Noname manuscript No.
(will be inserted by the editor)

A Kernel-based Approach to Learning Contact Distributions for Robot Manipulation Tasks

Oliver Kroemer, Simon Leischnig, Stefan Luettgen, and Jan Peters

Received: date / Accepted: date

Abstract Manipulation tasks often require robots to recognize interactions between objects. For example, a robot may need to determine if it has grasped an object properly or if one object is resting on another in a stable manner. These interactions usually depend on the contacts between the objects, with different distributions of contacts affording different interactions.

In this paper, we address the problem of learning to recognize interactions between objects based on contact distributions. We present a kernel-based approach for representing the estimated contact distributions. The kernel can be used for various interactions, and it allows the robot to employ a variety of kernel methods from machine learning. The approach was evaluated on blind grasping, lifting, and stacking tasks. Using 30 training samples and the proposed kernel, the robot already achieved classification accuracies of 71.9%, 85.93%, and 97.5% for the blind grasping, lifting and stacking tasks respectively. The kernel was also used to cluster interactions using spectral clustering. The clustering method successfully differentiated between different types of interactions, including placing, inserting, and pushing. The contact points were extracted using tactile sensors or 3D point cloud models of the objects. The robot could construct small towers of assorted blocks using the classifier for the stacking task.

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreements 610878 (3rdHand), 600716 (CoDyCo), and 610967 (TACMAN).

O. Kroemer
University of Southern California, USA
E-mail: okroemer@usc.edu

S. Leischnig, S. Luettgen, and J. Peters
Technische Universitaet Darmstadt, Germany
E-mail: peters@ias.tu-darmstadt.de

J. Peters is also with the
MPI for Intelligent Systems, Germany

1 Introduction

Manipulation tasks usually involve direct physical contact between an object and either the robot or another object. Different types of interactions and manipulations are afforded depending on the locations of the contacts. For example, contacts on the side of an object afford pushing the object, while contacts on the bottom can be used to support the object. In order to use these interactions, the robot needs to be able to determine if the contacts resulting from a certain hand-object or object-object configuration will afford the desired interaction.

Analytical approaches for assessing interactions, such as stable grasps or object placements, generally require information about the mass and frictional properties of the objects as well as detailed 3D object models. This information is often not readily available to the robot. The robot can however approximate the positions and normals of the contact points using either tactile sensors or coarse point cloud models of the objects. The robot can then learn to predict if an interaction is afforded based on the distribution of these extracted contact points. Acquiring positive and negative training samples of interactions is also usually easier than performing an analytical analysis of the interaction. The robot can often even acquire the training samples' labels autonomously by interacting with objects to verify if they afford the interaction (Ugur and Piater, 2015; Griffith et al, 2012).

We present a kernel-based learning approach to recognizing object interactions from contacts. The kernel

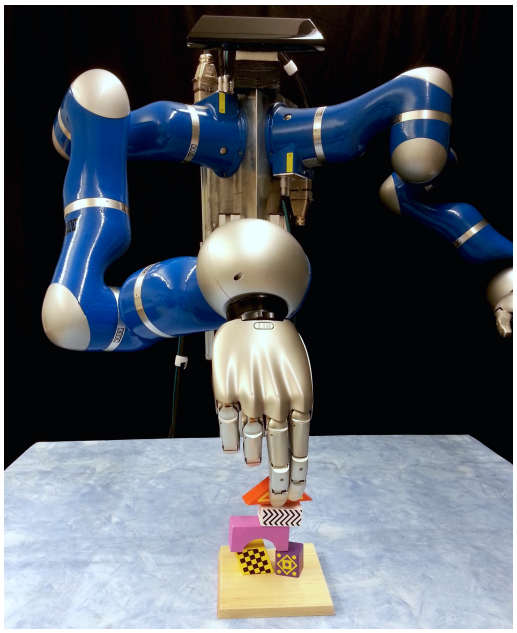


Fig. 1 The Darias robot performing a block stacking task. The robot learns suitable block placements using a kernel function for comparing contact distributions.

defines the similarity between different sets of contacts, and it can be used with a wide range of kernel methods from machine learning. To compute the kernel value, the robot first extracts a set of contact points using either 3D point cloud models or tactile sensing. The extracted contact positions and normals are defined in an interaction frame, e.g., the object's or hand's coordinate frame. This interaction frame defines a shared basis for comparing the contacts across different scenarios. The robot then computes the distribution over the contacts. We use a Gaussian form for representing the contact distributions. Although the contact distributions have the same form as probability distributions, they should not be interpreted as actual probability densities. The robot subsequently computes the Bhattacharyya kernel between pairs of distributions to determine their similarity (Jebara and Kondor, 2003). We adapt the standard Bhattacharyya kernel to include additional hyperparameters that define the task-specific similarities between individual contacts. Given the contact kernel, the robot uses kernel logistic regression to predict whether or not a set of contacts affords a certain interaction. For example, the robot can predict whether the contacts between its hand and an object will allow it to lift the object. The robot can also use spectral clustering (Shi and Malik, 2000; Luxburg, 2007) with the kernel to cluster different interactions. The details for computing and using the contact kernel are explained in Section 2.

The proposed approach was implemented on the robot shown in Fig. 1 using a ReflexHand and a five-fingered DLR hand. The robot uses the ReflexHand's TakkTile sensors to extract the contacts between the hand and the objects. The DLR hand does not have tactile sensors, and the contacts for these experiments were extracted using 3D point cloud models of the objects. The classification evaluations involved three different tasks: grasping objects, lifting an elongated box, and stacking assorted blocks. The grasping experiment was used to benchmark contact kernels based on different types of contact distributions. The lifting and stacking experiments explored the effects of using different contact representations and hyperparameter structures respectively. The fourth experiment explored using the proposed kernel to cluster different interactions. The details of the experiments are given in Section 3.

