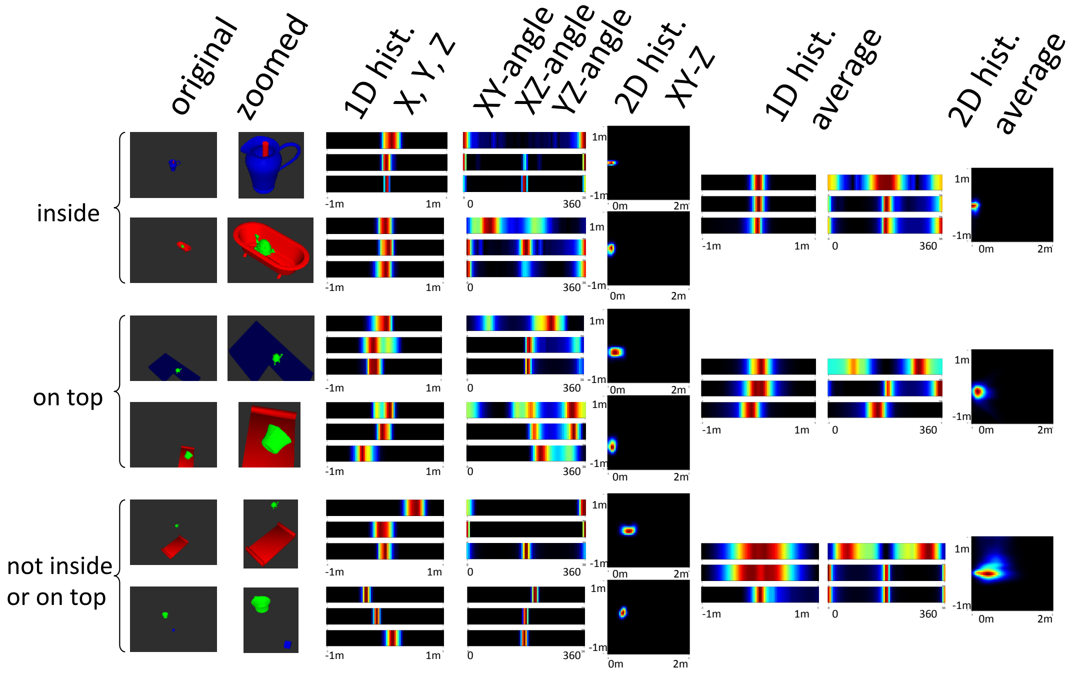


Figure 2.2: Obtained histograms. On the left side examples of the specific spatial relation categories (”inside”, ”on top” and ”not inside and not on top”) are shown. The middle shows the histograms produced by these specific examples while the right shows the histogram generated by all the training samples belonging to the category. This figure illustrates comparatively how well the different histograms can discriminate between the various relationships we have experimented with. (Taken from [FMM$^+$14] where an extended version of this figure can be found.)

histograms and labels to a random forest learning mechanism. For each category we were were able to achieve a classification rate significantly above 90 % with the right choice of histogram.

In the future we will revisit the learning of visual representations for the success of given schemas (e.g., pulling with object on top, lifting with object on top) and then use the models learned for these individual schemas to come up with a combined visual representation for the underlying category (e.g., on top), see ”representational redescription” in Figure 2.2. These more generic models will then also be used for new actions and allow a drastically reduced learning time.

2.3 3D Object Category Modeling

We developed a full learning framework to learn part-based models for different categories of 3D objects. The training data are labeled mesh or point cloud files, and labels are the identities of categories. At first, objects belonging to the same category are collected and aligned (using an original method [XSP13b]). Then unsupervised learning is performed to discover common parts shared by different instances of the same category, providing a probabilistic structural model for each part. The unsupervised learning algorithm is inspired by latent Dirichlet allocation (LDA), which is widely used to extract latent topics from document collections. In our application, we transform the 3D space into discrete grids, where the aligned point clouds can be considered as documents of 3D words, and parts correspond to topics. However, different from document topic modeling, spatial and structural coherence has to be taken into account in our object modeling case. Therefore, we extended LDA with a Markov random field over the labels of the points, and an extra spatial parameter to enhance the smoothness of discovered parts. We refer to the extended model as Spatial latent Dirichlet Markov random fields (SLDMRF). The empirical results [XSP13a] show that SLDMRF can provide much more consistent part segmentation and modeling (i.e., parts discovered are consistent for all instances) than LDA. The learned category models can also be further used for various tasks, e.g. new object recognition, segmentation and pose estimation [XSP13a].

References

[1] Severin Fichtl, John Alexander, Frank Guerin, Jimmy Alison Jørgensen, Dirk Kraft, and Norbert Krüger. Rapidly learning preconditions for means-end behavior using active learning. In *IEEE Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 1–2, 2012.

[2] Severin Fichtl, John Alexander, Dirk Kraft, Jimmy Alison Jørgensen, Norbert Krüger, and Frank Guerin. Learning object relationships which determine the outcome of actions. *Paladyn*, 3(4):188–199, 2012.

[3] F. Guerin, N. Krüger, and D. Kraft. A survey of the ontogeny of tool use: from sensorimotor experience to planning. *IEEE Transactions on Autonomous Mental Development*, 5(1):18–45, 2013.

[4] Wail Mustafa, Nicolas Pugeault, and Norbert Krüger. Multi-view object recognition using view-point invariant shape relations and appearance information. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

Attached Articles

[FAG$^+$13] Severin Fichtl, John Alexander, Frank Guerin, Wail Mustafa, Dirk Kraft, and Norbert Krüger. Learning spatial relations between objects from 3D scenes. In *2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, 2013.

[FMM$^+$14] Severin Fichtl, Andrew Mcmanus, Wail Mustafa, Norbert Krüger, and Frank Guerin. Learning spatial relationships from 3D vision using histograms. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. (accepted).

[RU13] Barry Ridge and Aleš Ude. Action-grounded push affordance bootstrapping of unknown objects. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2791–2798, 2013.

[XSP13a] Hanchen Xiong, Sandor Szedmak, and Justus Piater. 3D Object Class Geometry Modeling with Spatial Latent Dirichlet Markov Random Fields. In *35th German Conference on Pattern Recognition (former DAGM)*, volume 8142 of *LNCS*, pages 51–60. Springer, 9 2013.

[XSP13b] Hanchen Xiong, Sandor Szedmak, and Justus Piater. Efficient, General Point Cloud Registration With Kernel Feature Maps. In *Tenth Conference on Computer and Robot Vision*, pages 83–90, 5 2013.

Learning Spatial Relations Between Objects From 3D Scenes

Severin Fichtl, John Alexander, Frank Guerin
Computing Science
University of Aberdeen, Aberdeen AB24 3UE, Scotland
Email: f.guerin@abdn.ac.uk

Wail Mustafa, Dirk Kraft, Norbert Krueger
Maersk Mc-Kinney Moller Institute
Niels Bohrs Allé 1, DK-5230 Odense M, Denmark
Email: norbert@mmmi.sdu.dk

I. Introduction

Ongoing cognitive development during the first years of human life may be the result of a set of developmental mechanisms which are in continuous operation [1]. One such mechanism identified is the ability of the developing child to learn effective preconditions for their behaviours. It has been suggested [2] that through the application of behaviours involving more than one object, infants begin to learn about the relations between objects.

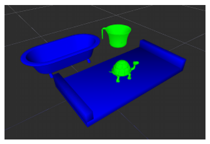
We consider a precondition to be a learnt decision rule by which some features of the environment are used to predict the successful outcome of a behaviour. This can be used as a planning operator to allow a robot to sequence learnt actions to achieve a goal. The limited scope of this definition allows us to approach the problem computationally. This concept of a precondition is loosely related to the notion of an affordance [3] used as a planning operator, which has been well studied within the field of developmental robotics (see e.g. [4], [5])

Learning a precondition for a motor action from raw sensor data is challenging as it may take many thousands of examples to learn an effective rule. For this reason we first perform an abstraction to convert data into a form which simplifies the learning procedure.

In this work, we learn a limited number of abstractions which can then be used to form preconditions for motor actions. These abstractions take the form of spatial relations amongst objects. We consider three "classes" of spatial relation: The objects either are *separated* from, *on-top* of, or *inside* each other. We have tackled this same problem in previous work [6]. Here we report on recent improved results using a novel application of histograms to visually recognise a spatial relation between objects in the environment. Using this histogram based approach we are able to report a very high rate of success when the system is asked to recognise a spatial relation.

II. Learning Spatial Relations

For learning spatial relations between objects we used data from a sophisticated 3D vision system inside the physically realistic simulator RobWorkSim [7]. In our experiments we use 4 different household objects (see Figure 1a). Using these objects we are able to design combinations of object pairs accounting for each of the three classes of spatial relation.


(a) Coffee cup, turtle, tray and bathtub.

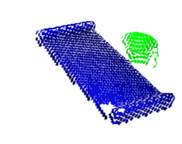

(b) Texlet representation of a scene.

Fig. 1: Objects in the simulated environment.

To collect samples, we placed pairs of objects in the simulated environment and used a Kinect-based vision system to create a representation of the scene and segment the objects.

For the *separated* class, two objects were placed randomly in the scene with the distance between the objects always greater than 0. The objects pairings were: the coffee cup next to the tray, the turtle next to the tray and the turtle next to the bathtub. For the *on-top* class, the tray was placed randomly in the scene and then either the coffee cup or the turtle was placed randomly on-top of the tray. The objects parings were: the coffee cup on-top of the tray and the turtle on-top of the tray. For the *inside* class the bathtub was placed randomly in the scene and the turtle was placed randomly inside the bathtub. See Figure 2 for an illustrated example of each of the three classes.

Our system uses Kinect-based vision [8] to extract information about objects in the scene. Kinect produces a depth map which describes the distance from the camera to each point of the surfaces visible to the camera system. It is important to note that within our simulator, the Kinect device includes realistic noise, allowing for more realistic data about the depth to the objects the scene. Using the picture of the scene and the depth map, our vision system calculates a 3D point cloud. Based on this 3D point cloud and the colour information of the scene, our vision system creates surface patches (called textlets) as shown in Figure 1b. These texlets describe the surfaces within the scene with additional information, such as the position, the orientation and colour of the surface [9].

In the scene, the objects used were each a uniform colour and the colour of each object was distinct. This allowed for a system of purely colour based segmentation. Although this is a simplification it is justified given the scope of this work. After segmentation, each object is assigned its unique set of texlets. A texlet representation of a scene with two segmented objects can be seen in Figure 1b.

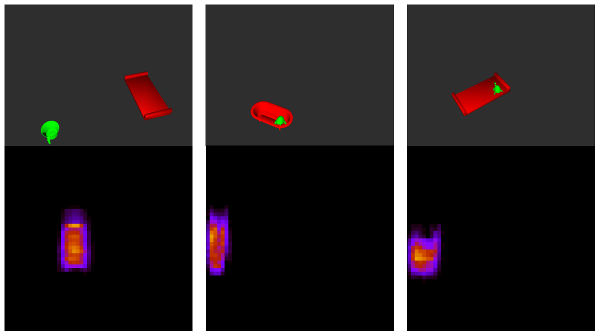


Fig. 2: Three texlet representations and the corresponding histograms.

We create 2D histograms which store relevant information about the spatial relations between objects. To extract this information from the sets of texlets, we calculate two distance measures between all texlets of one object to all texlets of another object: The absolute distance of a pair of texlets along the xy plane and the relative distance of the first object's texlets to the second object's on the z-axis, such that if the first texlet is above the second texlet, the distance is positive, otherwise negative. Since we always consider pairs of objects, if the first object has n texlets and the second has m texlets, this results in a nm vector for each measure considered. These distances are used to fill the histograms, such that the the x-axis is the absolute xy distance and the y-axis of the histogram is the relative distance between textlets along the z-axis. The histograms have 50 bins per axis, experientially this value produced the best results. The x-axis runs from 0mm to 2000mm, and the y-axis from $\pm$150mm. Example histograms are given in Figure 2.

For learning to differentiate the spatial relations between objects, we used a Random Forest (RF) classifier [10]. The RF had 100 trees, an overlap of 0.9 and 400 inputs per tree. For each class we used 5400 samples to build a training set. The training set contained samples from all configurations of each class in equal numbers (e.g. from *on-top* there were 2700 sample from coffee cup and tray and 2700 from the turtle and tray.) Similarly, in the validation set each class was represented equally with 1404 samples per class[1]. After training, the system classified 99.95% of the 4212 validation samples correctly. Although our result is not directly comparable, we achieved a higher success rate than [11] with similar data.

We tested the system on its ability to generalise to novel objects: We introduced a 6-sided die either next to or on-top of wedge. The system performed well, classifying 95% of the samples correctly. Results are shown in Table II

III. Discussion and Related Work

The most closely related work on learning spatial relations between objects in a 3D space is [11] who use a support vector machine based approach. In this approach the support vectors are picked from for their ability to differentiate the point cloud into two objects. This has the effect that the subset of points

[1]For some classes, the object pairings were not equally represented.

TABLE I: Known objects

	True Pos.	False Pos.	True Neg.	False Neg.
Separated	1404	1	2807	0
On-top	1404	0	2808	0
Inside	1403	0	2808	1

TABLE II: Novel objects

	True Pos.	False Pos.	True Neg.	False Neg.
Separated	1421	105	1347	31
On-top	1262	30	1422	190
Inside	0	86	2807	0

considered by the classifier are on the edges of the object. Relations are then learnt based upon the relative positions of clusters of the support vectors. For any classification based approach to be successful, it requires that similar relations have a similar representation; at the level of point clouds/textlets the representation of a relation can be very different. In the case of [11], the scene is reduced to clusters with xzy coordinates. We feel that our histogram based approach allows for a more generic representation of the scene — we maintain a higher proportion of the important information about the relations between objects.

Acknowledgment

This work was supported by the EU Cognitive Systems project XPERIENCE (FP7-ICT-270273) and Leverhulme Grant F/00 152/AL.

References

[1] Frank Guerin, Dirk Kraft, and Norbert Krüger. A survey of the ontogeny of tool use: from sensorimotor experience to planning. *IEEE Transactions on Autonomous Mental Development*, 5(1):18–45, 2013.
[2] J. Piaget. *The Construction of Reality in the Child.* London: Routledge & Kegan Paul, 1937. (French version 1937, translation 1955).
[3] James J. Gibson. *The Ecological Approach To Visual Perception.* Lawrence Erlbaum Associates, 1986.
[4] E. Ugur, E. Oztop, and E. Sahin. Goal emulation and planning in perceptual space using learned affordances, 2011.
[5] Lucas Paletta and Gerald Fritz. Reinforcement learning of predictive features in affordance perception. In Erich Rome, Joachim Hertzberg, and Georg Dorffner, editors, *Towards Affordance-Based Robot Control*, volume 4760 of *Lecture Notes in Computer Science*, pages 77–90. Springer Berlin Heidelberg, 2008.
[6] Severin Fichtl, John Alexander, Dirk Kraft, Jimmy Alison Jorgensen, Norbert Krüger, and Frank Guerin. Learning object relationships which determine the outcome of actions. *Paladyn*, (Special Issue on Advances in Developmental Robotics):1 – 12, 2013.
[7] Jimmy A Joergensen, Lars-Peter Ellekilde, and Henrik 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)*, pages 1–8, June 2010.
[8] Sø renMaagaard Olesen, Simon Lyder, Dirk Kraft, Norbert Krüger, and JeppeBarsø e Jessen. Real-time extraction of surface patches with associated uncertainties by means of Kinect cameras. *Journal of Real-Time Image Processing*, pages 1–14, 2012.
[9] N. Pugeault, F. Wörgötter, and N. Krüger. Visual primitives: Local, condensed, and semantically rich visual descriptors and their applications in robotics. *International Journal of Humanoid Robotics (Special Issue on Cognitive Humanoid Vision)*, 7(3):379–405, 2010.
[10] N Pugeault and R Bowden. Spelling it out: Real-time ASL fingerspelling recognition. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1114–1119, 2011.
[11] Benjamin Rosman and Subramanian Ramamoorthy. Learning spatial relationships between objects. *Int. J. Rob. Res.*, 30(11):1328–1342, September 2011.

Learning Spatial Relationships From 3D Vision Using Histograms

Severin Fichtl, Andrew McManus, Wail Mustafa, Norbert Krüger and Frank Guerin

Abstract— **Effective robot manipulation requires a vision system which can extract features of the environment which determine what manipulation actions are possible. There is existing work in this direction under the broad banner of recognising "affordances". We are particularly interested in possibilities for action afforded by relationships among pairs of objects. For example if an object is "inside" another or "on top" of another. For this there is a need for a vision system which can recognise such relationships in a scene. We use an approach in which a vision system first segments an image, and then considers a pair of objects to determine their physical relationship. The system extracts surface patches for each object in the segmented image, and then compiles various histograms from looking at relationships between the surface patches of one object and those of the other object. From these histograms a classifier is trained to recognise the relationship between a pair of objects. Our results identify the most promising ways to construct histograms in order to permit classification of physical relationships with high accuracy. This work is important for manipulator robots who may be presented with novel scenes and must identify the salient physical relationships in order to plan manipulation activities.**

I. Introduction and Motivation

Effective robot manipulation requires a vision system which can extract features of the environment which determine what manipulation actions are possible. The preconditions of a robot's planning operators use these features of the environment to predict the successful outcome of manipulation actions. If a robot's planning operators have reasonable accuracy then they allow the robot to sequence learnt actions to achieve a goal. The use of planning precondition here is loosely related to the notion of an affordance [1], which has been well studied within the field of developmental robotics (see e.g. [2], [3]). We are particularly interested in possibilities for action afforded by relationships among pairs of objects. The physical relationship between a pair of objects is very important for manipulation, because many relationships determine the outcome of an action. For example if one object rests "on" another then pulling the lower one also causes the upper one to move. If one object is contained "inside" another, then shaking the container will not make the contained object fall out, whereas if the first object is merely "on" the second, then it will easily fall off. We need to be able to recognise these relationships in a generic way, in scenes with objects we have not been trained on. Within existing robotics work on affordances most seems to focus on single objects, although some of this work does implicitly capture to a relationship, e.g. the effect of pushing on a spherical object [2] depends on a relationship between that object and the object it is resting on (sphere on smooth surface will roll, but not on rough surface). Thus looking at relationships rather than single objects could lead to a more generic framework for affordances.

Of course it would be relatively easy for a programmer to simply code in the required relationship so that a robot would not have to learn it; however we are motivated by ideas of developmental robotics (see [4], [5] for background), and are looking for techniques which would permit a robot to learn relationships which are salient for its own actions, as it needs to. For this reason we do not wish to tailor our system for a predefined set of relationships decided by a human designer; especially we do not want to hardwire the system to use a different set of features for each relationship, where those features are hand designed to be suitable for that particular relationship. Instead we want make available one set of features which seem to be useful for multiple relationships, and then use a classifier to learn particular relationships with the generic features as input. With generic features we strengthen the possibility that the system could then learn relationships which might not have been envisaged by the human designer (although that is not explored in this paper). To this end we have looked at a number of ways of constructing object-relation data from our vision system, in order to find feature vectors which are good for multiple relations.

We consider three physical relationships: The objects may be *on-top* of, or *inside* each other, or in a position such that pulling one will cause it to contact the other and also bring it closer, as in the use of a rake; we call this *rakeable*. In addition we have negative examples of all of these. We train the system from a large set of scenes of pairs of objects in randomly generated positions, and then test its classification accuracy on novel positions and some novel objects. We have tackled part of this problem in previous work [4]. Here we report on recent improved results using a novel application of histograms to visually recognise a spatial relation between objects in the environment. Using this histogram based approach we are able to report a very high rate of success when the system is asked to recognise a spatial relation.

II. Related Work

Work on learning "affordances" is quite close to ours; Ugur et al. [2] learns affordance predictors for behaviours by learning the mapping from the object features to discovered object effect categories. These predictors can then be used by an agent to make plans to achieve desired goals. Apart from the fact that we use pairs of objects rather than single objects, this work is quite similar to ours in that essentially it boils down to classification; i.e. once effect categories have been clustered Ugur et al. use a classifier to learn the mapping from the initial object features to these effects. They use SVMs where we use random forests.

In more recent work Ugur et al. [6] use an approach somewhat close to ours in that they look at parts of objects, e.g. to recognise a handle. Identifying a part of an object based on its relationship with the main object is somewhat akin to considering it as two objects in a relationship. Ugur et al. [6] also compile histograms from low level visual features. Our histogram approach is quite different however, as it is the extension of an idea in a different work; our work is heavily inspired by the approach of Mustafa et al. [7]. That work compiles histograms over relationships between surface patches (distances and angles) in a single object. These histograms characterise the object, and are quite robust to variations in viewpoint. Mustafa et al. use this for object recognition. In our work we borrow the idea of compiling histograms over relationships among surface patches; however we look at pairs of objects, and compile histograms which relate every patch on the first object with every patch on the second. Our idea is that these histograms should characterise the relationship between the objects.

We do not feel that our work is particularly close to computer vision work in scene understanding (e.g. [8]) because those works typically recognise all objects, and then can use higher level knowledge to assist in understanding. Our work in contrast is at a lower level, and is more concerned with the physical relationships among surfaces without regard for object knowledge. We think of it more like how an infant might recognise simple physical relationships between household objects without any idea of what their names are or what their typical purposes are.

The most closely related work on learning spatial relations between objects in a 3D space is [9] who use a support vector machine based approach. In this approach the support vectors are picked from for their ability to differentiate the point cloud into two objects. This has the effect that the subset of points considered by the classifier are on the edges of the object. Relations are then learnt based upon the relative positions of clusters of the support vectors.

For any classification based approach to be successful, it requires that similar relations have a similar representation; at the level of point clouds/texlets the representation of a relation can be very different. In the case of [9], the scene is reduced to clusters with xyz coordinates. We feel that our histogram based approach allows for a more generic representation of the scene — we maintain a higher proportion of the important information about the relations between objects.

We can also relate our work to infant development. In the period from six months of age through to two years human infants undergo significant development in their skills and understanding relating to physical world objects and their manipulation. Observations of infants show that, at any particular age, they possess a repertoire of behaviours or manual skills which they apply to various objects or surfaces they encounter [10], [11]. Each such behaviour could be seen as roughly analogous to a planning operators in Artificial Intelligence, because there are situations which make them likely to be executed (like the precondition of a planning operator), and expected effects (postcondition), as well as some motor control program describing the behaviour executed. As infants develop they solve the problems of (i) identifying when a new behaviour should be created, (ii) learning the new precondition, (iii) postcondition, and (iv) motor program for the new behaviour. In this paper, we focus on learning the precondition for a new behaviour. This is a particularly interesting problem in the case of means-ends behaviours (i.e. where one action is used in order to facilitate another [12]), because it is through learning means-ends behaviours that infants begin to learn about relationships between objects [13]. The precondition must capture the relationship between objects which determines where the behaviour works or does not work. In preconditions the infant is learning new important abstractions over its sensor space. This can change how an infant understands a scene because the infant can begin to see things at a higher level of abstraction, seeing precisely those relationships which are important in determining what object manipulations are possible (by itself or other agents).

III. Method

A. Overview

Learning a spatial relationship from raw sensor data is challenging as it may take many thousands of examples to train an effective classifier. For this reason we first perform an abstraction using histograms to convert data into a form which simplifies the learning procedure.

After suitable abstractions have been found, to learn classification models for the different *Spatial Relations* we use the Random Forest[14] algorithm. Random Forests (RF) are particularly well suited for our use case, as they inherently do feature selection and hence identify the relevant features amongst large amount of the input variables.

B. Data Collection

In this work, for learning spatial relations between objects, we collected data using a physically realistic simulation environment designed for Robot Simulations and a sophisticated vision system using a simulated Kinect camera (See III-C). This Simulator, called RobWorkSim[15], is developed and maintained by the Robotics department of the Maersk McKinney Moller Institute at the University of Southern Denmark.

The objects used range from simple to complex household items and incorporate *supporting* objects like trays, *container* objects like soup bowls and other objects like plastic toys. Figure 1 depicts the objects used in the experiments.

Using the Simulator, we (randomly) placed pairs of objects on a surface and labeled their spatial Relation, before extracting a relational histogram based representation of the scene using the CoViS vision system (See III-C). Figure 2 illustrates different *Spatial Relations* we focussed on in this work.

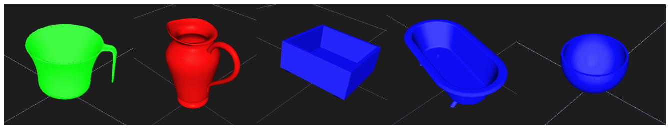


Fig. 1a. Container Objects

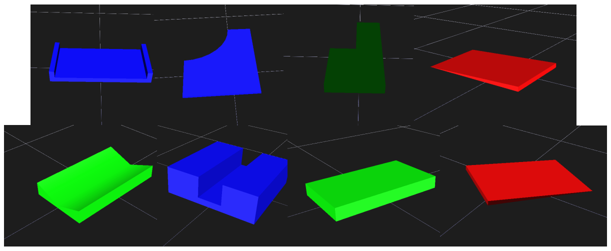


Fig. 1b. Support Objects

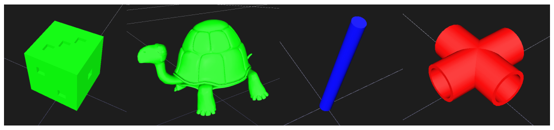


Fig. 1c. Other Objects

Fig. 1: Overview of all Objects used in this work

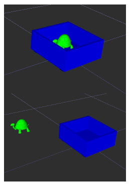


Fig. 2a. Inside / not inside

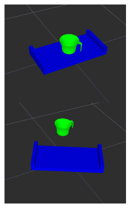


Fig. 2b. On top / not on top

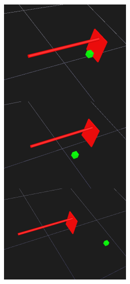


Fig. 2c. Rakeable / almost rakeable / not rakeable

Fig. 2: Overview of the three relationships used in this work. 2a illustrates the "Inside" case with a turtle toy either inside a box or not, 2b similarly illustrates the "On-top" case and 2c depicts from top to bottom Rake catching a die, Rake close to catching a die and Rake not near the die.

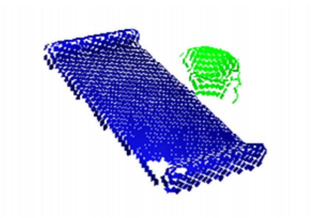


Fig. 3: Texlet representation of a scene. Note that some texlets (surface patches) are missing where surfaces appeared too bright.

For collecting the data, using the RobWork simulator, we randomly distributed one object on a workspace using a normal distribution around the centre of the camera view. For the second object we followed different strategies. This was necessary to promote the chance of certain Relations amongst the objects as with complete uncorrelated random positioning the likelihood of "Inside" and "On-top" scenarios occurring by uninformed random placement is too low to make a reasonable sized training set. In order to promote “desired” cases to collect the necessary data, placed the 2nd object around the first object with a normal distribution and a smaller variance. In a few cases we limited the possible positions of the second object to “Inside” or “On-top” relations. The orientation of objects in the 3D space was uniformly distributed over roll, pitch and yaw, with the only limitation to ensure *Support, Container* objects stand upright. The *Rakes* had a further restriction to be always in the same orientation perpendicular to the camera, to ensure the 2nd object is visible to the camera and not obscured by the rake itself. Other objects had no limitations to their orientations.

C. *Vision System*

In our work we use a kinect based vision system called CoViS[16], [17], which is developed at the Cognitive Vision Lab at the Maersk McKinney Moller Institute at the University of Southern Denmark.

1) Visual Representation: Using the picture of the scene and the depth map, both provided by the *Kinect*[18] camera, our vision system calculates a 3D point cloud as it is common amongst state of the art vision systems[19].

Based on this 3D point cloud and the colour information of the scene, our vision system creates surface patches as shown in Figure 3. There are different layers of surface patches. We only use the basic layer with surface patches which we call texlets. These texlets describe the surface of the scene with additional information, e.g. not only position in the space, but also the orientation and colour of the surface [20].

In our simulator, we also simulate the noise of real Kinect devices. This gives us data about the depth to the objects in our 3D scene just as we would have obtained from a real Kinect looking at a real scene with 3D objects. The data from the simulated vision system is hence more noisy and less accurate than the perfectly accurate data which could be provided the simulator. The noise affects the position and orientation of the texlets. In addition some texlets can be missing, dependent on the placement of light sources in the simulator, as happens in a real scene; see Figure 3.

2) Object Segmentation: We acknowledge that highly sophisticated Object Segmentation algorithms exist and we assume they could be employed to work in a more complex environment. In this work, however, we used a trivial method for Object Segmentation. The method we present here is based on colour information of the Texlet “Cloud”.

For this simple method to work, it is assumed that the objects are coloured in one of a known set of colours. This

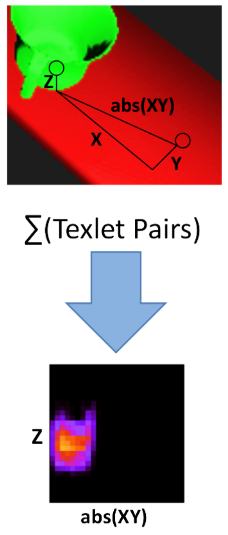


Fig. 4: Illustration of the Histogram creation process from texlets to histograms.

is a very strong assumption, but it could be relaxed by using more sophisticated segmentation methods, which could take into consideration factors like discontinuities of surface curvatures and colour differences. We simplified our segmentation problem because in this work we wanted to focus on finding which type data from the pairs of objects would be most useful for characterising relationships, without our results being affected by errors in segmentation. Future work could look at how the segmentation problem interacts with the problem of recognising spatial relationships.

We coloured our objects either Red, Blue or Green and the background was Grey. The Texlets where then grouped based on their colour or where neglected if they were Grey (or any colour other than Red, Green or Blue).

After segmentation, each object is assigned its unique set of texlets. A texlet representation of a scene with two segmented objects can be seen in Figure 3.

3) Relational-Histogram Creation: Using the segmented texlet based scene representation, we create *Relational Histograms* to capture the spatial relations between objects. These *Relational Histograms* form a relational space into which the absolute geometric information (3D position and orientation) of the 3D texlets is transferred. To achieve this transfer, we define a set of relational features which encode the spatial relationship structure of the objects in the scene.

More specifically, for each scene we have 2 texlet groups Π^1 and Π^2 representing the segmented objects 1 and 2 in the scene. For each cross object texlet pair of the form $\Pi_i^1 \oplus \Pi_j^2$ we calculate four *Euclidean Distances* $R_d(\Pi_i^1, \Pi_j^2)$ (The Euclidean Distances along the X, Y and Z axes respectively and in the XY plane) and three *Angle Relations* $R_a(\Pi_i^1, \Pi_j^2)$ (The line through the two texlets is projected onto one of the planes XY, XZ, or YZ, and we look at the angle between the projected line and the axes X, Z and Z respectively).

The size of these feature vectors, describing the relation between the two objects in the scene, is variable and determined by the amount of texlets extracted by the vision system. As we want to apply *Supervised Learning Algorithms*, we need the input vector to be generic and of fixed length, for all possible scenarios.

For this, instead of using the data vectors $R_d(\Pi_i^1, \Pi_j^2)$ and $R_a(\Pi_i^1, \Pi_j^2)$ directly, we compute 1-, 2- or 3 Dimensional *"Relational" Histograms* from the data vectors and use these as learning data input, similar to Mustafa et al.[7].

4) Histogram Types: In this work we experimented with 4 different kinds of Histograms for learning *Spatial Relations*. Two 1D composites of Relational Histograms, one 2D and one 3D Histogram. The two 1D composites of Histograms are combinations of 1D Histograms.

- 1D Histograms capture simple relational features between inter-object texlet pairs. For the first 1D composite Relational Histogram, we calculate 3 1D histograms capturing the distances between texlets along each of the 3 main axes X, Y and Z respectively and put them together as 1D learning input. For the second 1D composite Relational Histogram we compute the angle relations in the 3 planes in the space opened by the 3 main axes (XY, XZ and YZ planes) and calculate the angles between texlets in these planes as described above. These angle relations put together alongside the 3 distance histograms, make up the second 1D Relational Histogram.
- The 2D Histogram used in this work, captures the absolute distance of inter-object texlet pairs in the XY plane and puts it into relation with the height difference of the two texlets (i.e. Z difference). In Figure 4 a plain 2D histogram is illustrated.
- The 3D Relational Histogram captures distances between texlets amongst three Dimensions, in a similar fashion as the 2D Histogram does for two Dimensions. For the 3D Histogram, however, we used the actual position differences amongst all three main axes (X, Y and Z). 3D Histograms have not been graphically illustrated in this paper mainly because they did not give particularly good results, so it was less interesting to inspect them visually.

In Fig. 5 we show examples of all 1D and 2D histograms. 3D histograms have not been illustrated here, but the results are presented in Sec. IV. Note that the first (leftmost) figure in each row is all that the camera sees, so that sometimes some part of an object might be missing if it is very close, or sometimes objects might be far away. Note also that texlets also come only from seen parts; this explains why the first 1D angle histogram shows a peak at both 0 and 360 degrees; it is because the parts of the pitcher close to the camera dominate (the vector from object to camera is 0 or 360 degrees; from the object to the right is 90 degrees, and 180 points to the back). Thus even though the pitcher totally surrounds the rod, the system does not see most of the texlets surrounding it. Note that our system is different to Mustafa et al.[7] in this respect because they use 3 cameras, surrounding the object.

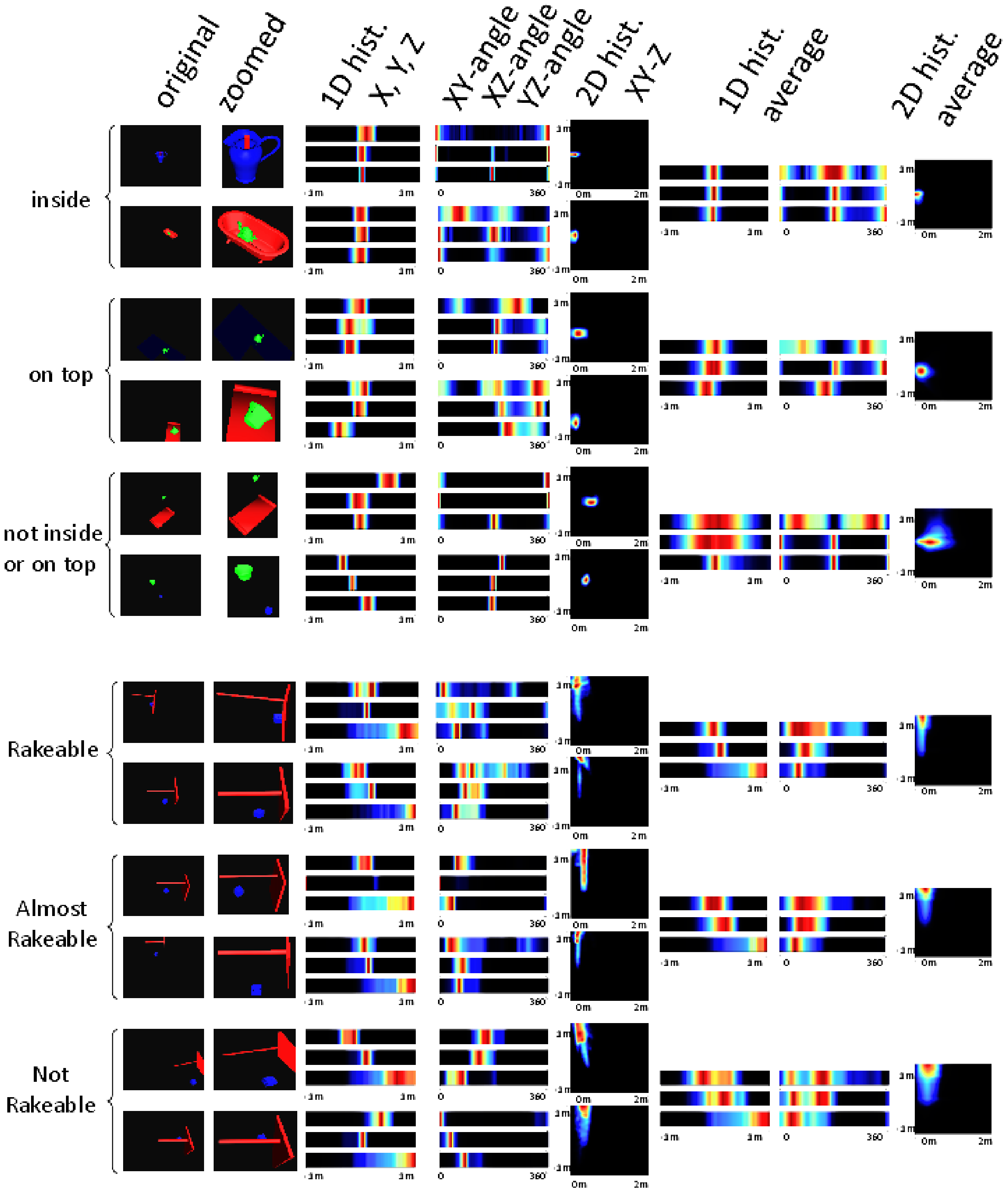


Fig. 5: This Figure is provided to illustrate comparatively how well 1D and 2D histograms can discriminate between the various relationships we have experimented with. For each row of this figure, from left to right we have: an illustration of a scene with a pair of objects, the 1D histograms of that scene, the 2D histograms of that scene, the 1D average histogram across all training samples, the 2D average histogram across all training samples.

D. Data & Histogram Post Processing

Mustafa et al.[7] have demonstrated the potential of histograms for object recognition. However, we found the lack of generalisation capabilities of Random Forests to be a limitation in their applicability when it comes to learning Spatial Relations, as it is of major importance to be able, to not only recognise relations between known objects, but also for never before seen objects. In this we differ from Mustafa et al. who only want to recognise well known objects. Therefore in order to not only increase the learning performance, but especially increase the robustness of recognising spatial relations among novel objects, we implemented some feature vector and histogram post processing methods.

The efficiency of these post processing methods on the spatial recognition rate and robustness was investigated in prior work [21]. Given the clear improvements shown there, we use these methods for all learning in this paper.

1) Histogram Normalisation:

Histogram Normalisation proved to vastly increase the robustness of the recognition rate when it comes to novel object pairs and their relations [21]. This is not surprising as the numbers in the un-normalised histograms rely heavily on the sizes of the objects, and the amount of texlet extracted for them by the vision system. Hence, two large objects would generate bigger numbers than two small objects in the same relation. The according histograms would hence look very similar but with different scales. Normalising these histograms removes these scaling effects caused by the sizes of the objects.

E.g. the two imaginary histograms $[1|2|4|1]$ and $[2|4|8|2]$ could describe the same relations for objects pairs with different sized objects. Normalisation would bring both histograms down to $[0.25|0.5|1|0.25]$ and hence remove the differences caused by the object sizes, allowing them to be recognised as the same *Spatial Relation*.

2) Histogram Smoothing:

Histogram Smoothing using normal Gaussian smoothing considering only direct neighbours (i.e. Window size 3) was also found to increase performance, but with a smaller effect on the robustness in case of novel objects [21]. For Smoothing we applied a standard Gaussian Smoothing algorithm with a variance $\sigma^2 = 1$ and a window size of 3 bins, i.e. only direct neighbours to values are taken into account for the smoothing.

Smoothing was found especially useful when used on 2D and 3D Histograms as these are naturally quite sparse, also compared to the according 1D histograms. The smoothing accounts for noise in the histograms caused by kinect camera and the limits in its resolution.

3) Data Scaling:

We apply *Logarithmic Scaling* to the feature vectors preceding the creation of Histograms. This logarithmic scaling had biggest impact on general classification performance [21] but was only applied on distance features; the angle relation features were not scaled as this would not be sensible.

To scale the data, we replaced the original values of the feature vectors, i.e. distances, with $f(x) = ln(x + 1)$. To compensate for the fact that the logarithm is not defined for negative values, we applied the logarithm on the absolute value and used the negative of the result for originally negative values. Adding 1 to the each absolute values before taking the logarithm ensures that the return value is always positive and the values do not overlap for positive and negative values. This logarithmic scaling has the effect that in the histograms created from the scaled feature vectors, for small distances there is a higher resolution than for larger distances. This has a positive effect because in the smaller distances lies the most useful information about *Spatial Relationships*. It is evident, that if the distance between inter-object texlet pairs is large, the two objects are unlikely to be in a "On-top" or "Inside" relation, but instead are unrelated distributed in the scene.

The Logarithmic Scaling leads to distortion effects of the histograms which makes them slightly less intuitive as can be seen when comparing the 2D histograms of figure 5 with the 2D histogram at the bottom of figure 4. They show the same features but the latter 2D histogram is not logarithmic scaled.

IV. Results

To test the performance of the different Histograms and the robustness when it comes to Novel objects we use two different test sets. For every scenario we have a Validation set. This Validation set contains the same object pairs as the Training set, but different instances; i.e. of the overall set of available cases of each object pair setup, some were put into the training set, some others into the Validation set. Furthermore, for the “Inside” and “On-top” scenarios, we also have kept some object pairs out of the Training and Validation sets, to test the performance on not before seen object pairs, to verify the robustness of the Histograms when it comes to novel objects.

For the “Inside” and “On-top” Relations, we considered “Inside” as a subgroup of “On-top”. Any “Inside” case is hence also considered to be “On-top” but not necessarily the other way round.

For the “Inside” Relation we have 9738, 11799 and 5381 instances in the Training, Validation and Novel sets respectively. For the “On-top” Relation we have 11348, 14551 and 5616 instances in the Training, Validation and Novel sets respectively, on-top of the “Inside” instances. We also have 11008, 12943 and 5621 instances in the respective sets of the relationship free instances (neither “inside” nor “On-top” samples).

For the Rake Relation sets, we have 1750 and 4087 samples in the Training and Validation sets respectively for each of the 3 classes “Rakeable”, “Almost Rakeable” and “Not Rakeable”.

In the following we call the trained classifiers *XYZ* for the classifier based on the 1D Histograms without angle information. *XYZABC* identifies the 1D Histogram based classifier with the angle features. The 2D Histogram based classifier we call *XYabs_Z* and the 3D Histogram *X_Y_Z*.

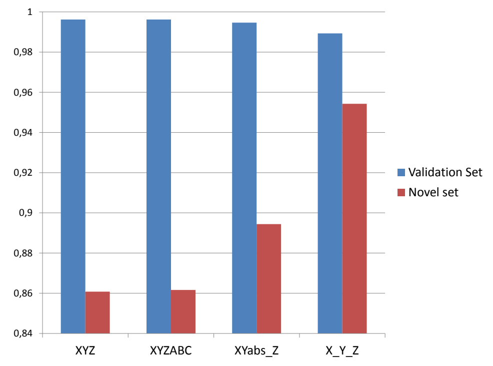


Fig. 6: Performance of "Inside" classifier on the Validation and the Novel test sets.

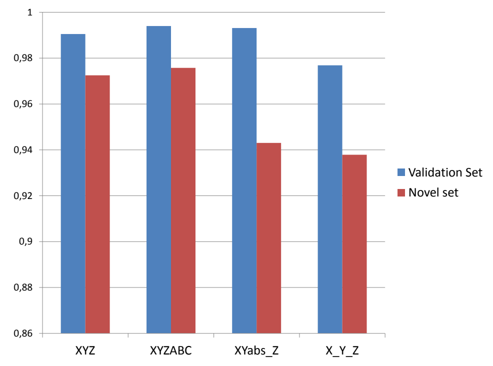


Fig. 7: Performance of "On-top" classifier on the Validation and the Novel test sets.