This paper builds on two previous conference papers on contact kernels (Kroemer and Peters, 2014; Leischnig et al, 2015). In this paper, we extend our research to using the kernel in a clustering framework. We explain the spectral clustering method in Section 2.5, and present an experimental evaluation in Section 3.4. We also extend our previous experiments. The grasping experiment incorporates a grid search to select individual hyperparameter settings for each approach, and we discuss the resulting distributions over the hyperparameters. The grasping experiment also explores using the force readings from the tactile sensors in the contact vectors. When using a large variety of grasps and objects, the results suggest that the force information decreases performance, possibly due to interaction forces with the table. Our lifting experiment incorporates evaluations of hand-relative contact vectors including only position information, as well as position and normal information. The experiment thus demonstrates the effects of using different interaction frames. The stacking experiment has been extended to evaluate the hyperparameter structure where the three position and three normal dimensions each share one single hyperparameter. This hyperparameter structure is used in both the grasping and lifting experiments.

1.1 Problem Statement

From a machine learning perspective, we can frame the problem of predicting object interactions from contacts as a classification problem. However, rather than classifying a single element, the robot has to classify a set of elements, i.e., the contacts. The i^{th} sample $\boldsymbol{X}_i$ is a set of n_i unordered elements $\boldsymbol{X}_i = \{\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \ldots, \boldsymbol{x}_{in_i}\}$, where each element is defined in a d dimensional space $\boldsymbol{x} \in \mathbb{R}^d$. Each sample also corresponds to a binary label $Y_i \in \{0, 1\}$, which indicates the class of the sample.

These labels are known for the training data, but need to be predicted for the test data.

For predicting interactions from contacts, the elements correspond to contacts and the samples are sets of contacts that may or may not afford the desired interaction, as indicated by their labels. For example, a blind grasp attempt corresponds to one sample, and the contact points extracted from the hand's tactile sensors are the elements of the sample. In our experiments, we represent contacts using $d = 6$ dimensional feature vectors that include the 3D contact positions and either the 3D normals or forces at the contact points. The contacts are defined relative to interaction frames. These interaction frames may, for example, be the coordinate frame of the hand for blind grasping or the gravity-aligned object frame for stacking objects. The contact vectors could potentially be extended with additional information, such as forces and friction coefficients. However, this information is often not readily available to the robot. We estimate the contact points and normals using either tactile sensor arrays or 3D models of the objects. The forces are estimated using the robot's tactile sensor arrays.

A sample's label is positive $Y_i = 1$ if the interaction is occurring between the objects for the corresponding sample and $Y_i = 0$ otherwise. As object interactions are not mutually exclusive, the robot can learn to predict multiple interactions between objects by learning separate binary classifiers for each type of interaction.

1.2 Related Work

Learning object interactions is an important ability for robots to define symbolic states of objects. For example, Kulick et al (2013) proposed an active learning approach for determining relational symbols between objects based on their relative positions and geometric properties. Rosman and Ramamoorthy (2011) learned spatial relations between objects, e.g. ON and ADJACENT, using k-means classifiers based on the normals of the contact points. Sjoo and Jensfelt (2011) learn to select relevant features for determining spatial relations between objects using an automatic relevance determination approach. Abdo et al (2013) selected features with small variations to learn the preconditions and effects of manipulations.

Learning interactions allows the robot to predict whether objects in the environment afford certain manipulations, e.g., pushing and placing (Gibson, 1986; Sahin et al, 2007; Montesano et al, 2007). Jiang et al (2012) proposed a method for classifying locations to place an object based on local geometric features of the scene. Hermans et al (2013) learned locations on objects for pushing based on their overall shape and the local geometry around the pushing point. Montesano et al (2007) learned the Bayesian network structure for modeling affordances. Their experiments showed that including the duration of the contacts improved the robot's accuracy when classifying affordances. Kopicki et al (2011) presented a product of experts framework for modelling pushing interactions. Their approach incorporates local models to capture object proximity and contact information. Our approach uses a task-independent kernel function to capture the spatial contact information of different types of interactions. Given data from specific tasks, the robot then uses this kernel to learn classifiers with task-specific parameters and hyperparameters to classify the interactions.

Grasping is one of the most extensively studied manipulations in robotics. The quality of a grasp depends on the contacts made between the hand and the object (Roa and Suàrez, 2015; Bicchi and Kumar, 2000). Early work on grasping focused on deriving analytical solutions (Bicchi and Kumar, 2000), and a number of grasp quality measures have been proposed as a result of this work (Miller and Allen, 1999; Ferrari and Canny, 1992; Li and Sastry, 1988; Roa and Suàrez, 2015). However, analytical analysis generally requires additional information about the object and its material properties. The analytical approach would also require one to derive additional methods for new types of interactions (Trinkle and Paul, 1990), rather than just providing the robot with samples.

The robot can alternatively use data-driven methods to learn how to grasp objects (Bohg et al, 2014). Suitable contact points are often learned implicitly based on the local shape of the object relative to the robot's hand (Detry et al, 2012; Herzog et al, 2013; ten Pa and Platt, 2015; Kroemer et al, 2012b). Although these methods can predict grasps precisely, they usually assume a constant preshape of the hand and only explicitly consider variations in the shape of the object being grasped. As a result, these methods are not well-suited for object-object interactions, e.g. tool use, wherein the robot must generalize over variations in the shapes of both objects.