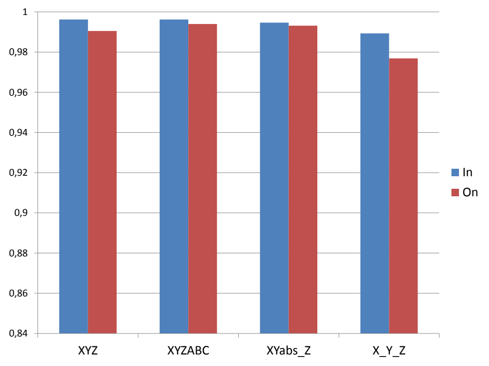


Fig. 8: Comparison of the the "Inside" and "On-top" classifiers performance on the Validation set.

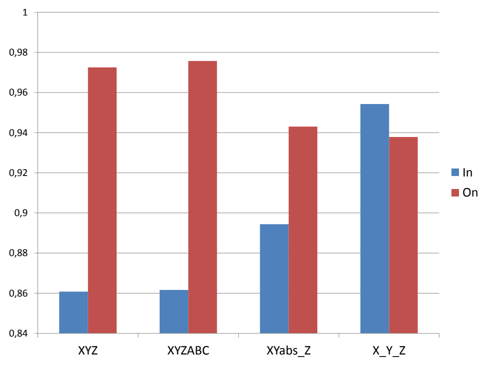


Fig. 9: Comparison of the the "Inside" and "On-top" classifiers performance on the Novel set.

For each Relationship we trained four individual Binary classifiers, one per Histogram type. Each Binary Classifier was trained and tested 10 times on the Validation and Novel sets where applicable. The results presented here are always the averages of these 10 runs.

A. *Inside and On-top Relations*

In Figure 6 we show the performance of the four different classifiers for the "Inside" case. The Blue bars show the performance on the Validation set, the red bars show the performance on the Novel set.

Figure 7 shows the same graph as Figure 6, but for the "On-top" case.

Figures 8 and 9 directly compare the four classifiers of "Inside" and "On-top" on the "Validation" and "Novel" test sets respectively.

B. *Rake Relations*

Figure 8 compares the classifier performances of the different Relations and of the different classifiers on each Relation at the same time.

V. Discussion

Overall the best histograms for classification purposes seem to be the 1D histograms including angles. Their advantage is marginal for "inside" or "on top", but more pronounced for the rake. The 3D histogram performs worst, and this is likely due to the sparsity of data in a 3D histogram.

Above we have presented an empirical evaluation of how well our various histograms can characterise a relationship between two objects. We can also do a thought-experiment type of analysis, somewhat akin to a mathematical proof where we try to construct a counterexample for our classifiers. E.g. for the relationship "inside" we can contrive objects which would be misclassified by our classifier. For a false positive we can think of how to "fool" the 1D histogram of angles in the XZ plane for example. This histogram has large values for angles which are at the same Z value as the contained object, but offset from it in the $Y-$axis; these are surface patches of the container bounding the contained object. There are two weaknesses: (i) the orientation of the surface patch on the container is not considered - so a sort of louvered surface full of gaps would be admissible; (ii) the Y of the surface patch on the container is not considered, so a large hole at small Y values could be mitigated by surface patches at larger Y values, leading to a container with a missing side being a false positive. Similar examples can be contrived for other histograms. It is harder to contrive a false negative, meaning that we have a "weak" notion of container, because it admits a large set of objects, even with gaps.

Clearly we could upgrade the histograms with a histogram which looks at Y values in conjunction with the XZ angles,

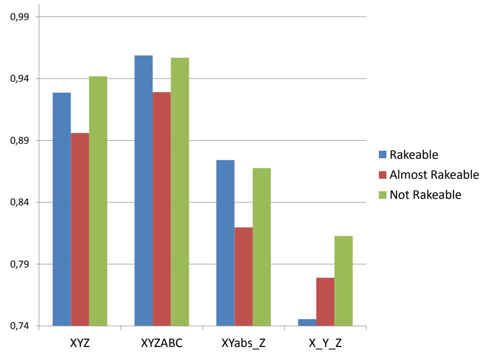


Fig. 10: Comparison of the four classifiers on the Rake relations.

however we are wary of constructing features which are specifically tailored to the recognition of one particular relationship ("inside" in this case), because of our desire to allow the system to have generic features so it could learn new relationships which the designer might not have foreseen the need for.

One major weakness of the system is that is makes no effort to guess at unseen parts of objects. This is probably why the results for "inside" are worse than "on top" for novel objects. We suspect that a human recognising relationships such as the stick in the pitcher at the top of Fig. 5 would complete absent texlets based on object knowledge and gestalt principles, and so "see" a rod completely surrounded by texlets.

VI. Future Work

In future work we plan to repeat the experiments here with a set of real objects, and real Kinect cameras. We may also experiment with objects which are not simply coloured so that we need to tackle a more realistic segmentation problem. In addition we plan to test our classifiers on the training set of objects used by [9], in order to have a direct comparison which would permit us to consider the relative strengths and weaknesses of the two approaches.

Acknowledgment

This work was supported by the EU Cognitive Systems project XPERIENCE (FP7-ICT-270273).

This work was performed using the Maxwell High Performance Computing Cluster of the University of Aberdeen IT Service (www.abdn.ac.uk/staffnet/research/hpc.php), provided by Dell Inc. and supported by Alces Software.

We thank Nicolas Pugeault for help with his random forests library.

References

[1] James J. Gibson. *The Ecological Approach To Visual Perception*. Lawrence Erlbaum Associates, 1986.

[2] E. Ugur, E. Oztop, and E. Sahin. Goal emulation and planning in perceptual space using learned affordances, 2011.

[3] Lucas Paletta and Gerald Fritz. Reinforcement learning of predictive features in affordance perception. In Erich Rome, Joachim Hertzberg, and Georg Dorffner, editors, *Towards Affordance-Based Robot Control*, volume 4760 of *Lecture Notes in Computer Science*, pages 77–90. Springer Berlin Heidelberg, 2008.

[4] Severin Fichtl, John Alexander, Dirk Kraft, Jimmy Alison Jorgensen, Norbert Krüger, and Frank Guerin. Learning object relationships which determine the outcome of actions. *Paladyn*, (Special Issue on Advances in Developmental Robotics):1 – 12, 2013.

[5] Frank Guerin, Dirk Kraft, and Norbert Krüger. A survey of the ontogeny of tool use: from sensorimotor experience to planning. *IEEE Transactions on Autonomous Mental Development*, 5(1):18–45, 2013.

[6] E. Ugur, H. Celikkanat, E. Sahin, Y. Nagai, and E. Oztop. Learning to grasp with parental scaffolding. In *IEEE Intl. Conf. on Humanoid Robotics, Bled, Slovenia, October*, pages 480–486, 2011.

[7] Wail Mustafa, Nicolas Pugeault, and N Krüger. Multi-View Object Recognition using View-Point Invariant Shape Relations and Appearance Information. In *ICRA 2013*, 2013.

[8] W. Choi, Y.-W. Chao, C. Pantofaru, and S. Savarese. Understanding indoor scenes using 3d geometric phrases. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[9] Benjamin Rosman and Subramanian Ramamoorthy. Learning spatial relationships between objects. *The International Journal of Robotics Research*, 30(11):1328–1342, 2011.

[10] J. Piaget. *The Origins of Intelligence in Children*. London: Routledge & Kegan Paul, 1936. (French version 1936, translation 1952).

[11] Jeffrey J. Lockman. A perception-action perspective on tool use development. *Child Development*, 71(1):137–144, 2000.

[12] P. Willatts. Development of problem-solving strategies in infancy. In D.F. Bjorklund, editor, *Children's Strategies: Contemporary Views of Cognitive Development*, pages 23–66. Lawrence Erlbaum, 1990.

[13] J. Piaget. *The Construction of Reality in the Child*. London: Routledge & Kegan Paul, 1937. (French version 1937, translation 1955).

[14] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[15] Jimmy A Joergensen, Lars-Peter Ellekilde, and Henrik 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)*, pages 1–8, June 2010.

[16] Norbert Krüger, Nicolas Pugeault, and Florentin Wörgötter. Visual primitives: local, condensed, semantically rich visual descriptors and their applications in robotics. *International Journal of Humanoid Robotics*, 07(03):379–405, 2010.

[17] Sø renMaagaard Olesen, Simon Lyder, Dirk Kraft, Norbert Krüger, and JeppeBarsø e Jessen. Real-time extraction of surface patches with associated uncertainties by means of Kinect cameras. *Journal of Real-Time Image Processing*, pages 1–14, 2012.

[18] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors*, 12(2):1437–1454, 2012.

[19] R B Rusu and S Cousins. 3D is here: Point Cloud Library (PCL). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4, May 2011.

[20] N. Pugeault, F. Wörgötter, and N. Krüger. Visual primitives: Local, condensed, and semantically rich visual descriptors and their applications in robotics. *International Journal of Humanoid Robotics (Special Issue on Cognitive Humanoid Vision)*, 7(3):379–405, 2010.

[21] Andrew McManus. Learning Spatial Relationships. *Master Thesis*, 2013.

2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Action-Grounded Push Affordance Bootstrapping of Unknown Objects*

Barry Ridge and Aleš Ude†

Abstract— **When it comes to learning how to manipulate objects from experience with minimal prior knowledge, robots encounter significant challenges. When the objects are unknown to the robot, the lack of prior object models demands a robust feature descriptor such that the robot can reliably compare objects and the effects of their manipulation. In this paper, using an experimental platform that gathers 3-D data from the Kinect RGB-D sensor, as well as push action trajectories from a tracking system, we address these issues using an action-grounded 3-D feature descriptor. Rather than using pose-invariant visual features, as is often the case with object recognition, we ground the features of objects with respect to their manipulation, that is, by using shape features that describe the surface of an object relative to the push contact point and direction. Using this setup, object push affordance learning trials are performed by a human and both pre-push and post-push object features are gathered, as well as push action trajectories. A self-supervised multi-view online learning algorithm is employed to bootstrap both the discovery of affordance classes in the post-push view, as well as a discriminative model for predicting them in the pre-push view. Experimental results demonstrate the effectiveness of self-supervised class discovery, class prediction and feature relevance determination on a collection of unknown objects.**

I. INTRODUCTION

Endowing an autonomous robot with both the ability to learn about object affordances [1] from experience and the ability to use these learned affordances to make useful predictions and manipulations in its environment is no easy task, and simplifying assumptions are often made in order to make the problem more soluble. For example, in the case of object push affordance learning [2]–[6], 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 or their affordances, such techniques may not be as feasible, and while this increases the complexity of the learning problem, it is an important research approach from a cognitive and developmental robotics perspective.

In this paper, we explore object push affordance learning by gathering data from human object push experimental trials (see Fig. 1). Objects on a table surface were pushed by a human, whose hand motion trajectories were tracked while 3-D point clouds of the objects before and after interaction 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 goal was to extend a type of bootstrap learning whereby significant clusters are discovered in features extracted from the post-push point clouds that are used as affordance classes in order to train an affordance classifier using features extracted from the pre-push point clouds as input in a developmental multi-view online learning process. Note that a similar learning process could be realised on an autonomous robot. We use the above setup to ease the process of data gathering.

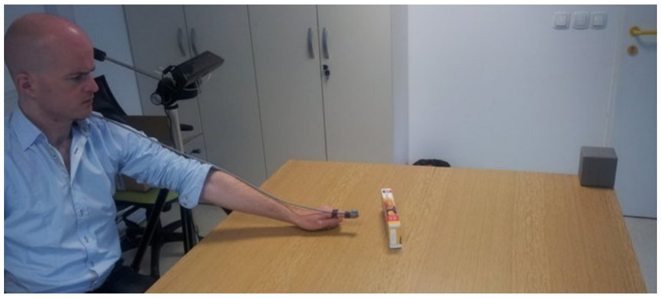


Fig. 1. Our setup for human object push affordance data gathering.

An idea for grounding the affordance learning task in the pushing actions informed our research. We argue that the approach of visual object recognition followed by object manipulation informed by a prior object model (see e.g. [6]). is, although quite useful when the main focus is accurate prediction, perhaps less important when the main focus is learning from experience. Instead, here we favour an approach where little or no prior information on the structure of the objects being pushed is assumed. To this end, we propose a features-based approach where, rather than using pose-invariant visual features, as is commonly the case with object recognition, we ground the visual features of objects with respect to their manipulation, that is, by using shape features that describe the surface of an object relative to the push contact point and direction.

A. Related Work

Past work on object affordance learning with robots has seen a varied succession of approaches, ranging from learning affordances for particular objects [2], to supervised discriminative learning of pre-defined affordance classes from object features [7], to unsupervised discovery and subsequent

*This research has been supported by EU FP7 projects Xperience (FP7-ICT-270273) and IntellAct (FP7-ICT-269959).

†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

discriminative learning of affordance classes [5], [8]–[11] to probabilistic frameworks [3], [6], and others. One of the earliest works in the literature to deal with affordance learning in a robot was by Fitzpatrick *et al.* [2]. The authors trained a humanoid robot to recognize rolling affordances of four household objects using a fixed set of actions to poke the objects in different directions as well as simple visual descriptors for object recognition.

Paletta *et al.* [7], [12] developed a mobile robotic platform equipped with a crane featuring a magnetic end-effector which was used to pick up metallic objects in its surroundings. Their affordance learning system used decision trees and reinforcement learning of predictive features (SIFT descriptors) to distinguish between two affordance classes of liftable and non-liftable metallic objects. Ugur *et al.* [11] worked with a robotic system consisting of a range scanner and a robotic arm that learned affordances of objects in a table-top setting using an unsupervised two-step approach of effect class discovery and discriminative learning for class prediction. More recently they have applied similar techniques to a scenario involving self-discovery of motor primitives and learning grasp affordances [13].

In [3], the authors used a humanoid robot to push objects on a table and used a Bayesian network to form associations between actions, objects and effects. In the work of Omrčen *et al.* [4], the robot first observes how an object moves when pushed in a certain direction. The collected data are used as input to a neural network which learns to predict the motion of pushed objects. Kopicki *et al.* [6] used a probabilistic framework to address prediction of rigid body transformations in an object pushing scenario both in simulation and using a real robotic system. Their work was similar to ours here in that it explicitly addressed the representation of object parts as well as the combination of knowledge from multiple models. However, their visual system relied on the use of prior object models for object localisation, something we explicitly avoid in this work.

Object shape features have been used in prior work on push affordance learning [5], [6], [11], [13], but grounding object shape features relative to pushing actions has not been studied as extensively. Recent work by Hermans *et al.* [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. In this paper, we develop a similar idea employing full 3-D shape features.

The remainder of the paper is structured as follows. In the following section, we describe how the action-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 III, we describe our learning approach. In Section IV we describe our experiments and results. Finally, in Section V, we conclude.

II. GROUNDING 3-D SHAPE FEATURES IN PUSH ACTIONS

In our experimental setup for human object push data gathering, shown in Figure 1, 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 1. 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 [15] 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 [16].

B. Action coordinate frame transformation

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 perpendicularly to both of them such that a left-handed coordinate system is formed. 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

[1]http://pointclouds.org

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 both the pre-push and post-push object point clouds as well as the push trajectory to the action frame as defined by the contact point and the fitted line.

C. *Action-grounded shape features*

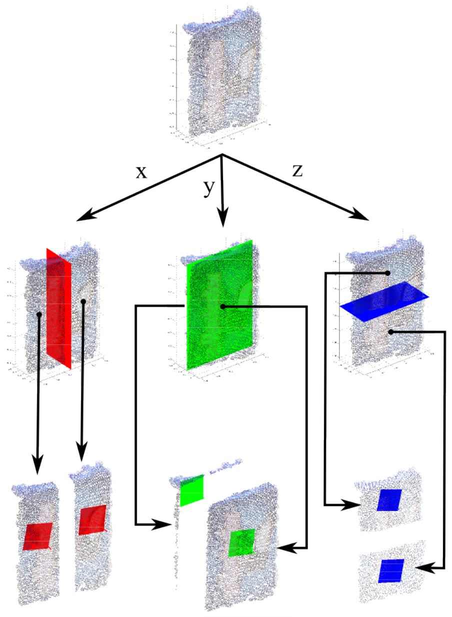


Fig. 2. Partitioning a sample object point cloud into sub-parts. Top row: original pre-push object point cloud. Middle row: partitioning planes divide the point cloud evenly in each dimension to create sub-parts. Bottom row: planes are fitted to each sub-part for feature extraction.

With both pre-push and post-push object point clouds now grounded in the action coordinate frame, we turn to generating a feature 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 the x and y components of the plane normal as features. We discard the z component to reduce the feature space dimensionality, the x and y components being sufficient to quantify the angle of the plane from the normal. Examples of these features being extracted from different point clouds are shown in Figure 5.

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 before $\{O^1, \ldots, O^{35}\}$ and after $\{E^1, \ldots, E^{35}\}$ the push interaction:

- global object point cloud features.
- x-axis division, left part features.
- x-axis division, right part features.
- y-axis division, front part features.
- y-axis division, back part features.
- z-axis division, top part features.
- z-axis division, bottom part features.

III. BOOTSTRAPPING OBJECT PUSH AFFORDANCES

To enable the type of bootstrap learning discussed in the introduction, we frame our scenario as a multi-view online learning problem. Multi-view learning [17], as well as the related fields of cross-modal and multi-modal learning, [18]–[21], are machine learning areas which are concerned with the problem of learning from data represented by multiple distinct feature sets in different data views or modalities. Given that common theme, the learning objective may otherwise differ depending on the particular context [17]. In our scenario, object pre-push shape features $\mathbf{x}_i = [O_i^1, \ldots, O_i^{35}]^T$ define the feature space in one data view, the input space, whereas object post-push shape features $\mathbf{y}_i = [E_i^1, \ldots, E_i^{35}]^T$ define the feature space in another view, the output space. Our learning goal is to find significant clusters amongst the $\mathbf{y}_i$ feature vectors in the output space, then use these clusters as classes to train a classifier using the $\mathbf{x}_i$ feature vectors in the input space, that is to find a mapping $f : \mathbb{R}^n \to \mathbb{N}$ from input space feature vectors to class labels representing affordances grounded in output space feature clusters. We considered this as a multi-view learning problem since there is a natural separation between the two feature spaces under consideration, which model potential causes and potential effects respectively, and we wished to use information in the output view (effect) to influence learning in the input view (cause).

With that in mind, we extended the self-supervised multi-view online learning algorithm originally proposed in [5]. This algorithm is self-supervised in the sense that the data distribution in the output space coupled with the co-occurrence information, is used to form a supervision signal that directs learning in the input space. The algorithm achieves something similar to methods for self-discovery and prediction of affordance classes proposed in other work [13], but can be trained online and makes use of the class information for discriminative learning as it emerges dynamically during training. In the following, we summarise the relevant parts of the algorithm. Further detail can be found in [5].

A. Self-supervised Multi-view Online Learning

Assuming there are two datasets of co-occurring data, $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^m \mid i = 1, \ldots, D\}$ in the input space and $\mathcal{Y} = \{\mathbf{y}_i \in \mathbb{R}^n \mid i = 1, \ldots, D\}$ in the output space, where we work under the assumption that the $\mathbf{x}_i$ and $\mathbf{y}_i$ data vectors are not all available at once and arrive in an online data stream, we aim to represent each space via vector quantization [22] using *codebooks of prototype vectors* $\mathcal{W} = \{\mathbf{w}_i \in \mathbb{R}^m \mid i = 1, \ldots, M\}$ for the input space and $\mathcal{V} = \{\mathbf{v}_i \in \mathbb{R}^n \mid i = 1, \ldots, N\}$ for the output space respectively, approximating the data distributions in each view. We implement multi-view connectivity between the two codebooks via a weight matrix which we refer to as *co-occurrence mapping*, defined as follows:

$$\mathcal{H}(\mathcal{W}, \mathcal{V}) = \begin{bmatrix} \gamma_{1,1} & \gamma_{1,2} & \cdots & \gamma_{1,\mathrm{N}} \\ \gamma_{2,1} & \gamma_{2,2} & \cdots & \gamma_{2,\mathrm{N}} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{\mathrm{M},1} & \gamma_{\mathrm{M},2} & \cdots & \gamma_{\mathrm{M},\mathrm{N}} \end{bmatrix} \tag{1}$$

where the $\gamma_{i\,j}$ are weights that are used to record the level of data co-occurrence between nearest-neighbour prototypes in each of the codebooks and are adjusted by applying the Hebbian rule to cross-view prototype activations. We aim to find significant clusters of prototypes in the output space which we dub *class clusters* that we treat as classes to be used for discriminative learning in the input space. Codebook training within the input space uses two learning phases. The first phase involves unsupervised clustering of the prototypes in each data view using the self-organizing map (SOM) algorithm [22] such that the data distributions are roughly approximated. The second phase involves self-supervised discriminative learning using a modified form of learning vector quantization (LVQ) [5] such that the positions of the prototypes in the input space are refined for classification purposes using cross-view co-occurrence information. Throughout, the nearest-neighbour rule is employed using a weighted squared Euclidean distance,

$$d^2(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{n} \lambda_i (x_i - w_i)^2, \tag{2}$$

where x_i and w_i are feature components of $\mathbf{x}$ and $\mathbf{w}$ respectively, and the λ_i are weighting factors for each feature which facilitate feature relevance determination as described later in Section III-E.

The co-occurrence mapping can be used to infer the relationship between the prototypes in the different feature spaces by projecting the weights for a given prototype in one space onto the codebook in the other space. Given prototype $\mathbf{w}^i \in \mathcal{W}$ we define

$$P(\mathbf{v}_j | \mathbf{w}_i) = \frac{\gamma_{i\,j}}{\sum_j^N \gamma_{i\,j}} \tag{3}$$

which describes the probability of prototype $\mathbf{v}_j \in \mathcal{V}$ matching prototype $\mathbf{w}_i \in \mathcal{W}$ based on the co-occurrence mapping, where $\gamma_{i\,j}$ is the connection weight from (1) between prototypes $\mathbf{w}_i$ and $\mathbf{v}_j$. Thus for a given $\mathbf{w}^i \in \mathcal{W}$, making such projections for all $\mathbf{v}^i \in \mathcal{V}$ forms a spatial probability distribution over codebook $\mathcal{V}$ and is a useful tool that allows us to measure for measuring how one data view looks from the perspective of another in terms of past co-occurrences of data. A visualisation is provided on the left side of Fig. 3.

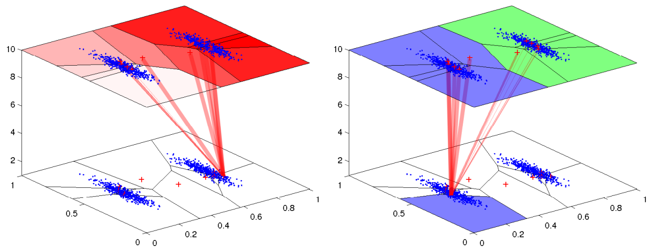


Fig. 3. Multi-view classifier construction. Left figure: Cross-view co-occurrence weight projection. The weights of the $\mathcal{H}(\mathcal{W}, \mathcal{V})$ mapping (red lines) from a prototype $\mathbf{w}_i$ in the $\mathcal{W}$ input space codebook (lower red crosses & Voronoi regions) are projected using (3) (upper red shaded regions) onto the prototypes of the $\mathcal{V}$ output space codebook (upper red crosses & Voronoi regions) Right figure: Class discovery and cross-view class projection. After both the codebooks and co-occurrence mapping are trained (cf. Section III-A), the upper codebook prototypes are clustered to form class labels (upper blue & green regions, cf. Section III-B). An appropriate class label is then projected onto a prototype $\mathbf{w}_j \in \mathcal{W}$ (lower blue region) using (5) (cf. Section III-C).

B. Class Discovery

In order to find class clusters in the output space, we treat the prototype vectors as data points and employ traditional unsupervised clustering to cluster the prototypes. Often, the k-means clustering algorithm is used for such purposes, but one issue with regular k-means is that k, that being the number of clusters, must be selected in advance. It is possible, however, to augment the algorithm such that k may be estimated automatically. We employ the X-means algorithm [23] for that purpose here to find both the optimal k^* number of clusters for the prototypes in output space codebook $\mathcal{V}$, as well as the actual clustering $C_{k^*}^{\mathcal{V}} = \{V_1, \ldots, V_{k^*}\}$, where the V_i are subsets of prototypes in $\mathcal{V}$. The V_i clusters that are discovered in this way are treated as classes grounded in output space features that can be mapped back onto the input space layer using Hebbian projection. This class discovery and projection process is visualised in Fig. 3.

C. Cross-View Class Projection

For cross-view classification, we require a mapping $f : \mathbb{R}^m \to L(\mathcal{V})$ that maps input space samples to class labels, where $L()$ is some labelling function. To realise this labelling function, the $C_{k^*}^{\mathcal{V}}$ class clusters found in output space codebook $\mathcal{V}$ via class discovery are projected back to the input space codebook $\mathcal{W}$. By summing the posterior probabilities $P(\mathbf{v}_j|\mathbf{w}_i)$ provided by such a projection, we can determine the posterior probability of class cluster V_l in output view codebook $\mathcal{V}$ given prototype $\mathbf{w}^i$ in input space codebook $\mathcal{W}$ as follows

$$P(V_l|\mathbf{w}_i) = \sum_{\mathbf{v}_j \in V_l} P(\mathbf{v}_j|\mathbf{w}_i) P(\mathbf{w}_i). \tag{4}$$

This allows us to assign an output space class cluster label to each of the prototypes in the input space codebook by maximizing the category cluster posterior probability for each of them. Thus, given $\mathbf{w}^i$, we define a labelling function

$$L(\mathbf{w}_i) = \arg\max_{l=1,\ldots,k^*} P(V_l|\mathbf{w}_i) \tag{5}$$

that labels the input space prototypes on that basis.

D. Class Prediction

Given an input space test sample $\mathbf{x}$, its predicted output space class cluster may finally be determined using the labelling function from (5), the weighted squared Euclidean distance function from (2), and the nearest-neighbour rule as follows:

$$f(\mathbf{x}) = L\left(\arg\min_{\mathbf{w}_i \in \mathcal{W}} d^2(\mathbf{x}, \mathbf{w}_i)\right). \tag{6}$$

E. Feature Relevance Determination

Some features can prove to be more relevant than others for class prediction, and determining the extent of their relevance and exploiting this information can improve classification accuracy. To this end, we make use of an algorithm developed in previous work for feature relevance determination, specifically Algorithm 1 from [24]. It exploits the positioning of the prototypes in the input feature space to estimate Fisher criterion scores for the input dimensions, and subsequently, to estimate the λ_i weighting factors in (2) for an adaptive distance function that accounts for feature relevance with respect to classifier output. Due to the bootstrapped nature of the learning algorithm described in this paper, class information may not be fully formed at a given time step during training. Therefore, to avoid corrupting the online learning process we do not apply the feature weights during training, but apply them at classification time instead. Further details on this method may be found in [24].

IV. EXPERIMENTS

To test our affordance learning system, the experimental environment was set up as shown in Figure 1. We selected 5 household objects (cf. Fig. 4) 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. A number of object push tests were carried out for each of the 5 objects listed previously and the resulting data was processed, leaving 134 data samples. Objects were placed at random start locations 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 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. For evaluation the samples were hand-labelled with four ground truth labels: left rotation, right rotation, forward translation and forward topple, but this information was of course not used by our self-supervised learning algorithm, and was used strictly for evaluation and for training the supervised classifiers outlined below in Section IV-A. Sample object interactions are shown in Fig. 5. The results presented below examine three different aspects of our learning framework: class discovery, class prediction and feature relevance determination.

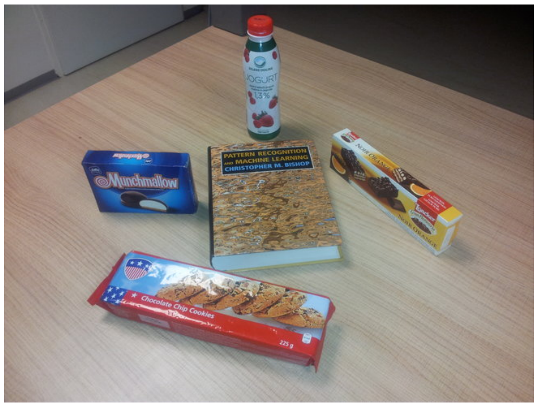


Fig. 4. Test objects used in our experiments.

A. Evaluation Procedure

To test our self-supervised learning paradigm on the dataset described above, we performed 10-fold cross-validation, evaluating performance online at regular intervals over the training period. Our self-supervised learning vector quantization algorithm (SSLVQ) [5], as well as a variation employing feature relevance determination at classification time (SSLVQ (FRC)) [5], [24], were compared alongside the supervised LVQ algorithms GLVQ [25], GRLVQ [26] and SRNG [27] in this online evaluation. Two main experiments were performed, the first one performing 10-fold cross-validation for 1 epoch over the training data to test short-term training performance, and the second one performing 10-fold cross-validation for 10 epochs over the training data over 10 trials to test more long-term training performance. In each case, codebooks in both the input space and output space consisted of 64 prototypes arranged in a 8×8 hexagonal lattice with a sheet-shaped topology [5], [22]. The feature weights of the codebook prototype vectors were randomly initialized to test the abilities of the algorithms to learn from scratch. The 10-fold cross-validation was therefore performed in 10 trials and results were averaged in order to account for the variation in codebook initialization between trials. The learning phase was switched from unsupervised learning to self-supervised learning one tenth of the way through training. Batch SVM was also performed outside of the online evaluation for reference. Batch methods, as opposed to online methods that are trained sample-by-sample, have access to the entire training set during training, and therefore usually provide superior results. SVM parameters were optimized using cross validation over the training data prior to training.

Given the self-supervision aspect, the evaluation criteria, by necessity, differed from the traditional match-counting

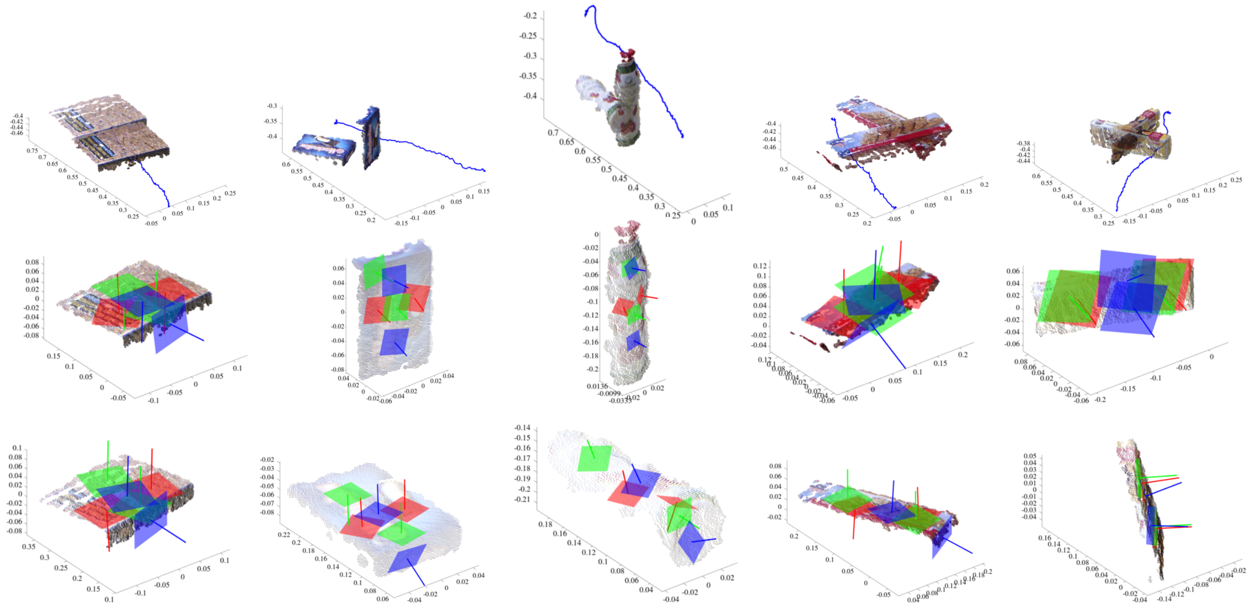


Fig. 5. Action-grounded shape feature extraction. Top row: pre-push and post-push 3-D point clouds and action trajectories for the five test objects being pushed in various different ways. Second & third rows: action-grounded shape feature extraction (cf. Section II-C) for the pre-push and post-push point clouds, respectively. 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.

utilized to evaluate fully-supervised classifiers. Clusters of prototypes were found in the output space codebook as described in Section III-B and subsequently matched to the ground truth classes by first matching all ground truth labelled training data to nearest-neighbour output space prototypes, then assigning each class cluster the ground truth label which their respective prototypes matched to most frequently. Then, given a test sample consisting of an input space test vector $\mathbf{x}_i$ and an output space test vector $\mathbf{y}_i$, the input space codebook was tasked with predicting an output space class cluster V_j for $\mathbf{x}_i$ using the process described in Section III-D. The output space test vector $\mathbf{y}_i$ was then matched to a class cluster V_k in the output modality via the nearest-neighbour rule. If the V_j class cluster predicted by the input codebook matched the V_k cluster and that cluster also matched the ground truth label for the test sample, this was deemed to be a correct classification.

B. *Results: Full Feature Set*

1) Class Discovery: An important consideration in evaluating whether or not our algorithm is capable of self-supervised multi-view learning is to examine if it is capable of successfully finding class clusters in the output space, without which self-supervised discriminative learning in the input space would not be possible. Recall that this is achieved by clustering prototypes in the output space at classification time using X-means clustering (cf. Section III-B). But how quickly do the prototypes position themselves such that this clustering may happen successfully? The leftmost graphs of Figures 6 and 7 answer this question by showing the rate at which ground truth labelled output space test samples fall within output space class clusters with matching ground truth labels (cf. Section IV-A) over time. As is evident from the graph for short-term training over 1 epoch, performance starts to peak around half-way through training and reaches near-optimal performance by the end, in which case output view test samples were correctly matched to the four ground-truth affordance classes over 94% of the time by the end of training. In the case of long-term training over 10 epochs, near-optimal performance is achieved early in the training process and is maintained throughout, meaning that cross-view prediction should at least have the opportunity to reach optimal ground truth prediction rates early on.

2) Class Prediction: With regard to class prediction, batch SVM scores 92% classification accuracy using ground truth labels, and although the other learners were not expected to perform at this level given the fact that they were trained on-line from a random initialization, this serves as a good reference point. Turning to the middle graphs on class prediction in Figs. 6 and 7, most of the learners perform poorly in short-term training, most of them scoring below 50% classification accuracy, likely due to there being insufficient training time to tackle the complexity of the problem. The self-supervised learners, reaching a rate of 42%, out-perform all of the supervised classifiers apart from GRLVQ, which reaches just under 50% accuracy. The self-supervised learners capitalise slightly in this case from the dynamic class labelling process that occurs when classifying as described in Section III-D, while the class labels for the prototypes of the supervised classifiers are randomly distributed, which means that it takes time for the labelled prototypes to cluster appropriately. Long-term training sees all of the learners performing better,

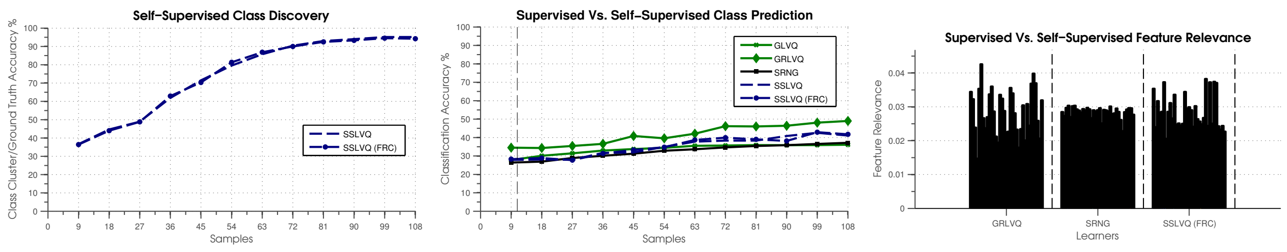


Fig. 6. Results for 1 epoch of online training over the full feature set. From left to right: class discovery, class prediction and feature relevance results are shown for 10-fold cross-validation averaged over 10 trials with random prototype initialization (cf. Section IV-A). Bold vertical dashed lines in the class prediction graphs indicate training phase shifts from unsupervised to self-supervised learning (cf. Section III-A). Comparitive batch SVM class prediction score: 92%.

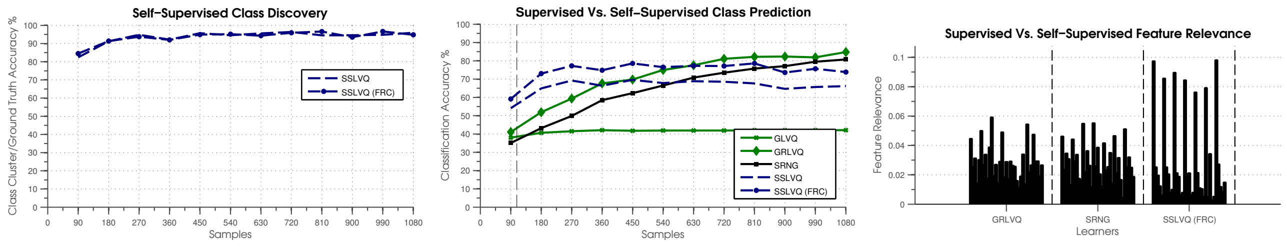


Fig. 7. Results for 10 epochs of online training over the full feature set. Results are shown for 10-fold cross-validation averaged over 10 trials with random prototype initialization (cf. Section IV-A). Comparative batch SVM class prediction score: 92%.

with GRLVQ scoring 85%, SRNG scoring 81%, SSLVQ (FRC) scoring 74% SSLVQ scoring 66%, and GLVQ scoring 42%. by the end of training. In this instance it is possible to see the benefit of feature relevance determination, with its addition boosting the performance of self-supervised learning by 8%. GLVQ is known to suffer issues with poor prototype initialization, hence its relatively poor performance. SRNG has a slower update rule than GRLVQ, which likely accounts for its relatively slow adaptation here.

3) Feature Relevance Determination: The rightmost graphs of Figures 6 and 7 show average feature relevances at the end of training as determined by the three learners GRLVQ, SRNG and SSLVQ (FRC) which have feature relevance determination capabilities. In the 10-epoch case, all three of them highlight the following list of features as being most significant for class prediction:

- Overall point cloud, centroid x-coordinate.
- x-axis, left side part, centroid x-coordinate.
- x-axis, right side part, centroid x-coordinate.
- y-axis, front side part, centroid x-coordinate.
- z-axis, top side part, centroid x-coordinate.

In general, the object part centroid features were determined to be the most discriminative for affordance class prediction.

C. Results: Reduced Feature Set

Using this knowledge, we performed an additional set of experiments on a reduced feature set over the same experimental data, this time using only the centroids of the sub-parts to form the feature vectors in both the input and output spaces. The experimental parameters were kept the same as in Section IV-B and the results for these experiments are shown in Figures 8 and 9. This refinement of the feature spaces boosts the predictive performance of SSLVQ (FRC) up to 87% over 10 epochs of training, a significant improvement over the full feature set.

V. CONCLUSIONS

In summary, the main contribution of this paper was the proposal of an action-grounded 3-D visual feature descriptor to be used for bootstrapping object affordances in autonomous robots when little prior information is available about the objects. We have demonstrated through experimental results how, when used in combination with a self-supervised learning algorithm, this feature descriptor is effective at facilitating both affordance class discovery and prediction in an online learning setting with a number of initially unknown objects and object affordances. A feature relevance determination extension to the self-supervised algorithm was also shown to boost affordance class prediction results by emphasizing the discriminative contribution of particular features within the descriptor.

With regard to future work, firstly, we aim to migrate the affordance learning techniques presented here to a humanoid robotic system. The design of the shape features could potentially be improved by matching 2-D invariant image features to 3-D surface features and thereby directly tracking object part motion during interaction. Looking towards improving the affordance learning aspects, we aim to implement regression capabilities that would allow for continuous prediction of object and object part positions. The bootstrapping of significant affordance classes as presented in this paper would mitigate this task, constraining the problem by guiding the development of multiple continuous models.

REFERENCES

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

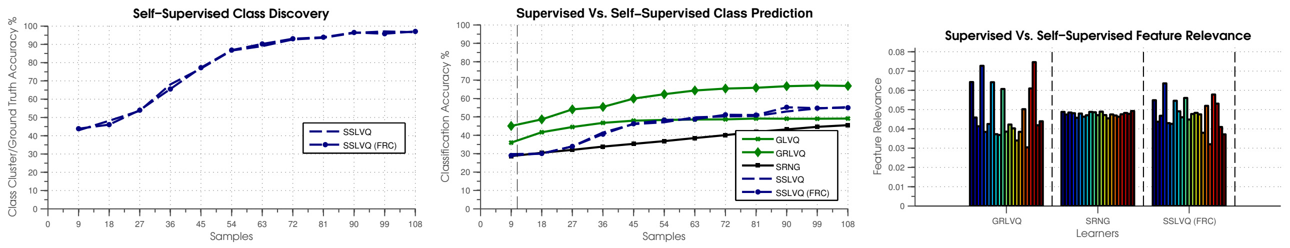


Fig. 8. Results for 1 epoch of training over the reduced feature set. Class discovery, class prediction and feature relevance results are shown for 10-fold cross-validation averaged over 10 trials with random prototype initialization (cf. Section IV-A). Comparative batch SVM class prediction score: 96%.

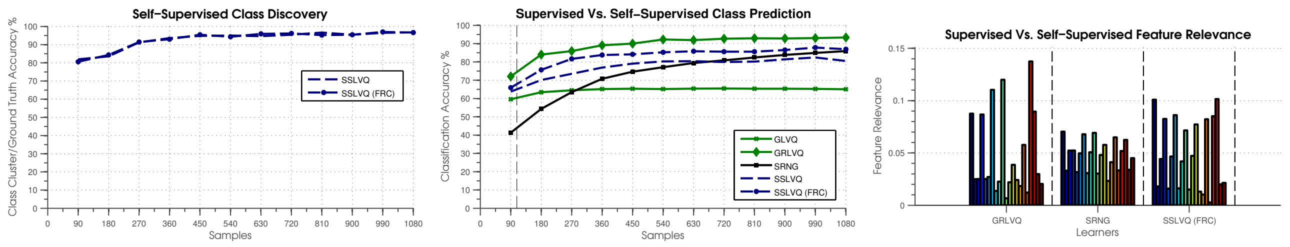


Fig. 9. Results for 10 epochs of online training over the restricted feature set using only part centroids. Again, results are shown for 10-fold cross-validation averaged over 10 trials with random prototype initialization (cf. Section IV-A). Batch SVM class prediction score: 96%.

[2] 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)*, vol. 3, 2003.

[3] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory-motor coordination to imitation," vol. 24, no. 1, p. 15–26, 2008.

[4] 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)*, Dec. 2009, pp. 277 –283.

[5] B. Ridge, D. Skočaj, and A. 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)*. Anchorage, USA: IEEE, May 2010, pp. 5047–5054.