Recent grasping approaches have incorporated deep neural networks to learn suitable features for representing the local object shape (Lenz et al, 2013; Kappler et al, 2015). Deep learning methods allow the robot to learn from large amounts of grasping data. However, the large number of parameters in these networks makes them difficult to train on small training sets. We focus on learning interactions from training sets with tens

or hundreds of samples, which is orders of magnitude smaller than most datasets used for deep learning.

Some grasp synthesis methods explicitly model the contact points to predict the quality of the grasp (Dang and Allen, 2012; Kroemer and Peters, 2014; Leischnig et al, 2015). The contact points are often predicted using a grasp simulator or they are estimated after the grasp has been executed to verify the quality of the grasp. These explicit approaches can be used to model contact interactions between pairs of objects. In Section 3, we compare our kernel approach to a bag-of-features approach (Dang and Allen, 2012).

Kopicki et al (2015) propose a one-shot learning method for *synthesizing* grasps of novel objects. Instead of relying on multiple training samples, their method learns grasp models from single grasp demonstrations. In this manner, the robot can learn to generalize specific grasps between different objects. To capture the nuances of the individual types of grasps, the grasp model uses a product of experts to incorporate contact models for the individual links, as well as a hand configuration model. Their contact models also incorporate curvature information. Ben Amor et al (2012) proposed a one-shot approach to generalizing grasps between similar objects. Their approach is based on warping the 3D models of objects to extract similar contact points, and using distinct low-dimensional hand configuration spaces to capture different types of grasps. Our approach focuses on learning discriminative classifiers, from positive and negative samples, to determine which contact distributions afford or inhibit the object interactions.

Contact information is also important for compensating for errors and verifying if a manipulation was successful. Guarded motions detect when the robot has made contact between an object in its environment and a certain amount of force has been applied (Will and Grossman, 1975). Force-torque sensors can be used to estimate the tooltip contact point after the tool has shifted in the robot's hand (Karayiannidis et al, 2014). Tactile servoing allows the robot to maintain contact and track tactile features on objects during manipulations (Li et al, 2013; Veiga et al, 2015). The robot can also verify the quality of grasps based on tactile readings (Madry et al, 2014; Bekiroglu et al, 2011). Our approach considers the distribution of contacts between objects to detect interactions. We explain how contact points are estimated using either tactile or 3D point cloud data in Section 2.1.

The problem of classifying sets of contact points is similar to set classification (Ning and Karypis, 2008) and multi-instance classification (Amores, 2013) problems. However, these problem formulations generally assume that each element is associated with a class label. The sample's label is then defined by the proportions of the element's labels. This assumption does not hold for classifying sets of contacts. For example, two opposing contact can pinch an object, but neither contact can pinch the object individually. In this manner, interactions depend on the relationships between the contacts.

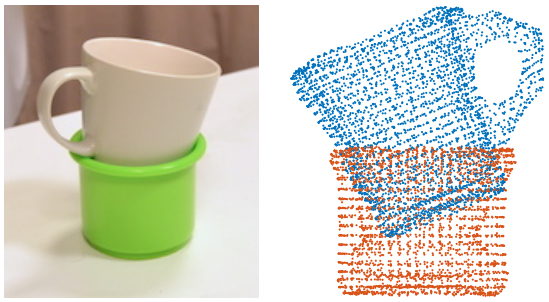


Fig. 2 The figure shows an example scene of two objects interacting and the corresponding point cloud models from a side view. The primary object's points $\hat{\boldsymbol{p}}_{ij}$ are shown in blue, and the secondary object's points $\check{\boldsymbol{p}}_{ij}$ are in red. The point cloud models were generated from single depth images obtained using a Microsoft Kinect. The models were created by exploiting their rotationally extruded shapes (Kroemer et al, 2012a). These models capture the coarse shape of the object, e.g., the slanted sides of the cup, but not all of the details, e.g., the precise shape of the handle. The proposed method works well even with coarse object models.

2 A Kernel for Contact Distributions

In Sections 2.1 and 2.2, we explain how contacts between objects are detected and represented for computing contact distributions. In Sections 2.3, we provide a kernel function for computing the similarity between contact distributions. We explain how the kernel is used to classify and cluster sets of contacts in Sections 2.4 and 2.5 respectively.

2.1 Extracting Contact Points

The first step for computing the i^{th} sample's contact distribution is to extract the set of n_i contacts on the object of interest. The contacts are estimated using either tactile sensor arrays or full 3D point cloud models of the objects. Tactile sensor arrays are well-suited for extracting hand-object contacts. Full point cloud models of object are useful for estimating object-object contacts in the scene, or if the robot does not have tactile sensors. A partial point cloud from a vision sensor can usually not be used to extract a set of contact points

as the contact points will be occluded. We therefore focus on extracting points from full models. These models can be fairly coarse. For the clustering experiment, we generated coarse models by acquiring single views using a Microsoft Kinect and then fitting a linearly or rotationally extruded shape to complete the model (Kroemer et al, 2012a). We used partial point clouds from a Kinect in our block tower experiment to *predict* contact points.

In our grasping experiment, the robot uses a tactile sensor array to extract a set of contact points between the hand and the object. Using the robot's forward kinematics, we can compute the position $\hat{\boldsymbol{p}}_{ij} \in \mathbb{R}^3$ and surface normal $\hat{\boldsymbol{u}}_{ij} \in \mathbb{R}^3$ of each of the $\hat{n}_i$ taxels in the tactile array for the i^{th} sample. These positions and normals are defined in the world coordinate frame. Each taxel also provides a sensor reading of the applied normal force s. This sensor value is used to determine if the taxel is in contact with the object. The n_i contact points, with positions $\tilde{\boldsymbol{p}}_{ij} \in \mathbb{R}^3$ and normals $\tilde{\boldsymbol{u}}_{ij} \in \mathbb{R}^3$ in the world coordinate frame are given by the taxels with sensor values greater than a threshold $s \geq \tau$. We zero the sensor values before grasping and use a threshold value of $\tau = 15$.