[6] 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)*, May 2011, pp. 5722 –5729.

[7] G. Fritz, L. Paletta, M. Kumar, G. Dorffner, R. Breithaupt, and E. Rome, "Visual learning of affordance based cues," vol. 9, p. 52–64, 2006.

[8] I. Cos-Aguilera, L. Canamero, and G. Hayes, "Using a SOFM to learn object affordances," in *Proceedings of the 5th Workshop of Physical Agents (WAF'04), Girona, Spain*, 2004.

[9] J. Sinapov and A. Stoytchev, "Detecting the functional similarities between tools using a hierarchical representation of outcomes," in *Proceedings of the 7th IEEE International Conference on Development and Learning (ICDL)*, Aug. 2008, pp. 91 –96.

[10] S. Griffith, J. Sinapov, M. Miller, and A. Stoytchev, "Toward interactive learning of object categories by a robot: A case study with container and non-container objects," in *Proceedings of the 8th IEEE International Conference on Development and Learning (ICDL)*. IEEE, June 2009, pp. 1–6.

[11] E. Ugur, E. Oztop, and E. Sahin, "Goal emulation and planning in perceptual space using learned affordances," 2011.

[12] L. Paletta and G. Fritz, "Reinforcement learning of predictive features," in *Proceedings of 31st Workshop of the Austrian Association for Pattern Recognition (AAPR/OAPR)*, 2007, p. 105–112.

[13] 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)*, Oct. 2012, pp. 3260 –3267.

[14] T. Hermans, F. Li, J. M. Rehg, and A. 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, Aug. 2013.

[15] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," vol. 24, no. 6, p. 381–395, 1981.

[16] R. B. Rusu, "Semantic 3D object maps for everyday manipulation in human living environments," Ph.D. dissertation, Computer Science department, Technische Universitaet Muenchen, Germany, Oct. 2009.

[17] S. Sun, "A survey of multi-view machine learning," p. 1–8, 2013.

[18] V. R. de Sa, "Learning classification with unlabeled data," in *Advances in Neural Information Processing Systems 6*. Denver, CO, USA: Morgan Kaufmann, 1994, pp. 112–119.

[19] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, 1998, p. 92–100.

[20] S. Bickel and T. Scheffer, "Multi-view clustering," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, Washington D.C., USA, Nov. 2004, pp. 19–26.

[21] V. de Sa, P. Gallagher, J. Lewis, and V. Malave, "Multi-view kernel construction," vol. 79, no. 1, pp. 47–71, 2010.

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

[23] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *Proceedings of the Seventeenth International Conference on Machine Learning*, vol. 1, 2000, p. 727–734.

[24] B. Ridge, A. Leonardis, and D. Skočaj, "Relevance determination for learning vector quantization using the fisher criterion score," in *Proceedings of the Seventeenth Computer Vision Winter Workshop (CVWW)*, Mala Nedelja, Slovenia, Feb. 2012.

[25] A. Sato and K. Yamada, "Generalized learning vector quantization," in *Advances in Neural Information Processing Systems 8*. Denver, CO, USA: MIT Press, 1996, p. 423–429.

[26] B. Hammer and T. Villmann, "Generalized relevance learning vector quantization," vol. 15, no. 8-9, pp. 1059–1068, 2002.

[27] B. Hammer, M. Strickert, and T. Villmann, "Supervised neural gas with general similarity measure," vol. 21, no. 1, p. 21–44, 2005.

3D Object Class Geometry Modeling with Spatial Latent Dirichlet Markov Random Fields[⋆]

Hanchen Xiong Sandor Szedmak Justus Piater

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

Abstract. This paper presents a novel part-based geometry model for 3D object classes based on latent Dirichlet allocation (LDA). With all object instances of the same category aligned to a canonical pose, the bounding box is discretized to form a 3D space dictionary for LDA. To enhance the spatial coherence of each part during model learning, we extend LDA by strategically constructing a Markov random field (MRF) on the part labels, and adding an extra spatial parameter for each part. We refer to the improved model as spatial latent Dirichlet Markov random fields (SLDMRF). The experimental results demonstrate that SLDMRF exhibits superior semantic interpretation and discriminative ability in model classification to LDA and other related models.

1 Introduction

During the past decades, computer vision has made remarkable progress in visual object understanding, e.g. classification, pose estimation and segmentation, etc. However, most previous study of object modeling is based on 2D images, in which appearance is the main and only information source for various tasks, so most attention is focused on increasing the robustness of algorithms to lighting changes, intra-class appearance variation and viewpoint variation [4]. Meanwhile, 3D geometry properties of objects have been rarely exploited and used to increase the expressiveness of object models. Recently, pioneering work [7,13] has attempted to add 3D geometric information to object models, demonstrating that the accuracy and robustness of such algorithms can be enhanced with extra 3D geometry clues. However, there still exists an obvious gap between 2D appearance modeling and 3D geometry modeling with respect to their interpretation and representation abilities, and it has been advocated [7,13] that robust 3D geometry modeling is highly desirable. Motived by this gap and desire, this paper puts forward a novel 3D object class geometry model in the light of state-of-the-art techniques developed in machine learning and computer graphics. Part-based models have displayed merits in 2D appearance modeling [4] for handling partial occlusion, our 3D geometry model is likewise part-based and inherits these strengths. The training data of our algorithm are collections of 3D

⋆ The research has received funding from the European Community's Seventh Framework Programme (FP7) under grant agreement no. 270273, Xperience.

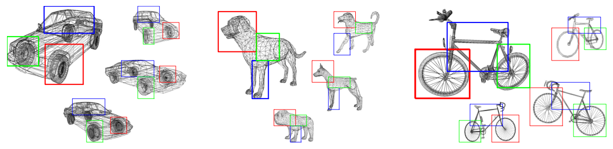


Fig. 1. Different object instances of the same class should share similar 3D structure of composing parts, although their parts can slightly vary from one instance to another.

models of different instances which belong to the same category (Figure 1). The basic underlying principle of our modeling is the concept that different object instances of the same class should share similar 3D structure of composing parts, although their parts can slightly vary from one instance to another. For example, all bicycles are composed of a frame and two wheels, and the geometric relation between these three parts are similar across different instances (Figure 1). In this paper, 3D objects are represented by point cloud data (PCD), which is a general and popular representation of 3D shapes and can easily be converted from other data formats (e.g. meshes). First, for each class, different PCDs of object instances are aligned using point cloud registration methods. Secondly, the main learning step is inspired by latent Dirichlet allocation (LDA) [1] and computer graphics [5]. LDA is a state-of-the-art machine learning tool for discovering latent topics within document collections. Here we apply LDA by considering each object point cloud as a document, and each part as a topic. With the bounding box volume discretized into a 3D grid dictionary, the part can be mined out as a multinomial distribution over the discrete 3D space, and each object is a multinomial distribution over parts. However, standard LDA ignores the spatial coherence, which is of great importance in our task but not generally taken into account in natural language applications. Based on discoveries in computer graphics [5] and other work on LDA [8,11], we develop a spatial latent Dirichlet Markov random field (SLDMRF) model with extra undirected links between topic labels and spatial parameters. The proposed SLDMRF can co-segment all point clouds simultaneously under a prior of coherence of correspondence, spatial continuity and spatial smoothness. According to our empirical results (section 3), compared to standard LDA and other related models, SLDMRF can achieve much more consistent and semantically meaningful segmentations of 3D point clouds, and the learned class geometry models display better discriminative ability in classification.

1.1 Related work

The starting point of 3D geometry modeling in visual object understanding is the difficulty in dealing with appearance variation due to different viewpoints. There have been several attempts to embed 3D geometric information into object models [2,3,7,13], and all of them have reported improvement in accuracy and robustness, although different 3D geometry information are exploited and modelled in their work. In [2] 3D object shapes are probabilistically modelled as continuous distributions in $\mathbb{R}^3$ with a kernel density estimator (KDE). However, that work explicitly addresses neither category-level tasks nor semantic segmentation. Objects are considered as Gausssaian mixtures and expectation-maximization (EM) is applied to estimate corresponding Gaussian parameters and weights. One observation of Gaussian-mixture-based segmentation is that discovered parts rarely display good semantic interpretability since usually the part geometry is too complex to be modelled as a Gaussian (section 3.1). Other work attempts to improve the expressiveness of object geometry models in different ways. For example, Detry et al. [3] represent objects as hierarchically-organized spatial distributions of distinct feature types, but did not seek to produce semantically-meaningful segmentations. Other models [7,13] extract 3D geometry information at the class level. However, in [7] the segmentation is again based on Gaussian mixtures and EM, and most [13] do not model object classes in a part-based manner to avoid segmentation. Meanwhile, another thread of segmentation-based visual modeling is the application of Latent Dirichlet allocation (LDA) in computer vision [10,11,8]. LDA was originally developed to discover hidden topics in text corpora by clustering words into different topics [1]. Standard LDA, however, ignores spatial coherence, which is problematic in vision applications. Therefore, spatial LDA (SLDA) [11] and Latent Dirichlet Markov random fields (LDMRF) [8] were put forward to produce better, spatially-coherent segmentations. In addition, with higher emphasis of the smoothness of parts and consistent correspondences, 3D segmentation in computer graphics [5] constructs graphs with neighboring intra-links and correspondence inter-links among objects, and min-cut is used on graphs for segmentation.

The main contribution of this paper is an extension of LDA for 3D object class geometry modeling, which we refer to as Spatial Latent Dirichlet Markov Random Fields(SLDMRF). The proposed model is built with inspiration from recent advances in different fields [1,11,8,5], and it yields superior interpretability and representational capability in modeling 3D object class geometry.

2 3D Object Class Geometry Modeling

With the point cloud representations of 3D object shapes, the alignment of different instances of the same class is achieved with point cloud registration algorithms. While any suitable registration procedure can be used, we adopted a novel method [12] since it is very efficient and robust to non-rigid transformation, which suits the case of intra-category shape variation. An example of aligning dogs is displayed in Figure 2.

2.1 Latent Dirichlet Allocation

LDA [1] is a generative model that utilizes the information of co-occurring words to find out hidden topics shared by documents. In LDA, each document is considered as a finite mixture of topics; each topic is a finite mixture of words. The graphical model of LDA is shown in Figure 3(a). The generative process of LDA

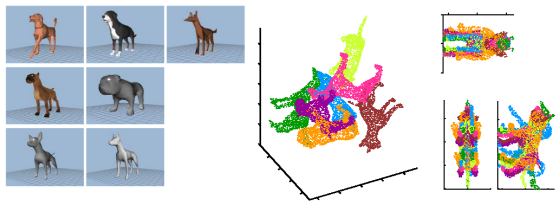


Fig. 2. Alignment of different dog instances by point cloud registration [12]. Left: original 3D shapes of different dog instances; middle: point clouds generated from the shapes on the left; right: three views (top, profile, front) of the point clouds (middle) after alignment.

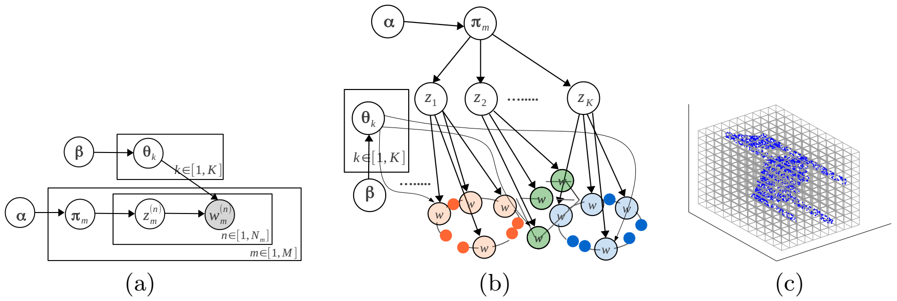


Fig. 3. (a) Graphical model of LDA; (b) Application of LDA to model 3D object categories; (c) Construction of 3D dictionary by discretizing the 3D space of the bounding box.

is as follows: (1) for each topic $k \in [1, K]$, a multinomial parameter $\boldsymbol{\theta}_k$ over words is sampled from Dirichlet prior $\boldsymbol{\beta}$; (2) for each document $m \in [1, M]$, a multinomial parameter $\boldsymbol{\pi}_m$ over K topics is sampled from Dirichlet prior $\boldsymbol{\alpha}$; (3) for each word $w_m^{(n)}$, $n \in [1, N_m]$ in document m, a topic label $z_m^{(n)}$ is first sampled from multinomial distribution $z_m^{(n)} \sim \text{Multinomial}(\boldsymbol{\pi}_m)$, then the word $w_m^{(n)}$ is sampled from the multinomial distribution parametrized with $\boldsymbol{\theta}_{z_m^{(n)}}$, $w_m^{(n)} \sim \text{Multinomial}(\boldsymbol{\theta}_{z_m^{(n)}})$. Hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ define the Dirichlet priors governing the parameters of multinomial distributions. Usually $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are set in a symmetric manner and using low values [6]. In [10], LDA is applied on a collection of images. Each image is considered as a document, objects correspond to topics, and visual words are generated using vector quantization on extracted features. In our case, however, LDA is utilized for 3D object class geometry modeling with objects of the same category as documents, and parts shared by different instances correspond to topics (Figure 3(b)).

3D Dictionary. In our task, LDA is expected to work effectively under the assumption that different objects of the same category should share very similar structure. Therefore, when LDA is applied on each collection of categorical object point clouds, the co-occurring patterns are the 3D space occupied by 3D points. In each collection, all instances can be aligned to a canonical pose, based on which the 3D space of the bounding box is discretized into a grid, where each block represents a 3D word. Therefore, when transferring point clouds to corresponding documents, word w_k will replace 3D point x_i if x_i lies within block w_k. In this way, the discovered part actually is a distribution over 3D space, and a category is a mixture of these distributions. The concept of dictionary discretization is illustrated in Figure 3(c).

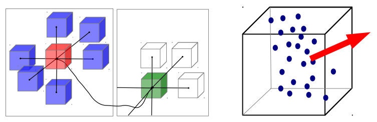
For label inference and parameters learning in LDA, a collapsed Gibbs sampling [6] can be formulated as

$$q_{\mathrm{LDA}}(z_m^{(n)} = k) \propto \frac{\mathbf{N}^{(k)}_{-mn,w_m^{(n)}} + \beta_{w_m^{(n)}}}{\sum_w^W (\mathbf{N}^{(k)}_{-mn,w} + \beta_w)} \cdot (\mathbf{N}^{(m)}_{-mn,k} + \alpha_k) \qquad (1)$$

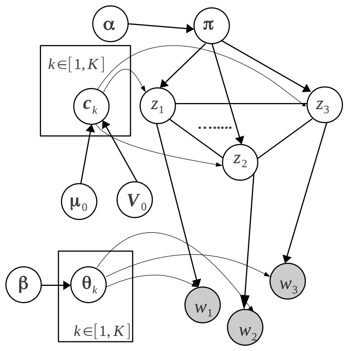
where $\mathbf{N}^{(k)}_{-mn,w}$ is the number of words in the corpus with value w assigned to topic k excluding the nth word in document m, and $\mathbf{N}^{(m)}_{-mn,k}$ is the number of words in document m assigned to topic k excluding the nth word in document m. From (1), it can be seen that LDA prefers to cluster together those words that often co-occur in the same document. Therefore, simply applying LDA on the 3D dictionary, unfortunately, is not expected to work because it misses a lot of spatial and correspondence information, which is not meaningful in the text processing case: (1) *Spatial coherence* is an important issue when LDA is applied in vision applications [11,8]. For example, in all point clouds of dogs, 3D words located in the hip and in the head will always co-occur. So by using (1), the hip and head of dogs can be clustered into a part, which is a spatially (of course also semantically) unreasonable segmentation. (2) *Correspondence coherence* is likewise important. LDA can find synonyms by finding their co-occurring patterns in documents. However, the "synonyms" in the 3D dictionary are identified by spatial correspondence. For example, in Figure 2, the legs of different dogs can rarely match exactly due to different species or standing poses. However, since all legs are close and correspond to each other, they should be clustered into the same part.

2.2 Spatial latent Dirichlet Markov random field

To enhance the spatial coherence, in Spatial LDA (SLDA) [11] 2D images are decomposed into small overlapping regions, which are used as documents to ensure that the pixels belonging to one part should be close to each other. Latent Dirichlet Markov random fields (LDMRF) [8], on the other hand, construct Markov random fields on the part label variables to enhance the local spatial coherence. However, both of them ignore the correspondences across the segmentations of different instances. Inspired by these improved versions of LDA and consistent co-segmentation in computer graphics [5], we put forward a novel spatial latent Dirichlet Markov random field (SLDMRF) that inherits virtues from both SLDA and LDMRF. However, rather than being a simple combination of SLDA and LDMRF, the proposed SLDMRF goes beyond them in several ways.



(a) Left: for each word (red block), there are two types of connections in SLDMRF: neighboring spatial connections (blue blocks) and correspondence connections (green block); right: the normal vector of each word in the document is estimated by using the points lying within the word.



(b) Graphical model of SLDMRF: compared to the standard LDA (Figure 3(a)), there are extra directed links between topic labels and spatial Gaussian parameters $\mathbf{c}_k$.

Fig. 4. SLDMRF modeling

First, instead of going through all small overlapping sub-volumes as SLDA does, we explicitly model the positions of all parts by parameters $\mathbf{c}_k$ such that 3D words that share the same label k are likely to be close to $\mathbf{c}_k$. Second, similarly to LDMRF, SLDMRF constructs a Markov random field on the neighboring label variables. However, different from LDMRF, we assign different potential functions based on the prior that the segmentation boundaries should be located at the point where abrupt curvature changes take place. The potential function is defined as

$$g(z_i, z_j) = \delta(z_i, z_j) \exp(|\langle o_i, o_j \rangle|) \tag{2}$$

where $\delta(z_i, z_j)$ equals 1 when z_i and z_j are neighbors, and 0 otherwise (Figure 4(a)), and o_i, o_j are the normal vectors estimated by using the points lying within word i and j respectively (Figure 4(a)). Last but not least, SLDMRF enhance the correspondences of segmentation across different instances. Inspired by the co-segmentation used in [5], we construct inter-connections between corresponding parts across different objects, and correspondences are matched by finding the nearest neighbors in other objects after alignment. In this way the segmentation can be more consistent within a category. The potential function $g(z_m^{(i)}, z_n^{(j)})$ for correspondence connections is set in the same way as spatial connections (2); $\delta(z_m^{(i)}, z_n^{(j)})$ is 1 if $z_m^{(i)}$ and $z_n^{(j)}$ are nearest neighbours of each other, and 0 otherwise. Because the part weights are already taken into account by LDA (parameter $\boldsymbol{\pi}$), the labels within the Markov random fields are modeled as:

$$p(\mathbf{Z}) \propto \exp\left(\sum_{(i,j)} g(z_i, z_j)\right) \tag{3}$$

The graphical model of SLDMRF is presented in Figure 4(b). Hyperparameters $\boldsymbol{\mu}_0$ and $\mathbf{V}_0$ (similar to $\boldsymbol{\alpha}, \boldsymbol{\beta}$) specify the Gaussian prior of part positions $\mathbf{c}_k \sim \mathcal{N}(\cdot; \boldsymbol{\mu}_0, \mathbf{V}_0)$. Given the part position $\mathbf{c}_k$, the label is sampled as $z_i \sim \mathcal{N}(\hat{w}_i; \mathbf{c}_k, \boldsymbol{\Lambda})$, where $\hat{w}_i$ denotes the 3D coordinates of word w_i. Since we do not expect the label distribution to be truly Gaussian, $\boldsymbol{\Lambda}$ is set relatively large. The joint probability of 3D words in SLDMRF $p(\{w_m^{(n)}\}_{m=1,n=1}^{M,N_m}, |\boldsymbol{\alpha}, \boldsymbol{\beta})$ is:

$$\frac{1}{\mathbf{Q}} \prod_{m=1}^{M} \prod_{n=1}^{N_m} \Big(\int_{\boldsymbol{\pi}_m} \int_{\boldsymbol{\theta}_{z_m^{(n)}}} \int_{\mathbf{c}_{z_m^{(n)}}} p(\boldsymbol{\pi}_m|\boldsymbol{\alpha}) \sum_{z_m^{(n)}=1}^{K} \Big(p(z_m^{(n)}|\boldsymbol{\pi}_m) p(w_m^{(n)}|\boldsymbol{\theta}_{z_m^{(n)}}) \\ \mathcal{N}(w_m^{(n)}; \mathbf{c}_{z_m^{(n)}}, \boldsymbol{\Lambda}) \mathcal{N}(\mathbf{c}_{z_m^{(n)}}; \boldsymbol{\mu}_0, \mathbf{V}_0) \Big) \times \prod_{x,y \in \tilde{z}_m^{(n)}} \sqrt{\exp(g(z_m^{(n)}, z_x^{(y)}))} \Big) \tag{4}$$

where $x, y \in \tilde{z}_m^{(n)}$ denotes the set of all word labels (the yth word in the xth document) connected with $z_m^{(n)}$, i.e. $\delta(z_m^{(n)}, z_x^{(y)}) = 1$, and $\mathbf{Q}$ is the normalization term induced by Markov random fields.

2.3 Inference and learning

Similar to the inference and learning in LDA, based on (4), we can develop a collapsed Gibbs sampler by integrating out $\boldsymbol{\pi}_m, \boldsymbol{\theta}_{z_m^{(n)}}, \mathbf{c}_{z_m^{(n)}}$ in SLMRF. The sampler can be more easily interpreted as a "combined" sampler by using clues from LDA, MRF and Guassian mixtures:

$$q^*(z_m^{(n)} = k) \propto q_{\text{LDA}}(z_m^{(n)} = k) \cdot q_M(z_m^{(n)} = k) \cdot q_c(z_m^{(n)} = k) \tag{5}$$

where $q_{\text{LDA}}(z_m^{(n)} = k)$ is the collapsed Gibbs sampler of LDA (1),

$$q_M(z_m^{(n)} = k) \propto \frac{\exp\left(\sum_{(z_j, z_m^{(n)})} g(z_j, z_m^{(n)} = k)\right)}{\sum_h \exp\left(\sum_{(z_j, z_m^{(n)})} g(z_j, z_m^{(n)} = h)\right)} \tag{6}$$

is the Gibbs sampler based on the Markov random field, and

$$q_c(z_m^{(n)} = k) \propto \frac{\mathcal{N}(\hat{w}_m^{(n)}; \boldsymbol{\mu}_l^{(k)}, \boldsymbol{\Lambda}_l^{(k)})}{\sum_h \mathcal{N}(\hat{w}_m^{(n)}; \boldsymbol{\mu}_l^{(h)}, \boldsymbol{\Lambda}_l^{(h)})} \tag{7}$$

is a collapsed Gibbs sampler of Gaussian mixtures, with

$$\boldsymbol{\Lambda}_l^{(k)^{-1}} = \boldsymbol{\Lambda}_0^{-1} + l\boldsymbol{\Lambda}^{-1} \qquad \boldsymbol{\mu}_l^{(k)} = \boldsymbol{\Lambda}_l^{(k)^2} \left(\frac{\boldsymbol{\mu}_0}{\boldsymbol{\Lambda}_0^2} + \frac{l\overline{\hat{w}^{(k)}}}{\boldsymbol{\Lambda}^2} \right) \tag{8}$$

where l is the number of words which are labeled with k, and $\overline{\hat{w}^{(k)}}$ is the mean of 3D coordinates of words which are assigned to part k until the current iteration. Similar to [6], parameters $\{\boldsymbol{\pi}_m\}_{m=1}^{M}$, $\{\boldsymbol{\theta}_k\}_{k=1}^{K}$ can be estimated after the convergence of Gibbs sampling:

$$\boldsymbol{\pi}_m^{(k)} = \frac{\mathbf{N}_k^{(m)} + \alpha_k}{\sum_{k=1}^{K}(\mathbf{N}_k^{(m)} + \alpha_k)} \qquad \boldsymbol{\theta}_k^{(w)} = \frac{\mathbf{N}_w^{(k)} + \beta_w}{\sum_w^W (\mathbf{N}_w^k + \beta_w)} \tag{9}$$

Since hyperparameter $\boldsymbol{\alpha}$, in our case, is categorical part weight, we estimate it by simply compute the average of π_m: $\boldsymbol{\alpha} = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{\pi}_m$. In addition, parameters $\{\mathbf{c}_k\}_{k=1}^{K}$ are read out as $\{\boldsymbol{\mu}_l^k\}_{k=1}^{K}$ (8).

3 Experiments

To evaluate the proposed model, 5 object classes (cars, bikes, dogs, motorcycles, airplanes) from the Princeton shape benchmark (PSB) [9] database are used. Since 3D shapes in the PSB are represented as triangulated meshes, we convert them to point clouds by uniformly sampling points within triangles.

3.1 3D Object class geometry modeling

For comparison, LDA [1], LDMRF [8] and Gaussian Mixtures (GM) models are tested on the same data (aligned point clouds of categorical instances). Since LDMRF requires manual interference (semi-supervision), to avoid human bias during comparison, here we construct the same Markov random fields for both LDMRF and SLDMRF so that LDMRF can also work in an unsupervised manner. All four models are implemented with Gibbs sampling for label inference and learning. To ensure fairness, the same part number and iteration number (200) is applied. A test example of motorcycle modeling is presented in Figure 5. LDA does not find consistent and meaningful parts because of the intra-class variation (each object is labeled as a part since LDA only focuses on co-occurring patterns). LDMRF, on the other hand, discovers some locally continuous and consistent segments on different objects. However, the global spatial coherence of parts is poor; parts are shattered. GM establishes more globally obvious segmentation pattern by finding more consistent and meaningful parts. Nevertheless, GM ignores local spatial coherence, so parts are not well segmented; they tend to be of blob shape and to overlap each other. By contrast, SLDMRF produces best convincing segmentations in terms of consistence, local and global spatial coherence and semantic meaning. The SLDMRF modeling results of other four object classes are illustrated in Figure 6.

3.2 Geometry model classification

To illustrate the parts learned by SLDMRF is more accurate, and thus more discriminative, we conduct quantitative comparisons on classification task. Since LDA and LDMRF are far from being qualified for practical part-based modeling, here we are only concerned with the comparison between GM and SLDMRF. 3D shapes of 5 object classes are divided into training (70%) and test sets (30%). The model learning is conducted in the same way as in section 3.1. Although Markov random fields and spatial parameters greatly assists in segmentation and model learning of SLDMRF, they are not used in the final category models. A learned bicycle model is shown in Figure 6(e). It can be seen that the part position information and neighboring correlation are already encoded in the categorical part parameter $\boldsymbol{\theta}$. Therefore, for the sake of simplicity and computation feasibility, we only use learned LDA as our 3D object category models for classification. To test an object M^*, it is first aligned to the canonical poses of different class models. In SLDMRF case, the likelihood that M^* belongs to a

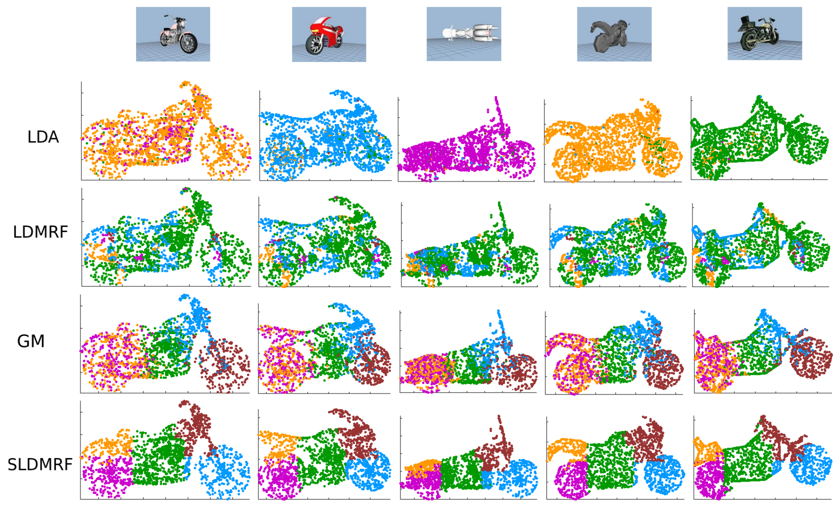


Fig. 5. Comparison of segmentation by using LDA, LDMRF, GM and SLDMRF, SLDMRF qualitatively yields more reasonable segmentations than the others.

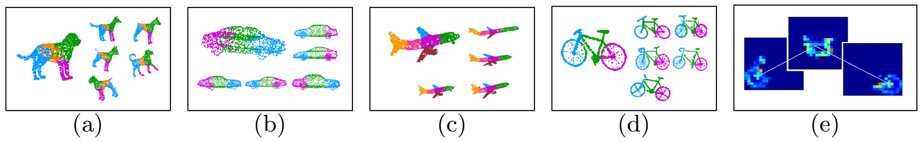


Fig. 6. SLDMRF modeling of dogs (a), cars (b), airplanes (c) and bikes (d); (e): the learned part parameters $\boldsymbol{\theta}_k$ of bikes.

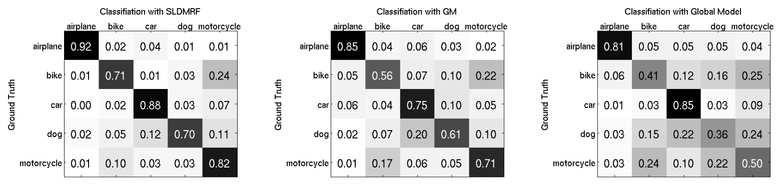
class $x \in \{\text{cars, bikes, dogs, motorcycles, airplanes}\}$ is computed as:

$$p(M^*|\mathcal{M}_x) = \prod_{i=1}^{|M^*|} \left\{ \sum_k \int_{\boldsymbol{\pi}} p(w_i|\boldsymbol{\theta}_k) p(k|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\boldsymbol{\alpha}) \right\} \tag{10}$$

where $|M^*|$ is the number of points in object M^*. By contrast, in the GM case:

$$p(M^*|\mathcal{M}_x) = \prod_{i=1}^{|M^*|} \left\{ \sum_k \mathcal{N}(w_i; \boldsymbol{\theta}_k) \pi_k \right\} \tag{11}$$

Since no other prior knowledge is given, the classification can be done in a maximum-likelihood fashion. A global model learned with one single multinomial distribution on 3D dictionary is also provided as baseline for comparison. The classification performances of GM, SLDMRF and global model are evaluated using confusion matrices. The comparison in in Figure 7 demonstrates that SLDMRF is superior to GM with respect to discriminative ability in classification.



Classifiation with SLDMRF

Ground Truth	airplane	bike	car	dog	motorcycle
airplane	0.92	0.02	0.04	0.01	0.01
bike	0.01	0.71	0.01	0.03	0.24
car	0.00	0.02	0.88	0.03	0.07
dog	0.02	0.05	0.12	0.70	0.11
motorcycle	0.01	0.10	0.03	0.03	0.82

(a)

Classifiation with GM

Ground Truth	airplane	bike	car	dog	motorcycle
airplane	0.85	0.04	0.06	0.03	0.02
bike	0.05	0.56	0.07	0.10	0.22
car	0.06	0.04	0.75	0.10	0.05
dog	0.02	0.07	0.20	0.61	0.10
motorcycle	0.01	0.17	0.06	0.05	0.71

(b)

Classifiation with Global Model

Ground Truth	airplane	bike	car	dog	motorcycle
airplane	0.81	0.05	0.05	0.05	0.04
bike	0.06	0.41	0.12	0.16	0.25
car	0.01	0.03	0.85	0.03	0.09
dog	0.03	0.15	0.22	0.36	0.24
motorcycle	0.03	0.24	0.10	0.22	0.50

(c)

Fig. 7. The classification confusion matrcices of 5 object classes with SLDMRF (a), GM (b) and global model (c).

4 Conclusion and Discussion

We improved LDA model for geometry modeling with better semantic interpretation and promising discriminative capabilities. Meanwhile, learning and application of the model require good initial alignment, which is difficult for noisy and partial occluded 3D point cloud in practice. So a promising future work is to cooperate 3D geometry model with 2D image models to describe both structure and appearance, which thus enhance model's expressiveness and practical value.

References

1. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet Allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
2. Detry, R., Piater, J.: Continuous Surface-Point Distributions for 3D Object Pose Estimation and Recognition. In: ACCV (2010)
3. Detry, R., Pugeault, N., Piater, J.: A Probabilistic Framework for 3D Visual Object Representation. PAMI 31(10), 1790–1803 (10 2009)
4. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object Detection with Discriminatively Trained Part-Based Models. PAMI 32(9), 1627–1645 (2010)
5. Golovinskiy, A., Funkhouser, T.A.: Consistent segmentation of 3D models. Computers and Graphics 33, 262–269 (2009)
6. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proceedings of the National Academy of Sciences 101(Suppl. 1), 5228–5235 (April 2004)
7. Jörg Liebelt and Cordelia Schmid: Multi-View Object Class Detection with a 3D Geometric Model. In: CVPR (2010)
8. Mackey, L.: Latent Dirichlet Markov Random Fields for Semi-supervised Image Segmentation and Object Recognition (2007)
9. Shilane, P., Min, P., Kazhdan, M.M., Funkhouser, T.A.: The Princeton Shape Benchmark. In: SMI. pp. 167–178. IEEE Computer Society (2004)
10. Sivic, J., Russell, B.C., Efros, A.A., Zisserman, A., Freeman, W.T.: Discovering object categories in image collections. In: ICCV (2005)
11. Wang, X., Grimson, E.: Spatial Latent Dirichlet Allocation. In: NIPS (2007)
12. Xiong, H., Szedmak, S., Piater, J.: Efficient,General Point Cloud Registration with Kernel Feature Maps. In: Canadian Conf. on Computer and Robot Vision (2013)
13. Yan, P., Khan, S.M., Shah, M.: 3D model based object class detection in an arbitrary view. In: ICCV (2007)

Efficient, General Point Cloud Registration With Kernel Feature Maps

Hanchen Xiong, Sandor Szedmak, Justus Piater
Institute of Computer Science, University of Innsbruck
Technikerstr.21a A-6020, Innsbruck, Austria
Email: {hanchen.xiong, sandor.szedmak, justus.piater}@uibk.ac.at

***Abstract*—This paper proposes a novel and efficient point cloud registration algorithm based on the kernel-induced feature map. Point clouds are mapped to a high-dimensional (Hilbert) feature space, where they are modeled with Gaussian distributions. A rigid transformation is first computed in feature space by elegantly computing and aligning a small number of eigenvectors with kernel PCA (KPCA) and is then projected back to 3D space by minimizing a consistency error. $SE(3)$ on-manifold optimization is employed to search for the optimal rotation and translation. This is very efficient; once the object-specific eigenvectors have been computed, registration is performed in linear time. Because of the generality of KPCA and $SE(N)$ on-manifold method, the proposed algorithm can be easily extended to registration in any number of dimensions (although we only focus on 3D case). The experimental results show that the proposed algorithm is comparably accurate but much faster than state-of-the-art methods in various challenging registration tasks.**

***Keywords*-kernel method; point cloud registration; SE(3) on-manifold optimization**

I. INTRODUCTION

3D free-form shape registration is an important problem in many fields and has sparked a large volume of related research literature. During the past two decades, 3D point clouds have become an increasingly more important and popular data structure to represent 3D shapes. Especially in contemporary robotics, 3D point cloud registration is an essential component of autonomous systems to assist in the perception of 3D objects and environments.

Until now, most existing 3D point cloud registration algorithms decompose the registration problem into two parts, *correspondence assignment* and *alignment*, because they argue that computing either of two steps will facilitate the other. A popular method is Iterative Closest Point (ICP) [1], which undoubtedly has been most widely used due to its simplicity in implementation. First, a pseudo correspondence is established by finding the nearest neighbor of each point. Then, the optimal rotation and translation are computed so as to minimize the average of Euclidean distances between all pairs of corresponding points. These two steps are iterated until converge. Obviously, the closest-distance criterion for correspondence is too weak, and therefore ICP can easily fail in practice when the displacement between two point clouds or outlier rate is relatively large. To enhance the accuracy of the correspondence, many improved versions of ICP were proposed by incorporating color, normal vectors, curvature, or strategically ignoring some unlikely correspondences [2]. However, despite various improvements, the *hard* assignment strategy employed by ICPs causes problems that require manual assistance in practical applications. To overcome this limitation, SoftAssign [3] and EM-ICP [4] were proposed by establishing one-to-many *soft* correspondences. Both methods assume that one point may correspond to all points in the other cloud with different likelihoods by constructing a correspondence matrix. To iteratively update this matrix, deterministic annealing is used in SoftAssign while an Expectation-Maximization (EM)-style method is employed in EM-ICP. Meanwhile, recently a Gaussian-mixture (GM) method [5] was developed to avoid iteratively computing the correspondences and alignment. GM probabilistically and globally models 3D point clouds as Gaussian mixtures in $\mathbb{R}^3$, and the optimal alignment between point clouds is computed by minimizing the discrepancy (L2 distance [5]) between their corresponding distributions.

For the task of aligning 3D point clouds $\mathbf{M}_1 = \{\mathbf{x}_i^{(1)}\}_{i=1}^{l_1}$ with $\mathbf{M}_2 = \{\mathbf{x}_i^{(2)}\}_{i=1}^{l_2}$, all methods described above can be interpreted as optimizing a common objective function:

$$\{\mathbf{R}^*, \mathbf{b}^*\} = \arg\min_{\mathbf{R},\mathbf{b}} \sum_{i=1}^{l_1}\sum_{j=1}^{l_2}\left(\mathbf{R}\mathbf{x}_i^{(1)} + \mathbf{b} - \mathbf{x}_j^{(2)}\right)^2 w_{i,j} \qquad (1)$$

where $w_{i,j}$ denotes the correspondence between every pair of $\mathbf{x}_i^{(1)}$ and $\mathbf{x}_j^{(2)}$. In ICP $w_{i,j} \in \{0, 1\}$, and (1) is solved by iteratively updating $w_{i,j}$ in a winner-take-all manner under a shortest-distance criterion and solving a least-squares problem with respect to $\mathbf{R}$ and $\mathbf{b}$. In EM-ICP, $w_{i,j}$ is interpreted as the probability of the correspondence, so a one-way constraint ($w_{i,j} \in [0, 1]$, $\sum_j^{l_2} w_{i,j} = 1$) is implicitly imposed. In SoftAssign, a stricter two-way constraint ($w_{i,j} \in [0, 1]$, $\sum_j^{l_2} w_{i,j} = 1$, $\sum_i^{l_1} w_{i,j} = 1$) is introduced to enforce globally consistent point correspondences. Although GM does not model the correspondences explicitly, they can likewise be understood as an instance of (1) with Euclidean distance replaced by Mahalanobis distance, and an uniform prior of $w_{i,j} = \frac{1}{l_1 l_2}$ for each pair of i, j. In conclusion, so far 3D point cloud registration can be achieved either by explicitly modeling correspondences and laboriously updating them (EM-ICP and SoftAssign), or by making some fragile correspondence assumptions to simplify the optimization procedure (ICP and GM). In addition, all these methods share the same computational complexity of $O(n^2)$ [1], which will be a heavy computational burden if the number of points n is relatively large. Therefore, a registration solution that can both express realistic priors over point correspondence matches and can be computed in a simpler (possibly non-iterative) and cheaper (possibly non-quadratic time) way is highly desirable. The method proposed in this paper fulfills both demands. Instead of doing point-wise correspondence search and computing in 3D space, our method first maps all points to a higher-dimensional (reproducing kernel Hilbert) feature space using kernel methods. The optimal transformation in feature space is then found by aligning Gaussians that approximate the two mapped point clouds. To project back to the 3D space, an objective function is constructed based on the fact that the transformed mapped points should be consistent with mapped transformed points. Finally, an $SE(3)$ on-manifold optimization scheme is exploited to provide an elegant and efficient gradient-type algorithm for registration.

Compared to previous registration methods, the strength of our method can be summarized in four points: (1) Although our algorithm was not developed on the basis of point correspondences, the form of its objective function (section III-A) suggests that correspondences are implicitly derived and to large degree it is consistent with the correct matches; (2) The experimental results (section IV) show that our method can work accurately and robustly in various challenging cases (large motion, outlier points); (3) Our method is much more efficient than other state-of-the-art methods, actually the computation complexity is $O(n \log n)$; (4) By using Kernel PCA and $SE(3)$ on-manifold optimization, the algorithm can be used as general point cloud registration framework with high flexibility and extensibility to any dimension.

II. RIGID TRANSFORMATION IN HILBERT SPACE

Intuitively, a straightforward way to align point clouds without point-wise correspondences is to first probabilistically fit each point cloud to a single Gaussian distribution in $\mathbb{R}^3$ and then align their means (translation) and covariances (rotation). However, the modeling ability of one single Gaussian in 3D space is too limited to capture the 3D point distribution of real-world objects, i.e. the mean and covariance in $\mathbb{R}^3$ are very sensitive to outliers. Inspired by kernel methods developed for set-format data [6], a 3D point clouds can be implicitly mapped to a much higher-dimensional Hilbert feature space, where a single Gaussian can fit well (Fig. 1) and hence yields higher tolerance to 3D disturbance in the original point cloud (e.g. outliers or non-rigid transformation). In addition, by applying kernel PCA, the eigenvectors of covariances can be efficiently computed and aligned without explicit computation in feature space.

[1]the complexity of ICP is $O(n \log n)$ if K-d trees are used for searching for the nearest neighbour

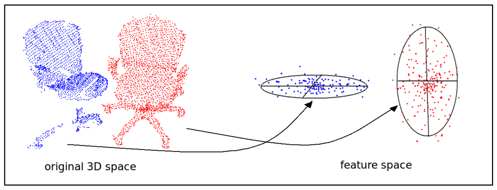


Figure 1. Mapping point clouds from 3D space to an infinite-dimensional Hilbert space, where a single Gaussian is sufficient to model distributions of complex shape.

A. Probabilistic modeling in Hilbert space

Inspired by kernel methods that have been widely used in machine learning, in order to map all points in a point cloud $\mathbf{M} = \{\mathbf{x}_i\}_{i=1}^l$ to a higher, possibly infinite-dimensional feature space, we can define a kernel function on 3D points $K(\mathbf{x}_i, \mathbf{x}_j)$. Then, a feature map can be implicitly induced by satisfying

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \tag{2}$$

where ϕ is the corresponding feature map: $\mathbb{R}^3 \to \mathcal{H}$, and $\mathcal{H}$ is referred to as the reproducing Hilbert feature space. Since the structure of $\mathbf{M}$ can be far too complicated in $\mathbb{R}^3$, to ensure that one single Gaussian is capable of modeling the distribution of $\{\phi(\mathbf{x}_i)\}_{i=1}^l$ in $\mathcal{H}$, in this paper we select the kernel function as the radial basis function (RBF)

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp \frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \tag{3}$$

because the induced feature map is a scaled Gaussian probability density function (PDF),

$$\phi(\mathbf{x}_i) = f(\xi | \mathbf{x}_i) = \exp \frac{-\|\xi - \mathbf{x}_i\|^2}{2\sigma^2}, \tag{4}$$

i.e., $\phi(\cdot)$ corresponds to an infinite-dimensional feature map.