We use a similar approach for extracting the contact points from 3D point cloud models. For the i^{th} sample, the robot has two sets of 3D point clouds. The first set represents the shape of the primary object in the scene and it includes $\hat{n}_i$ positions $\hat{\boldsymbol{p}}_{ij} \in \mathbb{R}^3$ and corresponding surface normals $\hat{\boldsymbol{u}}_{ij} \in \mathbb{R}^3$. The second set of points represents the shape of the object with which the primary object is interacting. This point cloud includes $\check{n}_i$ position vectors $\check{\boldsymbol{p}}_{ij} \in \mathbb{R}^3$ and corresponding surface normals $\check{\boldsymbol{u}}_{ij} \in \mathbb{R}^3$. Both of these point clouds are defined in the world coordinate frame. An example scene and its point cloud models are shown is Fig. 2. The individual object models were generated using a Kinect by exploiting the objects' extruded shapes (Kroemer et al, 2012a). The robot must subsequently select a subset of the points from the primary point cloud as the contact points. To select the contact points, we train a logistic regression classifier on ten training samples to classify each of the points in the primary point cloud. The classifier uses two features

$$\phi_1 = \sum_{k=1}^{\check{n}_i} \exp\left(-\frac{\left\|\hat{\boldsymbol{p}}_{ij} - \check{\boldsymbol{p}}_{ik}\right\|^2}{\sigma^2}\right)$$

and

$$\phi_2 = \sum_{k=1}^{\check{n}_i} \left(\hat{\boldsymbol{u}}_{ij}^T \check{\boldsymbol{u}}_{ik}\right) \exp\left(-\frac{\left\|\hat{\boldsymbol{p}}_{ij} - \check{\boldsymbol{p}}_{ik}\right\|^2}{\sigma^2}\right)$$

as well as a constant bias term $\phi_3 = 1$ to classify the j^{th} point in the primary point cloud. The hyperparameter $\sigma = 0.5$cm defines the length scale of the point cloud features. The subset of the primary points that were classified as being in contact with the secondary object then represent the n_i contact points for the i^{th} sample with positions $\tilde{\boldsymbol{p}}_{ij} \in \mathbb{R}^3$ and surface normals $\tilde{\boldsymbol{u}}_{ij} \in \mathbb{R}^3$. These points are again defined in the world coordinate frame for now.

The extracted set of contacts will ultimately be defined relative to an interaction frame and represented by a contact distribution. These representations capture the general distribution of the contacts, but they may not capture all of the contacts' details. Hence, although one should attempt to extract accurate and precise contact estimates, the proposed method does not rely on exact contact information. Our experiments have shown that the proposed method can perform well even when using coarse contact estimates.

2.2 Defining an Interaction Frame

In order to compare different sets of contacts, we also need to define a suitable interaction frame. The positions and normals of the contacts are then defined relative to the interaction frame. An interaction frame for the i^{th} sample is defined by a 3D position $\boldsymbol{\rho}_i$ and a set of three orthonormal axes $\boldsymbol{a}_i^x$, $\boldsymbol{a}_i^y$, and $\boldsymbol{a}_i^z$ in the world frame. For example, the interaction frame may be defined as the coordinate frame of the robot's palm for a blind grasping task. In this manner, the contacts of different grasps would always be defined relative to the hand.

In practice, one may often only be able to define the position $\boldsymbol{\rho}$ and one axis $\boldsymbol{a}_i^x$ of the interaction frame. For example, in a lifting task, the position $\boldsymbol{\rho}_i$ can be defined by the location of the object's center of mass and the axis $\boldsymbol{a}_i^x$ is defined by the direction of gravity. Similarly, for articulated objects such as levers and door handles, the position and the axis of the revolute joint can be used to define the interaction frame's position and first axis. In both of these examples, the remaining two axes are not defined due to the rotational symmetry of the tasks. In these situations, the remaining axes can be selected such that they align the contact points.

We first translate and project the contact points into a 2D plane, with the normal of the plane given by the first axis $\boldsymbol{a}_i^x$, such that the point $\tilde{\boldsymbol{p}}_{ij}$ becomes

$$\tilde{\boldsymbol{p}}'_{ij} = \left(\tilde{\boldsymbol{p}}_{ij} - \boldsymbol{\rho}_i\right) - \boldsymbol{a}_i^{xT} \left(\tilde{\boldsymbol{p}}_{ij} - \boldsymbol{\rho}_i\right) \boldsymbol{a}_i^x$$

We then compute the symmetric matrix $\boldsymbol{S} \in \mathbb{R}^3$ of second moments about the origin $\boldsymbol{\rho}$

$$\boldsymbol{S}_i = n_i^{-1} \sum_{j=1}^{n_i} \tilde{\boldsymbol{p}}'_{ij} \tilde{\boldsymbol{p}}'^T_{ij}$$

and compute its eigenvectors. The second axis $\boldsymbol{a}_i^y$ is defined by the eigenvector of $\boldsymbol{S}_i$ with the largest eigenvalue, such that the mean of the contact points is in the positive direction, i.e., $\sum_j \boldsymbol{a}_i^{yT}(\tilde{\boldsymbol{p}}_{ij} - \boldsymbol{\rho}_i) \geq 0$. Using this approach, the contact point clouds are aligned according to the radial direction with the largest variance. The third axis is simply given by the cross product of the first two $\boldsymbol{a}_i^z = \boldsymbol{a}_i^x \times \boldsymbol{a}_i^y$ to form a right-handed coordinate frame. One could also use a left-handed coordinate frame, but it is important to be consistent. Once the interaction frame has been computed, the extracted contact point positions $\tilde{\boldsymbol{p}}_{ij}$ and surface normals $\tilde{\boldsymbol{u}}_{ij}$ in the world frame are used to compute the positions $\boldsymbol{p}_{ij}$ and surface normals $\boldsymbol{u}_{ij}$ in the interaction frames. These positions and normals are then concatenated to form the final contact vectors $\boldsymbol{x}_{ij} = \begin{bmatrix}\boldsymbol{p}_{ij}^T & \boldsymbol{u}_{ij}^T\end{bmatrix}^T$.

Selecting a suitable interaction frame is important as it will affect the similarity value computed by the kernel. As a general rule, the interaction frame is usually linked to the variable being controlled in the interaction. For example, a robot may need to determine suitable contacts for turning a lever. The first axis of the interaction frame $\boldsymbol{a}_i^x$ would then correspond to the lever's axis of rotation.

2.3 A Contact Distribution Kernel

Given a set of contact points in the interaction frame, the next step is to define a kernel for comparing two interaction samples. Rather than comparing contact points individually, we first model the set of contact points as a distribution in the $\mathbb{R}^d$ contact space and then compute a kernel between these distributions. We propose the Bhattacharyya kernel (Jebara and Kondor, 2003). This kernel represents the contact distributions as Gaussians, although the contact distributions should not be interpreted as actual probability distributions.

The Bhattacharyya kernel (Jebara and Kondor, 2003) represents each set of contacts $\boldsymbol{X}_i$ as a single Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ where

$$\boldsymbol{\mu}_i = \frac{1}{n_i}\sum_{k=1}^{n_i} \boldsymbol{x}_{ik},$$

$$\boldsymbol{\Sigma}_i = \frac{1}{n_i}\sum_{k=1}^{n_i} (\boldsymbol{x}_{ik} - \boldsymbol{\mu}_i)(\boldsymbol{x}_{ik} - \boldsymbol{\mu}_i)^T + \boldsymbol{\Sigma}_0,$$

and the matrix $\boldsymbol{\Sigma}_0 \in \mathbb{R}^{d\times d}$ contains a set of hyperparameters. As Gaussians are uni-modal, this representation captures the $\mathbb{R}^d$ space spanned by the contacts rather than the locations of the individual contacts. The mean of the contact distribution may therefore correspond to a point with a normal of zero length. As a result, a single contact with a unit normal will not be able to create a similar contact distribution as two contacts with opposing normals. However, when using forces instead of normals, two opposing contact forces may have a similar mean to a single contact with almost zero force.

The additional covariance term $\boldsymbol{\Sigma}_0$ defines the similarity between individual contacts and has the same effect as adding a Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_0)$ to the contact points. It allows the representation to capture the relevance of different dimensions in the contact space. For example, when pushing open a door, the horizontal distance from the axis of rotation is more relevant than the vertical position along the axis. By adding more variance in the vertical dimension, the kernel can capture the fact that two contacts are more similar if they are offset vertically rather than horizontally from each other. The experiments in Section 3.3 show that the robot can use this additional similarity information to increase the sample efficiency of the learning algorithm. The hyperparameters are autonomously tuned using a grid search approach.

The Bhattacharyya kernel (Jebara and Kondor, 2003) between two contact distributions is given by

$$\begin{aligned} k(\boldsymbol{X}_i, \boldsymbol{X}_j) &= k((\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)) \\ &= \int \sqrt{\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}\sqrt{\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}\mathrm{d}\boldsymbol{x}. \end{aligned}$$

The computation of the kernel is given in (Jebara et al, 2004), and we include it here for completeness. The kernel function is computed in closed form as

$$k((\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)) = C\exp\left(-M/4\right),$$

where the values of C and M are given by

$$C = 0.5^{-d/2}|\hat{\boldsymbol{\Sigma}}|^{1/2}\,|\boldsymbol{\Sigma}_i|^{-1/4}\,|\boldsymbol{\Sigma}_j|^{-1/4},$$

$$M = \boldsymbol{\mu}_i^T\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i + \boldsymbol{\mu}_j^T\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j - \hat{\boldsymbol{\mu}}^T\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{\mu}}.$$

The vector $\hat{\boldsymbol{\mu}}$ is given by $\hat{\boldsymbol{\mu}} = \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i + \boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j$, and the matrix $\hat{\boldsymbol{\Sigma}}$ is computed as $\hat{\boldsymbol{\Sigma}} = (\boldsymbol{\Sigma}_i^{-1} + \boldsymbol{\Sigma}_j^{-1})^{-1}$. The kernel's value is $k(\boldsymbol{X}_i, \boldsymbol{X}_j) = 1$ if the contact distributions are identical, and tends to zero as the overlap between the distributions decreases. In our experiments, all of the samples include at least one contact. However, if a sample were to not contain any elements $\boldsymbol{X}_i = \emptyset$, then we would define the kernel to be $k(\boldsymbol{X}_i, \boldsymbol{X}_j) = 1$ if $\boldsymbol{X}_j = \emptyset$ and $k(\boldsymbol{X}_i, \boldsymbol{X}_j) = 0$ otherwise.

This basic kernel can be used to construct more complicated contact kernels. For example, the robot may compute one basic kernel to capture the set of object-hand contacts and another for the set of object-table contacts. A new kernel over both sets of contacts can then be created by multiplying or adding the two basic kernels (Schölkopf and Smola, 2001; Kroemer

et al, 2015). The basic kernels can even use different interaction frames. We only employ the basic kernel in our experiments.