With all points mapped into feature space, a Gaussian (mean and covariance) in $\mathcal{H}$ can be easily fitted by using maximum likelihood estimation (MLE):

$$\boldsymbol{\mu}_{\mathcal{H}} = \frac{1}{l} \sum_{i=1}^{l} \phi(\mathbf{x}_i) = \frac{1}{l} \phi(\mathbf{M})^\top \mathbf{1}_l \tag{5}$$

$$\boldsymbol{\Sigma}_{\mathcal{H}} = \frac{1}{l} \sum_{i=1}^{l} (\phi(\mathbf{x}_i) - \boldsymbol{\mu}_{\mathcal{H}}) (\phi(\mathbf{x}_i) - \boldsymbol{\mu}_{\mathcal{H}})^\top \tag{6}$$

where $\phi(\mathbf{M})^\top = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \cdots, \phi(\mathbf{x}_l)]$ and $\mathbf{1}_l$ is an l-dimensional vector with all elements equal to 1.

B. Kernel PCA

To achieve the alignment between two covariances, their eigenvectors should be computed first. However, this computation is non-trivial in feature space. Kernel principal component analysis (KPCA) [7] is a technique developed to compute eigenvectors in feature space without explicit computation in $\mathcal{H}$. Here we briefly review the procedure of KPCA with its application to 3D point clouds.

Assuming all points are already centered in feature space, the covariance $\mathbf{\Sigma}_{\mathcal{H}}$ of the Gaussian can be estimated as

$$\mathbf{\Sigma}_{\mathcal{H}} = \frac{1}{l}\sum_{i=1}^{l}\phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top \tag{7}$$

which is a symmetric bilinear form on $\mathcal{H}$. Analogous to the symmetric covariance matrices in the finite-dimensional case, its nonzero eigenvalue λ_k and corresponding eigenvector $\mathbf{u}_k$ should satisfy

$$\lambda_k\mathbf{u}_k = \mathbf{\Sigma}_{\mathcal{H}}\mathbf{u}_k. \tag{8}$$

By substituting (7) into (8), we have

$$\mathbf{u}_k = \frac{1}{\lambda_k}\mathbf{\Sigma}_{\mathcal{H}}\mathbf{u}_k = \sum_{i=1}^{l}\alpha_i^k\phi(\mathbf{x}_i) \tag{9}$$

where $\alpha_i^k = \frac{\phi(\mathbf{x}_i)^\top\mathbf{u}_k}{\lambda_k l}$. Therefore, all eigenvectors $\mathbf{u}_k$ with $\lambda_k \neq 0$ must lie in the span of $\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \dots, \phi(\mathbf{x}_l)$, and (9) is referred to as the dual form of $\mathbf{u}_k$. By left-multiplying $\sum_{j=1}^{l}\phi(\mathbf{x}_j)^\top$ on both sides of equation (8), we have

$$\begin{aligned} \sum_{j=1}^{l}\phi(\mathbf{x}_j)^\top\lambda_k\mathbf{u}_k &= \sum_{j=1}^{l}\phi(\mathbf{x}_j)^\top\mathbf{\Sigma}_{\mathcal{H}}\mathbf{u}_k \\ \Leftrightarrow\quad \lambda_k\sum_{i,j=1}^{l}\alpha_i^k K(\mathbf{x}_i,\mathbf{x}_j) &= \frac{1}{l}\sum_{i,j=1}^{l}\alpha_i^k K(\mathbf{x}_i,\mathbf{x}_j)^2 \\ \Leftrightarrow\quad l\lambda_k\boldsymbol{\alpha}^k &= \mathbf{K}\boldsymbol{\alpha}^k \end{aligned} \tag{10}$$

where $\mathbf{K}$ is an $l \times l$ kernel matrix with $\mathbf{K}_{ij} = K(x_i, x_j)$, $\boldsymbol{\alpha}^k = (\alpha_1^k, \alpha_2^k, \dots, \alpha_l^k)^\top$. It can be seen that $\{\boldsymbol{\alpha}^k, \eta^k = l\lambda_k\}$ is actually an eigenvalue-eigenvector pair of matrix $\mathbf{K}$. Therefore, by using dual forms of eigenvectors, the eigenvector decomposition of $\mathbf{\Sigma}_{\mathcal{H}}$ can be transformed to the decomposition of the finite matrix $\mathbf{K}$. In addition, because all $\mathbf{u}_k$ should be unit vectors:

$$1 = \mathbf{u}_k^\top\mathbf{u}_k = \langle\boldsymbol{\alpha}^k, \mathbf{K}\boldsymbol{\alpha}^k\rangle = \eta^k\boldsymbol{\alpha}^{k\top}\boldsymbol{\alpha}^k \tag{11}$$

the $\boldsymbol{\alpha}^k$ should be normalized as:

$$\boldsymbol{\alpha}^k \leftarrow \frac{\boldsymbol{\alpha}^k}{\sqrt{\eta^k}} \tag{12}$$

However, though the point cloud can be easily centered in 3D space, it does not mean it is also centered in feature space. By replacing $\phi(\mathbf{x}_i)$ with $\tilde{\phi}(\mathbf{x}_i) = \phi(\mathbf{x}_i) - \boldsymbol{\mu}$, the corresponding kernel matrix $\tilde{\mathbf{K}}$ is

$$\tilde{\mathbf{K}} = \mathbf{K} - \frac{1}{l}\mathbf{EK} - \frac{1}{l}\mathbf{KE} + \frac{1}{l^2}\mathbf{EKE} \tag{13}$$

where $\mathbf{E}$ denotes an $l \times l$ matrix with all entries equal to 1. After similar eigenvector decomposition (10) and normalization (12) steps, we obtain eigenvectors

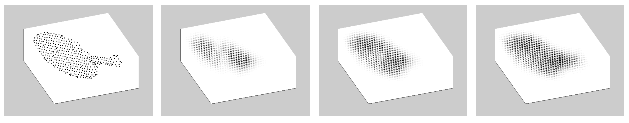


Figure 2. (a) A point cloud of table tennis racket; (b–d) reconstruction using the first 1–3 principal components. For each point in the bounding-box volume, the darkness is proportional to the density of the Gaussian in the feature space $\mathcal{H}$.

$$\tilde{\mathbf{u}}_k = \sum_{i=1}^{l}\tilde{\alpha}_i^k(\phi(\mathbf{x}_i) - \boldsymbol{\mu}) = \phi(\mathbf{M})^\top\underbrace{(\mathbf{I}_l - \frac{1}{l}\mathbf{E})}_{\mathbf{I^E}}\tilde{\boldsymbol{\alpha}}^k \tag{14}$$

where $\mathbf{I}_l$ is an $l \times l$ identity matrix and $\tilde{\boldsymbol{\alpha}}$ is the kth eigenvector of the matrix $\tilde{\mathbf{K}}$.

As analyzed in [6], it is misleading and wasteful to use full covariances, so only a small number of eigenvectors are sufficient to capture the structural property of the covariance $\mathbf{\Sigma}_{\mathcal{H}}$. Fig. 2 displays an example of a table tennis racket point cloud. Its Gaussian distribution in the feature space $\mathcal{H}$ can be well reconstructed using only 3 of its principal components associated with top largest eigenvalues.

C. Alignment of Gaussians

Assume the task is to align a point cloud $\mathbf{M}_1 = \{\mathbf{x}_i^{(1)}\}_{i=1}^{l_1}$ with $\mathbf{M}_2 = \{\mathbf{x}_j^{(2)}\}_{j=1}^{l_2}$, instead of computing the optimal alignment in 3D space directly, we can alternatively first align them in feature space, and then project them back to $\mathbb{R}^3$ (section III). With the modeling procedure above applied on $\mathbf{M}_1$ and $\mathbf{M}_2$, the alignment of two point clouds in feature space corresponds to aligning two Gaussians. In this paper, we assume $\mathbf{D}$ eigenvectors are computed for the covariance of each point cloud: $\tilde{\mathbf{U}}_1 = [\tilde{\mathbf{u}}_1^1, \dots, \tilde{\mathbf{u}}_1^k, \dots, \tilde{\mathbf{u}}_1^{\mathbf{D}}]$, $\tilde{\mathbf{U}}_2 = [\tilde{\mathbf{u}}_2^1, \dots, \tilde{\mathbf{u}}_2^k, \dots, \tilde{\mathbf{u}}_2^{\mathbf{D}}]$. Therefore, the rotation in feature space $\mathbf{R}_{\mathcal{H}}$ is estimated by simultaneously aligning $\mathbf{D}$ pairs of corresponding eigenvectors: $\tilde{\mathbf{U}}_2 = \mathbf{R}_{\mathcal{H}}\tilde{\mathbf{U}}_1$. Because different eigenvectors of each point cloud are orthogonal to each other, based on the computation result in (14), it is easy to derive:

$$\begin{aligned} \mathbf{R}_{\mathcal{H}} &= \tilde{\mathbf{U}}_2\tilde{\mathbf{U}}_1^\top \\ &= \phi(\mathbf{M}_2)^\top\underbrace{\mathbf{I}_2^{\mathbf{E}}\left(\sum_{k=1}^{\mathbf{D}}\tilde{\boldsymbol{\alpha}}_2^k\tilde{\boldsymbol{\alpha}}_1^{k\top}\right)\mathbf{I}_1^{\mathbf{E}}}_{\mathbf{\Theta}_{\boldsymbol{\alpha}}}\phi(\mathbf{M}_1) \end{aligned} \tag{15}$$

Since the rotation (15) can be applied only if $\mathbf{M}_1$ has already been centered in feature space, to fully align two Gaussians, the translation in feature space $\mathbf{b}_{\mathcal{H}}$ obviously should equal the mean of the Gaussian that models $\mathbf{M}_2$ in feature space:

$$\mathbf{b}_{\mathcal{H}} = \boldsymbol{\mu}_{\mathcal{H}}^{(2)} = \frac{1}{l_2}\phi(\mathbf{M}_2)^\top\mathbf{1}_{l_2} \tag{16}$$

$$\begin{tikzcd}
\mathbf{x}_t^{(1)} \arrow[rr, "\phi"] \arrow[d, "{\{\mathbf{R}, \mathbf{b}\}}"'] & & \phi(\mathbf{x}_t^{(1)}) \arrow[d, "{\{\mathbf{R}_{\mathcal{H}}, \mathbf{b}_{\mathcal{H}}\}}"] \\
\mathbf{R}\mathbf{x}_t^{(1)} + \mathbf{b} \arrow[r, "\phi"'] & \underbrace{\phi(\mathbf{R}\mathbf{x}_t^{(1)} + \mathbf{b})}_{\mathbf{\Phi}_t} \sim & \underbrace{\mathbf{R}_{\mathcal{H}}\tilde{\phi}(\mathbf{x}_t^{(1)}) + \mathbf{b}_{\mathcal{H}}}_{\mathbf{\Psi}_t}
\end{tikzcd}$$

Figure 3. The consistency error is defined as the discrepancy between $\phi(\mathbf{R}x_t + \mathbf{b})$ and $\mathbf{R}_{\mathcal{H}}\tilde{\phi}(x_t) + \mathbf{b}_{\mathcal{H}}$.

Now we can align two Gaussians in feature space with $\{\mathbf{R}_{\mathcal{H}}, \mathbf{b}_{\mathcal{H}}\}$ computed in (15) and (16). However, due to the infinite-dimensional feature map defined in (4), there still exist two obstacles to be overcome: First, (15) and (16) cannot be computed in an analytic form; secondly, there is no trivial way to map $\{\mathbf{R}_{\mathcal{H}}, \mathbf{b}_{\mathcal{H}}\}$ back to 3D space. Fortunately, by designing a consistency error (section III-A), these two issues can surprisingly be solved simultaneously in a very smooth and elegant manner.

III. CARTESIAN POINT CLOUD ALIGNMENT

In this section we will project the rotation and translation in the feature space $\mathcal{H}$ back to 3D space by minimizing a specifically-designed consistency error (Fig. 3). It turns out that the final objective function can be constructed and solved without explicit computation in feature space. In addition, further connections with other registration methods will be exposed by discovering the hidden commonality among their objective functions. To enhance the generality of the proposed algorithm, an $SE(3)$ on-manifold optimization scheme is employed to search for the optimal transformation.

A. Consistency Error

Instead of tediously finding the inverse function $\phi^{-1}(\cdot)$ corresponding to the definition in (4) and applying it to $\{\mathbf{R}_{\mathcal{H}}, \mathbf{b}_{\mathcal{H}}\}$ to map them back to 3D space $\{\mathbf{R}, \mathbf{b}\}$, here we define a novel consistency error between $\phi(\mathbf{R}x_t + \mathbf{b})$ and $\mathbf{R}_{\mathcal{H}}\left(\phi(x_t) - \boldsymbol{\mu}_1\right) + \mathbf{b}_{\mathcal{H}}$ based on the fact that mapping after transformation should be consistent with transformation after mapping (Fig. 3). Therefore, we can find the optimal rotation and translation in 3D space by minimizing the average consistency error:

$$\{\mathbf{R}^*, \mathbf{b}^*\} = \arg\min_{\mathbf{R},\mathbf{b}} \frac{1}{l_1} \sum_{t=1}^{l_1} \|\mathbf{\Psi}_t - \mathbf{\Phi}_t\|^2 \tag{17}$$

Because $\|\mathbf{\Phi}(\boldsymbol{x})\|^2$ is the integration over the square of a Gaussian, which preserves constant under any translation $\mathbf{b}$ and rotation $\mathbf{R}$, and $\mathbf{\Psi}_t$ is fixed, by substituting (15) and (16) into (17), we have

$$\begin{aligned}\{\mathbf{R}^*, \mathbf{b}^*\} &= \arg\max_{\mathbf{R},\mathbf{b}} \frac{1}{l_1} \sum_{t=1}^{l_1} \mathbf{\Psi}_t^\top \mathbf{\Phi}_t \\ &= \arg\max_{\mathbf{R},\mathbf{b}} \underbrace{\frac{1}{l_1} \sum_{t=1}^{l_1} K(\mathbf{R}\mathbf{x}_t^{(1)} + \mathbf{b}, \mathbf{M}_2)^\top \boldsymbol{\rho}_t}_{\mathbf{O}}\end{aligned} \tag{18}$$

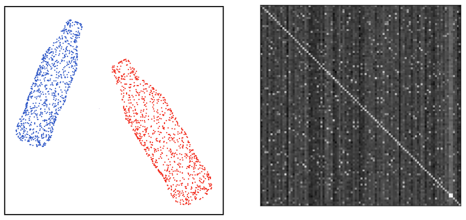


Figure 4. (a) Two identical point clouds with exactly the same point permutation. (b) Visualization of ρ_{ti} computed for all pairs of points.

$$\boldsymbol{\rho}_t = \mathbf{\Theta}_{\boldsymbol{\alpha}} \left(K(\mathbf{x}_t^{(1)}, \mathbf{M}_1) - \frac{1}{l_1}\mathbf{K}_1 \mathbf{1}_{l_1} \right) + \frac{1}{l_2}\mathbf{1}_{l_2} \tag{19}$$

where $K(\mathbf{R}\mathbf{x}_t^{(1)} + \mathbf{b}, \mathbf{M}_2)$ is an l_2-dimensional vector with $K(\mathbf{R}\mathbf{x}_t^{(1)} + \mathbf{b}, \mathbf{M}_2)_i = K(\mathbf{R}\mathbf{x}_t^{(1)} + \mathbf{b}, \mathbf{x}_i^{(2)})$, and $K(\mathbf{x}_t^{(1)}, \mathbf{M}_1)$ is an l_1- dimensional vector with $K(\mathbf{x}_t^{(1)}, \mathbf{M}_1)_j = K(\mathbf{x}_t^{(1)}, \mathbf{x}_j^{(1)})$. It can be seen that by employing the kernel trick (2), we can elegantly avoid computation in the feature space $\mathcal{H}$ in both (18) and (19).

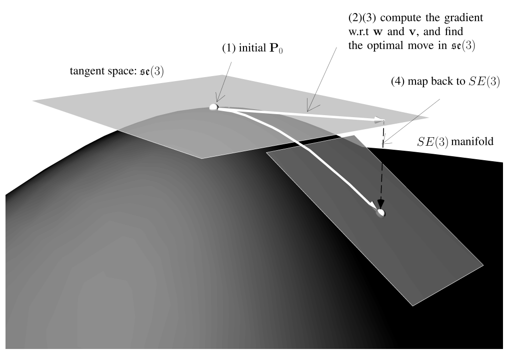
B. Relation to Other Approaches

As analyzed in section I, most existing registration methods can be unified to a general objective function (1) with different correspondence assumptions or iterative update strategies. The objective function (18) can be easily extended as follows:

$$\{\mathbf{R}^*, \mathbf{b}^*\} = \arg\min_{\mathbf{R},\mathbf{b}} \sum_{t=1}^{l_1} \sum_{i=1}^{l_2} -K(\mathbf{R}\mathbf{x}_t^{(1)} + \mathbf{b}, \mathbf{x}_i^{(2)})\rho_{t,i} \tag{20}$$

By considering $-K(\cdot,\cdot)$ as an exponential distance and replacing $\rho_{t,i}$ with $w_{t,i}$, it turns out that surprisingly our method (20) is also a special case of (1), although we arrive there from a completely different starting point. This suggests that $\rho_{t,i}$ somehow implicitly encodes the correspondence likelihood between $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_i^{(2)}$ as well. To verify this argument experimentally, in Fig 4(a) there are two identical point clouds with exactly the same point permutation. We compute ρ_{ti} for all pairs of points in Fig 4(b). It can be seen that Fig 4(b) shows a very evident diagonal pattern with uniformly distributed noise, which is the reflection of our prior knowledge. However, different from most of other approaches, we do not model $w_{t,i}$ explicitly or update them iteratively. Instead, $\rho_{t,i}$ is derived from eigenvector alignment in feature space and only need to be computed once.

There is another way our method is related to Gaussian mixtures. By relaxing the non-negative coefficient constraint in the definition of Gaussian mixtures, each eigenvector in (14) can be considered as a pseudo Gaussian mixture with $\phi(\cdot)$ defined as in (4). In this way, instead of aligning two Gaussian mixtures in 3D space, what our method is actually doing is to simultaneously align $\mathbf{D}$ pairs of Gaussian



(2)(3) compute the gradient w.r.t $\mathbf{w}$ and $\mathbf{v}$, and find the optimal move in $\mathfrak{se}(3)$
(1) initial $\mathbf{P}_0$
tangent space: $\mathfrak{se}(3)$
(4) map back to $SE(3)$
$SE(3)$ manifold

Figure 5. The $SE(3)$ manifold and its optimization scheme: (1) start from a rotation matrix $\mathbf{P}_0$; (2) use equation (26) as the local parametrization of the manifold at point $\mathbf{P}_0$, and compute the gradient with respect to $\{\mathbf{w}, \mathbf{v}\}$; (3) compute the best move in $\mathfrak{se}(3)$ by mapping the update of $\{\mathbf{w}, \mathbf{v}\}$; (4) map back to $SE(3)$: $\mathbf{P}_1 \leftarrow \exp(\Lambda)\mathbf{P}_0$; (5) repeat step (2)(3)(4) until convergence

mixtures in feature space, and then implicitly maps back to original 3D space.

C. SE(3) on-manifold optimization

When solving the optimization problem (18), the orthogonality constraint of the rotation matrix $\mathbf{R}$ must be taken into account: $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$. This constraint is a common obstacle in various rotation-related optimization problems, which drove many researchers to alternatives such as unit quaternions [5] or dual quaternion number [8]. However, although all these methods can work satisfactorily for 3D rotation, they are difficult to be extend to higher dimensions. Although we only focus 3D point cloud registration here, to make our algorithm more general, here we employ an $SE(3)$ on-manifold optimization scheme [9], which can be easily adapted to rotation matrices in any dimension. Another virtue of $SE(3)$ on-manifold optimization is that combined with gradient computing, an elegant optimizer can be developed based on its associated Lie algebra to circumvent the orthogonality constraint.

Each rotation and translation in 3D space $\{\mathbf{R}, \mathbf{b}\}$ can be jointly treated as a Euclidean transformation $\mathbf{P}$ in $\mathbb{R}^3$ by using homogeneous coordinates. From now on, $\mathbf{x}$ is used to denote homogeneous coordinate, and $\overline{\mathbf{x}}$ for the original one:

$$\mathbf{x}^\top = [\overline{\mathbf{x}}^\top, 1] \tag{21}$$

and correspondingly,

$$\mathbf{P} = \begin{bmatrix} \mathbf{R} & \mathbf{b} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{22}$$

One specific $\mathbf{P}$ is actually an element of Lie group $SE(3)$ (Special Euclidean Group), which is a smooth manifold embedded in $\mathbb{R}^3$. Intuitively, the $SE(3)$ manifold can be considered as a topological space wherein all points are 4×4 Euclidean transformation matrices, and at each point, there exists a tangent space Λ, which happens to be its associated Lie algebra $\mathfrak{se}(3)$. The mathematical connection between $SE(3)$ and $\mathfrak{se}(3)$ is

$$\mathfrak{se}(3) \to SE(3) : \mathbf{P} = \exp(\Lambda) \tag{23}$$

where $\exp(\cdot)$ denotes the exponential map. The tangent space $\mathfrak{se}(3)$ can be considered as a linearization of the $SE(3)$ manifold within the infinitesimally small vicinity of certain point $\mathbf{P}_0$, so inversely, the exponential map works as a 'de-linearization'. All concepts described above are illustrated in Fig. 5. The Lie algebra $\mathfrak{se}(3)$ is a collection of matrices of the form

$$\Lambda = \begin{bmatrix} J(\mathbf{w}) & \mathbf{v} \\ \mathbf{0}_{1\times3} & 0 \end{bmatrix} \tag{24}$$

where $J(\mathbf{w})$ is an skew-symmetric matrix, which can be constructed from a 3D vector $\mathbf{w}$ with a skew operator $J(\cdot)$:

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \to J(\mathbf{w}) = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix} \tag{25}$$

and $\mathbf{v}$ is an usual 3D vector. Therefore, Therefore, when $\exp(\Lambda) \to \mathbf{I}_3$ (e.g. computing gradient), we can establish a straightforward map from $\mathbb{R}^6$ to the local neighboring region of $\mathbf{P}_0$ on manifold as

$$\mathbf{P} = \exp\left(\begin{bmatrix} J(\mathbf{w}) & \mathbf{v} \\ \mathbf{0}_{1\times3} & 0 \end{bmatrix}\right) \cdot \mathbf{P}_0 \tag{26}$$

Last but not least, combination with a gradient-type method yields an final optimization procedure for $SE(3)$ parameters. By using (26), we can see that when computing the gradient with respect to $\mathbf{w}$ and $\mathbf{v}$, the orthogonality constraint will be avoided, and therefore, the constrained optimization problem in $SE(3)$ (18) is naturally and smoothly transformed to a much simpler, unconstrained problem in $\mathbb{R}^6$. Meanwhile, different from conventional gradient methods, instead of computing gradient and updating within the same space, in $SE(3)$ on-manifold optimization, after every update of $\{\mathbf{w}, \mathbf{v}\}$, it need to be mapped back to $SE(3)$, and subsequently the gradient is computed with the local parametrization of the corresponding neighboring region. The whole procedure of $SE(3)$ on-manifold optimization scheme is illustrated in Fig. 5.