2.4 Classifying Contact Distributions

Having defined a kernel between contact distributions, we can now use kernel methods from machine learning to classify the contact interaction (Schölkopf and Smola, 2001). We use kernel logistic regression to classify the interactions. Kernel logistic regression uses the similarity to previously observed distributions, with labels, to classify new contact distributions. The probability that a contact distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ allows for a certain interaction $\mathcal{I}$ is given by

$$p(\mathcal{I}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = (1 + \exp(\alpha))^{-1},$$

where

$$\alpha = \theta_0 + \sum_{j=1}^{m} \theta_j k(\boldsymbol{X}_i, \boldsymbol{X}'_j),$$

and the robot has m training samples $\boldsymbol{X}'_j$ of contact distributions. The weight parameters θ can be learned using iterative reweighted least squares. Previous contact distributions that afforded the interaction will generally have more negative weights, which will result in a probability closer to one. Contact distributions that are not similar to any previous distributions will have a probability defined by θ_0. As kernel logistic regression is a probabilistic classifier, it can model a contact distribution that only sometimes affords the interaction.

2.5 Clustering Contact Distributions

The proposed kernel can also be used to cluster different interactions. We use the spectral clustering method proposed of Shi and Malik (2000) to cluster the samples into κ samples. Given m samples, the robot first computes the normalized graph Laplacian $\boldsymbol{L} \in \mathbb{R}^{m\times m}$ as $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{K}$, where $\boldsymbol{I} \in \mathbb{R}^{m\times m}$ is the identity matrix, the element in the i^{th} row and j^{th} column of the kernel matrix $\boldsymbol{K} \in \mathbb{R}^{m\times m}$ is given by $[\boldsymbol{K}]_{ij} = k(\boldsymbol{X}_i, \boldsymbol{X}_j)$, and the diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{m\times m}$ has elements given by $[\boldsymbol{D}]_{ii} = \sum_{j=1}^{m} k(\boldsymbol{X}_i, \boldsymbol{X}_j)$. Given the normalized Laplacian, the robot computes the matrix $\boldsymbol{E} \in \mathbb{R}^{m\times\kappa}$, which contains the κ eigenvectors of $\boldsymbol{L}$ with the smallest eigenvalues. Finally, the robot performs k-means clustering, with κ clusters, on the *rows* of the matrix $\boldsymbol{E}$. The sample $\boldsymbol{X}_i$ is then associated to the cluster that the i^{th} row of $\boldsymbol{E}$ was assigned to.

The normalized spectral clustering algorithm requires the number of clusters κ to be predefined. We employ a heuristic to automatically select a suitable value for κ. The robot performs the clustering multiple times and samples the number of clusters κ from a uniform distribution each time. In our experiments, we ran the clustering 1500 times and sampled κ from a range between two and ten. For each clustering, the robot computes a matrix $\tilde{\boldsymbol{K}}$ with elements $\left[\tilde{\boldsymbol{K}}\right]_{i,j} = k(\boldsymbol{X}_i, \boldsymbol{X}_j)$ if $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ are in the same cluster, and $\left[\tilde{\boldsymbol{K}}\right]_{i,j} = 1 - k(\boldsymbol{X}_i, \boldsymbol{X}_j)$ otherwise. The robot then assigns a score to the clustering which is the sum of all of the elements in the matrix $\tilde{\boldsymbol{K}}$. The clustering thus receives a higher score for placing similar samples in the same cluster and distinct samples in different clusters. The robot ultimately selects the clustering with the highest score. This approach also reduces the effects of the random initialization of the k-means clustering.

3 Evaluations and Experiments

The proposed kernel approach was evaluated on grasping, lifting, and stacking tasks. The experiments were performed using the robot shown in Fig. 1. The robot has KUKA lightweight robot arms and a Microsoft Kinect head to observe the scenes. The grasping task was performed using a RightHand Robotics ReflexHand (Odhner et al, 2014) equipped with TakkTile sensors (Jentoft et al, 2013) in the fingers and palm, which we used to extract the contact points for the blind grasping experiment. We also evaluated the performance on simulated grasps using the GraspIT simulator (Miller and Allen, 2004). The lifting and stacking tasks were performed using a DLR five-fingered hand (Chen et al, 2010), and the contacts were extracted using 3D point cloud models of the objects. The three tasks were used to investigate the effects of using different kernels, contact representations, and hyperparameter structures. We investigated using the proposed kernel for clustering interactions as explained in Section 3.4.

3.1 Grasping Experiments

In this experiment, we compare different kernels for estimating if a grasp is successful based on the set of detected contacts. In addition to the Bhattacharyya kernel (Bhat), we also compare using a bag-of-features approach (BoF), an exponential χ^2 kernel (Exp χ^2), and a normalized expected likelihood kernel (NEL). We evaluated the different representations using both simulated and real robot grasping data. In contrast to our previous work (Leischnig et al, 2015), we use grid searches to select the kernels' hyperparameters instead of using

a constant setting for all four approaches. The kernels tend to select different sets of hyperparameters. We also explore using the force readings of the tactile sensors as part of the contact point vectors $\boldsymbol{x}_{ij}$.