D. Reduction of Computational Complexity

If we reexamine the objective function $\mathbf{O}$ (18), an interesting property can be leveraged to significantly reduce the computation complexity:

$$\begin{aligned} & \langle \mathbf{\Phi}_t, \mathbf{\Psi}_t \rangle \\ = \quad & \phi(\overline{\mathbf{P}\mathbf{x}_t^{(1)}})^\top \left(\sum_{k=1}^{\mathbf{D}} \tilde{\mathbf{u}}_2^k \tilde{\mathbf{u}}_1^{k\top} \left(\overline{\phi(\mathbf{x}_t^{(1)}) - \boldsymbol{\mu}_1} \right) + \boldsymbol{\mu}_2 \right) \\ = \quad & \sum_{k=1}^{\mathbf{D}} \langle \tilde{\boldsymbol{u}}_2^k, \phi(\overline{\mathbf{P}\mathbf{x}_t^{(1)}}) \rangle \langle \tilde{\boldsymbol{u}}_1^k, \overline{\phi(\mathbf{x}_t^{(1)}) - \boldsymbol{\mu}_1} \rangle + \langle \boldsymbol{\mu}_2, \phi(\overline{\mathbf{P}\mathbf{x}_t^{(1)}}) \rangle \\ = \quad & \sum_{k=1}^{\mathbf{D}} \langle \tilde{\boldsymbol{u}}_2^k, \phi(\overline{\mathbf{P}\mathbf{x}_t^{(1)}}) \rangle \langle \tilde{\boldsymbol{u}}_2^k, \mathbf{R}_{\mathcal{H}} \overline{\phi(\mathbf{x}_t^{(1)}) - \boldsymbol{\mu}_1} \rangle + \langle \boldsymbol{\mu}_2, \phi(\overline{\mathbf{P}\mathbf{x}_t^{(1)}}) \rangle \end{aligned} \tag{27}$$

where we can see that $\overline{\phi(\mathbf{P}\mathbf{x}_t^{(1)})}$ and $\mathbf{R}_{\mathcal{H}}\overline{\phi(\mathbf{x}_t^{(1)})-\boldsymbol{\mu}_1}$ are projected onto $\mathbf{D}$ eigenvectors $\left\{\tilde{\boldsymbol{u}}_2^k\right\}_{k=1}^{\mathbf{D}}$ respectively, and an additional projection of $\underline{\phi(\mathbf{P}\mathbf{x}_t^{(1)})}$ onto $\boldsymbol{\mu}_2$. Therefore, the computation of the objective function is actually done in a space spanned by $\mathbf{D}$ eigenvectors $\left\{\tilde{\boldsymbol{u}}_2^k\right\}_{k=1}^{\mathbf{D}}$ and one $\boldsymbol{\mu}_2$, which is a subspace of $\mathcal{H}$. We denote this subspace by $\mathcal{S}$. The feature map of each point $\phi(\mathbf{x}_i)$ can be projected onto $\mathcal{S}$ in the following way:

$$\mathcal{S}\left(\phi(\mathbf{x}_i)\right)=\sum_{k}^{\mathbf{D}+1}\beta_{i,k}\mathbf{r}_k \tag{28}$$

where $\mathbf{r}_k$ are referred to as $\mathbf{D}+1$ reference vectors in $\mathcal{S}$, and $\beta_{i,k}$ are the corresponding coefficients used to express $\mathcal{S}\left(\phi(\mathbf{x}_i)\right)$. In other words, in $\mathcal{S}$, only $\mathbf{D}+1$ reference vectors, of which $\mathbf{D}$ should be linearly independent, can represent any $\mathcal{S}(\phi(\mathbf{x}_i))$. Therefore, to ensure that $\boldsymbol{\Phi}_t$ and $\boldsymbol{\Psi}_t$ are consistent with each other for all points $\mathbf{x}_t^{(1)}$ in $\mathbf{M}_1$, we only need to align $\mathbf{D}+1$ predefined reference vectors. In practice, we can randomly select $\mathbf{D}+1$ $\mathcal{S}(\phi(\mathbf{x}_i))$ because they are very likely to be linear independent in $\mathcal{S}$. Thus, the objective function can be simplified to

$$\{\mathbf{R}^*,\mathbf{b}^*\}=\arg\max_{\mathbf{R},\mathbf{b}}\frac{1}{\mathbf{D}+1}\sum_{t=1}^{\mathbf{D}+1}K(\mathbf{R}\mathbf{x}_{\mathbf{S}_t}^{(1)}+\mathbf{b},M_2)^\top\boldsymbol{\rho}_t \tag{29}$$

where $\mathbf{S}$ denotes the randomly selected subset of $\mathbf{M}_1$. To practically apply $SE(3)$ on-manifold optimization on the objective function $\mathbf{O}$ (29), we compute the derivatives of (29) w.r.t. $\mathbf{w}$ and $\mathbf{v}$ as follows:

$$\frac{d\mathbf{O}}{d\left[\mathbf{w}^\top,\mathbf{v}^\top\right]^\top}=\frac{1}{\mathbf{D}+1}\sum_{t=1}^{\mathbf{D}+1}\left(\frac{d\mathbf{O}}{d\mathbf{P}\mathbf{x}_{\mathbf{S}_t}^{(1)}}\frac{d\mathbf{P}\mathbf{x}_{\mathbf{S}_t}^{(1)}}{d\left[\mathbf{w}^\top,\mathbf{v}^\top\right]^\top}\right) \tag{30}$$

where

$$\frac{d\mathbf{O}}{d\mathbf{P}\mathbf{x}_t^{(1)}}=\sum_{j=1}^{l_2}\rho_{\mathbf{S}_t,j}K\left(\mathbf{P}\mathbf{x}_{\mathbf{S}_t}^{(1)},\mathbf{x}_j^{(2)}\right)\frac{1}{\sigma^2}\left(\mathbf{x}_j^{(2)}-\mathbf{P}\mathbf{x}_{\mathbf{S}_t}^{(1)}\right)^\top \tag{31}$$

$$\frac{\partial\mathbf{P}\mathbf{x}_{\mathbf{S}_t}^{(1)}}{\partial\mathbf{w}}=\frac{\partial\exp(\Lambda)\mathbf{P}_0\mathbf{x}_{\mathbf{S}_t}^{(1)}}{\partial\mathbf{w}}=\left[J(\overline{\mathbf{P}_0\mathbf{x}_{\mathbf{S}_t}^{(1)}}),\mathbf{0}_{3\times1}\right]^\top \tag{32}$$

$$\frac{\partial\mathbf{P}\mathbf{x}_{\mathbf{S}_t}^{(1)}}{\partial\mathbf{v}}=\frac{\partial\exp(\Lambda)\mathbf{P}_0\mathbf{x}_t^{(1)}}{\partial\mathbf{v}}=\left[\mathbf{I}_3,\mathbf{0}_{3\times1}\right]^\top \tag{33}$$

IV. EXPERIMENTS

A. Implementation details

Since the computed eigenvectors (14) are of no direction, there could be $2^{\mathbf{D}}$ possible alignments for $\mathbf{D}$ eigenvectors in feature space. Fortunately, according to experiment, we found that $\mathbf{D}=3$ is actually enough to make sufficiently good alignment. Therefore, one has to compute all 8 possible alignments in feature space and project them back to $\mathbb{R}^3$, then the final optimal one is picked by checking the accumulated distances between every pair of corresponding points in two clouds, and we use shortest distance as the correspondence here. An outline of the proposed algorithm is given in Algorithm 1. In practice, to speed up the convergence of the algorithm, some sophisticated stepsize tricks can also be added. In addition, we also find that in Algorithm 1 computing eigenvectors (line 2) is the most time-consuming part, so in our implementation, fast-PCA [10] is employed to accelerate the computation.

Algorithm 1 3D Point Cloud Registration

```
Input:  $\mathbf{M}_1=\{\mathbf{x}_i^{(1)}\}_{i=1}^{l_1}$ and $\mathbf{M}_2=\{\mathbf{x}_j^{(2)}\}_{j=1}^{l_2}$, $\mathbf{x}\in\mathbb{R}^3$
Output: the optimal motion estimation $\mathbf{P}^*$ which can
        align $\mathbf{M}_1$ with $\mathbf{M}_2$
 1: construct two matrices $\tilde{\mathbf{K}}_1$ and $\tilde{\mathbf{K}}_2$ (13)
 2: compute eigenvalue-eigenvector pairs for $\tilde{\mathbf{K}}_1$ and $\tilde{\mathbf{K}}_2$:
    $\{\boldsymbol{\alpha}_{m,k},\eta_{m,k}\}\quad m=1,2$
 3: normalize eigenvectors (12)
 4: select $\mathbf{D}=3$ eigenvectors with largest eigenvalues for
    both $\mathbf{M}_1$ and $\mathbf{M}_2$
 5: randomly select a subset of $\mathbf{N}\geq\mathbf{D}+1$ size from $\mathbf{M}_1$
 6: set initial $\mathbf{P}_0$ randomly
 7: compute $\boldsymbol{\Theta}_{\boldsymbol{\alpha}}$ (15) with the subset
 8: while 1 do
 9:   compute the gradient $\nabla_{\mathbf{w}}$ and $\nabla_{\mathbf{w}}$ with current $\mathbf{P}_n$
      (30–33)
10:   if both $\nabla_{\mathbf{w}}$ and $\nabla_{\mathbf{v}}$ are small enough then
11:     return $\mathbf{P}_n$
12:   end if
13:   map the update of $\mathbf{w}$ and $\mathbf{v}$ back to $SE(3)$ (26)
14:   set $n\leftarrow n+1$
15: end while
16: repeat line 7–15 $2^{\mathbf{D}}$ times with different sign com-
    binations of eigenvectors, and select the final optimal
    $\mathbf{P}^*$ which yields the minimal accumulated distances
    between every pair of closest points in $\mathbf{P}_n\mathbf{M}_1$ and $\mathbf{M}_2$
```

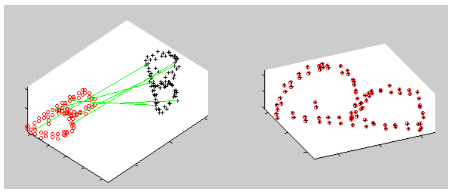
B. Qualitative Evaluation

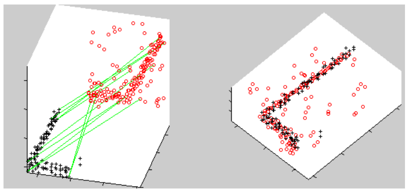
For the sake of visualization, we first test our algorithm on some toy point clouds to see how it work qualitatively. In Figure 6, some test examples on handwritten letters are displayed. It can be seen that in rather challenging circumstances, i.e. (1) the motion between two point clouds is arbitrarily large (Figure 6(a)), (2) a large portion of outliers are added (Figure 6(b)), (3) nonrigid transformation is applied (Figure 6(c)), the proposed algorithm can still discover roughly correct corresponding points [2] between two point clouds (green lines in Figure 6) and make qualitatively acceptable alignment.

C. Quantitative Evaluation

To obtain a more precise and convincing evaluation of the the proposed algorithm, KIT database [11] is used for

[2]the correspondence for point $\mathbf{x}_t^{(1)}$ is determined by finding the index $j^*=\arg\max_{j\in[1,l_2]}\rho_{tj}$


(a)


(b)

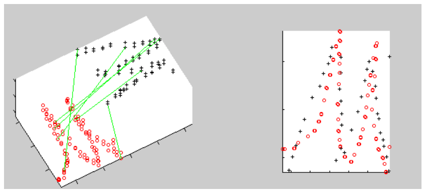

(c)

Figure 6. Test of the proposed algorithm in typical challenging circumstances for registration: (a) large motion; (b) outliers; (c) nonrigid transformation

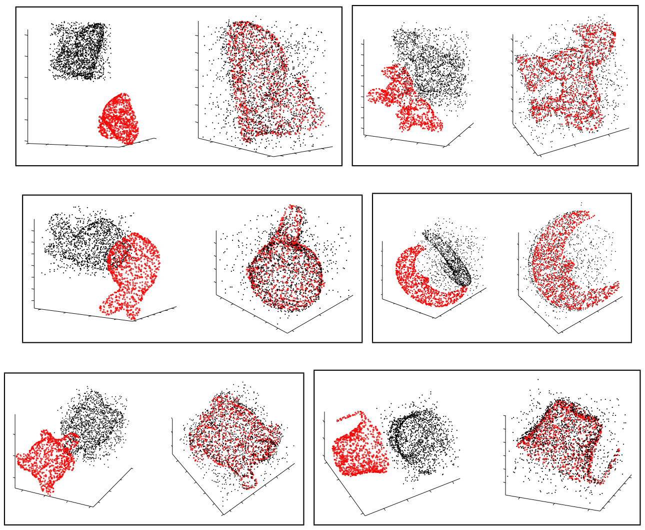
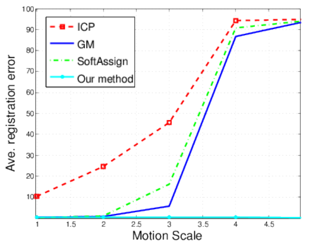


Figure 7. Some test results on KIT 3D object database

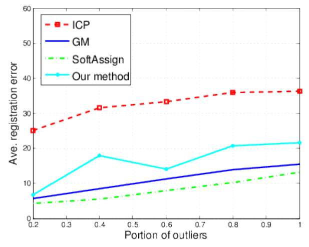
more intensive test (some test results can be seen in Figure 7). In addition, for quantitative comparison, ICP method, Gausssian Mixture(GM) and SoftAssign [3] are implemented as well. To ensure fairness, the same $SE(3)$ on-manifold optimization strategy is employed for their corresponding objective functions. Since the objects in KIT database is in triangulated mesh format, point clouds are generated by first sampling a triangle with the probability proportional to its area and then uniformly sampling a point from the selected triangle.

First, we test the robustness of four algorithms on different scales of motions. In our experiment, for motion scale i, the rotation angles of yaw, pitch and roll are $i \times [30°, 5°, 5°]$, and translations are $i \times [S_x, S_y, S_z]$, where $[S_x, S_y, S_z]$ are standard deviations of point clouds in three axes. Different motions are applied to the point cloud of each object (points cloud is sampled with size 1000) to generate a target point cloud to align with. Since we know the correspondence between the original and target point clouds, the error for each registration is computed as the average distance between every pair of corresponding points in two point

[3]the comparison in [12] has reported that SoftAssign and EM-ICP perform similarly, so we are not going to include EM-ICP in our experiment



(a)



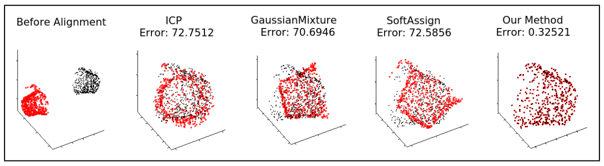
(b)

Figure 8. Test of four registration algorithm on (a) different scales of motions; (b) different portion of outliers added.

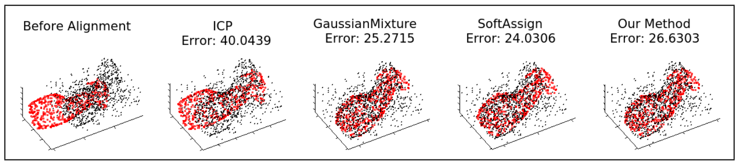
clouds. The test result is plotted in Figure 8(a). We can see that four algorithms can work similarly well for small motions (scale $i = 1$). However, as i increases, the accuracy of ICP, GM and SoftAssign decrease, although GM and SoftAssign are much more robust than ICP for intermediate motions (scale $i = [2, 3]$). Our method, by contrast, performs consistently well for all scales of motions. A test example is displayed in Figure 9(a), from which we can see that the instability of ICP, GM and SoftAssign stems from the fact that they are likely to to stuck into local optimum when the motion is large (although the local optimum can be avoided by setting up many intial poses, it would take more time to guarantee that the global optimum is found).

Secondly, we test the robustness of four algorithms by adding different portion (the percentage of point cloud size) of outliers which are randomly sampled within the space around objects. The generated outliers are concatenated into the original point clouds, so the correspondence is still available and the registration error is computed in the same way as in the motion experiment. To avoid the effect of large motion, a relatively small motion (motion scale $i = 1$) is applied to all point clouds. The test result is plotted in Figure 8(b). We can see that SoftAssign is most stable for the case in which outliers are presented, GM and our method are slightly worse, and ICP is very sensitive to outlier even when the portion is small. A test example is displayed in Figure 9(b), from which we can see that except ICP, the result of other three algorithms are acceptable.

Last but never least, efficiency is a significant strength of our method, which enables it can be used for real



(a)



(b)

Figure 9. Comparison of four registration algorithms: (a) motion scale $i = 5$; (b) outlier portion$= 0.8$.

	100	200	500	1000	2000
Our method	1.172	1.489	2.162	5.126	21.165
ICP [1]	0.012	0.023	0.051	0.154	0.469
GaussianMixtures [5]	1.859	3.998	15.245	43.570	172.4
SoftAssign [3]	2.059	4.801	83.925	592.1	3812

Table I
AVERAGE EXECUTION TIME (SECONDS)

time applications. As we can see in Algorithm 1, after computing eigenvectors for kernel matrices, the complexity of computing optimal motion is linear to the size of points n. Since the complexity of fast PCA is $O(n \log n)$ [10], the overall complexity of Algorithm 1 is $O(n \log n)$. To compare the efficiency, all four algorithms are implemented in Matlab and run on the same hardware platform (usual i7 intel core laptop). Point clouds of all objects are generated with 5 different sizes (100, 200, 500, 1000, 2000), on which four algorithms are tested respectively. For each point cloud, a randomly generated motion is applied and random portion of outliers are added (to get an approximate average). Note that in this experiment we are only concerned about the running time, so the algorithms will stop when they converge even if the registration is bad. The average execution time (in seconds) of four algorithms on five set of point clouds are presented in Table I. We can see that the complexity of our algorithm is the same as ICP $O(n \log n)$, and it is much faster than SoftAssign and Gausssian Mixtures(GM) with complexity $O(n^2)$ (however, SoftAssign is usually more expensive than GM because it needs to iteratively update correspondence matrix).

CONCLUSION

We introduced a novel point cloud registration algorithm based on kernel-induced feature maps, kernel PCA and $SE(3)$ on-manifold optimization. The framework is theoretically elegant, and exhibits robustness and accuracy in fairly challenging circumstances. It is quite general and flexible to be extended to different dimensions and intra-category instances alignment . Remarkably, it outperforms most other methods in terms of efficiency.

ACKNOWLEDGMENT

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

REFERENCES

[1] P. J. Besl and H. D. Mckay, “A Method for Registration of 3-D Shapes,” *PAMI*, vol. 14, no. 2, pp. 239–256, 1992.

[2] S. Rusinkiewicz and M. Levoy, “Efficient Variants of the ICP Algorithm,” in *3DIM*, 2001, pp. 145–152.

[3] S. Gold, A. Rangarajan, C. Lu, and E. Mjolsness, “New Algorithms for 2D and 3D Point Matching: Pose Estimation and Correspondence,” *Pattern Recognition*, vol. 31, pp. 957–964, 1997.

[4] S. Granger and X. Pennec, “Multi-scale EM-ICP: A Fast and Robust Approach for Surface Registration,” in *European Conference on Computer Vision (ECCV 2002), volume 2353 of LNCS*. Springer, 2002, pp. 418–432.

[5] B. Jian and B. C. Vemuri, “Robust Point Set Registration Using Gaussian Mixture Models,” *PAMI*, vol. 33, no. 8, pp. 1633–1645, 2011.

[6] R. Kondor and T. Jebara, “A Kernel between Sets of Vectors,” in *ICML*, 2003.

[7] B. Schölkopf, A. Smola, E. Smola, and K.-R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[8] Z. Zhang, “Iterative point matching for registration of free-form curves and surfaces,” *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[9] C. J. Taylor and D. J. Kriegman, “Minimization on the Lie Group SO(3) and Related Manifolds,” Yale University, Tech. Rep., 1994.

[10] A. Sharma and K. K. Paliwal, “Fast Principal Component Analysis using Fixed-point Algorithm,” *Pattern Recognition Letters*, vol. 28, no. 10, pp. 1151–1155, 2007.

[11] A. Kasper, Z. Xue, and R. Dillmann, “The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics,” *The International Journal of Robotics Research*, May 2012.

[12] Y. Liu, “Automatic Registration of Overlapping 3D Point Clouds using Closest Points,” *Image and Vision Computing*, vol. 24, no. 7, pp. 762–781, 2006.