3.1.1 Baseline Methods for Benchmarking

The first benchmark method is a bag-of-features (BoF) approach. The bag-of-features representation is defined using a dictionary $\boldsymbol{D}$ of $\acute{n}$ prototypical contact elements $\boldsymbol{D} = \{\acute{\boldsymbol{x}}_1, \acute{\boldsymbol{x}}_2, \ldots, \acute{\boldsymbol{x}}_{\acute{n}}\}$, $\acute{\boldsymbol{x}} \in \mathbb{R}^d$. The prototype elements were computed by performing k-means clustering on all of the contacts in the dataset. We used $k = 64$ clusters, which achieved good results and was also the dictionary size used by Dang and Allen (2012). The distribution over elements in $\boldsymbol{X}_i$ can then be described by a feature vector $\boldsymbol{\phi}(\boldsymbol{X}_i) \in \mathbb{R}^{\acute{n}+1}$, where the j^{th} component of the feature vector indicates the proportion of contacts assigned to $\acute{\boldsymbol{x}}_j$, and the last component is a constant bias term of 1. We employ a soft assignment approach to reduce discretization effects, such that the j^{th} feature is given by

$$[\boldsymbol{\phi}(\boldsymbol{X}_i)]_j = \sum_{m=1}^{n_i} \frac{\exp(-0.5 d(\acute{\boldsymbol{x}}_j, \boldsymbol{x}_{im})^2)}{n_i \sum_{l=1}^{\tilde{n}} \exp(-0.5 d(\acute{\boldsymbol{x}}_l, \boldsymbol{x}_{im})^2)},$$

where the Mahalanobis distance is given by

$$d(\boldsymbol{x}_a, \boldsymbol{x}_b) = ((\boldsymbol{x}_a - \boldsymbol{x}_b)^T \Sigma_0^{-1} (\boldsymbol{x}_a - \boldsymbol{x}_b))^{1/2},$$

and the diagonal matrix $\boldsymbol{\Sigma}_0 \in \mathbb{R}^{d \times d}$ defines the similarity between the samples and the dictionary elements. The features were normalized in preprocessing, which increased the performance of the bag-of-features approach significantly.

The second benchmark method uses the exponential χ^2 kernel (Exp χ^2) (Hofmann et al, 2008). This kernel compares histograms, and its computation for the BoF representation is given by

$$k_\chi(X_i, X_j) = \exp\left(- \sum_{l=1}^{\acute{n}} \frac{([\boldsymbol{\phi}(\boldsymbol{X}_i)]_l - [\boldsymbol{\phi}(\boldsymbol{X}_j)]_l)^2}{0.5([\boldsymbol{\phi}(\boldsymbol{X}_i)]_l + [\boldsymbol{\phi}(\boldsymbol{X}_j)]_l)} \right).$$

The Exp χ^2 kernel has been used in computer vision (Vedaldi et al, 2009) and to learn locations on objects for pushing (Hermans et al, 2013).

The third benchmark method uses a normalized expected likelihood kernel (NEL) to compare contact distributions (Kroemer et al, 2012b). This kernel represents the set of contacts using a mixture of H_i Gaussians in the form

$$f_i(\boldsymbol{x}) = \sum_{h=1}^{H_i} \nu_{ih} \mathcal{N}(\boldsymbol{x} | \boldsymbol{\mu}_{ih}, \boldsymbol{\Sigma}_{ih}),$$

where ν_{ih} is the mixture component. In our experiments, we assign each contact point its own Gaussian, such that

$$f_i(\boldsymbol{x}) = \frac{1}{n_i} \sum_{h=1}^{n_i} \mathcal{N}(\boldsymbol{x} | \boldsymbol{x}_{ih}, \boldsymbol{\Sigma}_0),$$

where the matrix $\boldsymbol{\Sigma}_0 \in \mathbb{R}^{d \times d}$ again defines the similarity between individual contacts. The NEL kernel is then computed as

$$k(f_i(\boldsymbol{x}), f_j(\boldsymbol{x})) = \frac{\int f_i(\boldsymbol{x}) f_j(\boldsymbol{x}) \mathrm{d}x}{\sqrt{\int f_i(\boldsymbol{x}) f_i(\boldsymbol{x}) \mathrm{d}x} \sqrt{\int f_j(\boldsymbol{x}) f_j(\boldsymbol{x}) \mathrm{d}x}},$$

in closed-form. This kernel function has a value of 1 when the contact distributions are identical, and tends to zero as the overlap decreases. The kernel is based on the expected likelihood kernel (Jebara et al, 2004) and is closely related to the Cauchy-Schwarz divergence (Jenssen et al, 2006). The NEL kernel can capture more details of the contact distribution than the single Gaussian representation. However, this level of detail is often not needed when learning robust manipulation skills (Eppner and Brock, 2013).

All four evaluated methods use the diagonal covariance matrix $\boldsymbol{\Sigma}_0 \in \mathbb{R}^{d \times d}$ to define the similarity between individual contacts. Rather than determining a separate covariance for each dimension, all of the position dimensions and normal dimensions share the same hyperparameters $\boldsymbol{\Sigma}_0 = \mathrm{diag}(\sigma_p^2, \sigma_p^2, \sigma_p^2, \sigma_n^2, \sigma_n^2, \sigma_n^2)$.

3.1.2 Grasping with a Simulated Barrett Hand

Using the GraspIT simulator (Miller and Allen, 2004) and Columbia grasp database (Goldfeder et al, 2009), we collected the contact points and normals for 2000 grasps with a simulated 3-fingered Barret hand of everyday objects, including mugs and knives. This experiment was inspired by the work on blind grasping by Dang and Allen (2012). We used the contact points computed by the GraspIT simulator. The contact points were defined relative to the palm frame of the hand. We assumed a coefficient of friction of 1.0 between the hand and the objects. For each grasp, we also recorded the epsilon ϵ and volume v grasp quality metrics (Miller and Allen, 1999; Ferrari and Canny, 1992; Li and Sastry, 1988). The goal of the experiment was to predict from a set of contacts whether a grasp would result in high grasp quality metrics. A grasp was considered successful if it achieved grasp metrics of both $\epsilon \geq 0.07$ and $v \geq 0.1$, as proposed by Dang and Allen (2012).

The representations were evaluated using five-fold cross validation, such that the test sets consisted of 400 samples and the training sets were sampled from

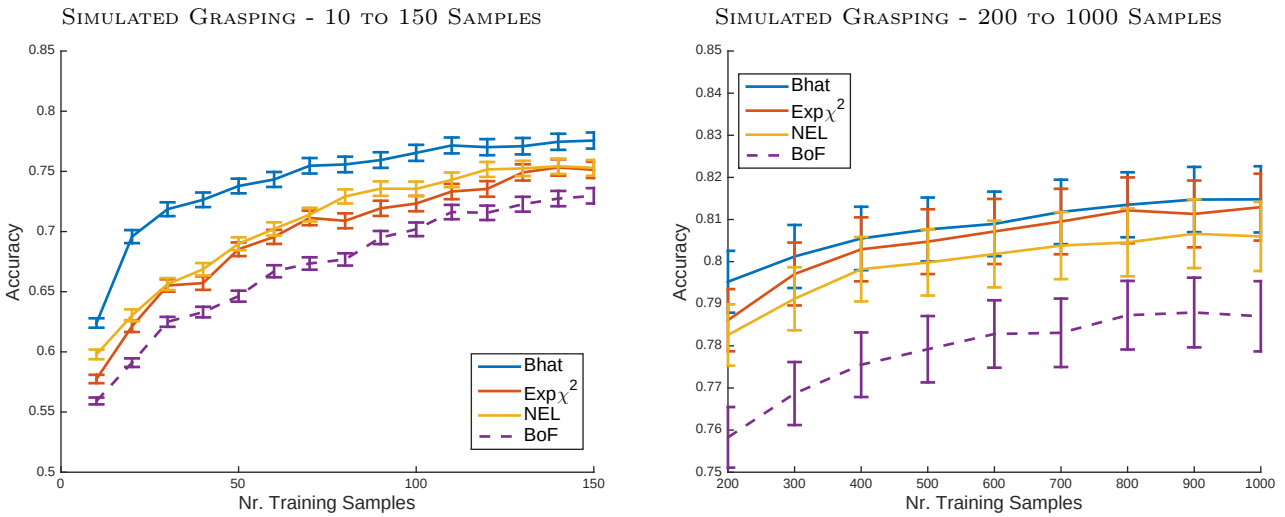


Fig. 3 The figure shows the classification accuracies for different training set sizes (left: 10 to 150, right: 200 to 1000) for the simulated grasping experiment. The compared methods include the Bhattacharyya kernel (Bhat), the exponential χ^2 kernel (Expχ^2), the normalized expected likelihood kernel (NEL), and the bag of features (BoF) representation. The errorbars indicate one standard error.

the remaining pool of 1600 samples. For each test set and training set size, we sampled 25 training sets with replacement. The classification accuracy for each test sample was computed based on these 25 classifiers' predictions. The standard deviations were then computed over the samples' accuracies.

The hyperparameters $\boldsymbol{\Sigma}_0$ were selected using a grid search with five-fold cross validation on the training set. The grid search was performed over the hyperparameters $\sigma_p \in \{0.5, 1.0, 2.5, 5.0, 10.0, 15.0\}$ cm and $\sigma_n \in \{0.05, 0.1, 0.25, 0.5, 1.0, 1.5\}$. Hence, to evaluate one training set size for one method with automatic hyperparameter selection, the robot had to train $25 \times 5 \times (5 \times 36 + 1) = 22,625$ classifiers, i.e., 125 classifiers for the five-fold cross validation with 25 repetitions and another 180 for each of these classifiers to select the hyperparameters using an inner five-fold cross validation on the training data. The hyperparameter values were fixed for training sets with more than 200 samples to the values selected for the 200 training sample trials. The plots on the left side of fig. 4 show the distribution over the hyperparameter values for training set sizes from 10 to 100. The learning curves for the simulation data are shown in Fig. 3.

3.1.3 Grasping with a Real ReFlex Hand

In order to compare the methods on real robot data, we also collected 200 grasp attempts using a ReflexHand equipped with TakkTile sensor arrays on the fingers and palm (Odhner et al, 2014; Jentoft et al, 2013). Each finger has a single row of nine sensors along its length, with five on the proximal portion and four on the distal. The tactile sensors cover the inner side of the fingers, but not the sides or backs of the fingers. The elements of the array are spaced approximately 8mm apart and have a sensitivity of 0.01N according to the manufacturer's description. The palm of the hand is covered by three large gel segments, which contain additional TakkTile sensors (Odhner et al, 2014; Jentoft et al, 2013). Localizing the contact point on these segments is not trivial. Therefore, if a contact was detected for a palm sensor, the contact point estimate was set to the most elevated point on that segment with a normal vector orthogonal to the palm plane.

We demonstrated 200 grasps using the 50 objects shown in Fig. 5. For each attempt, the robot executed a grasping action at a predefined location on the table. A human operator placed the object on the table, and adjusted the height of the grasp, to demonstrate the grasps to the robot. The robot then used the ReflexHand's guarded grasping movement to close the fingers using the default threshold value 20. Grasps were selected such that all of the contacts are made with the tactile sensors. The dataset contains 151 top grasps and 49 side grasps. All grasps were performed using a cylindrical preshape of the hand. Both successful and unsuccessful grasps were demonstrated to the robot. A grasp was considered to be a success if it lifted the object above the table and did not rotate the object by more than 30 degrees during the lifting process.

Contact points for each attempt were extracted after grasping the object, but before attempting to lift it. The contact points were detected by thresholding

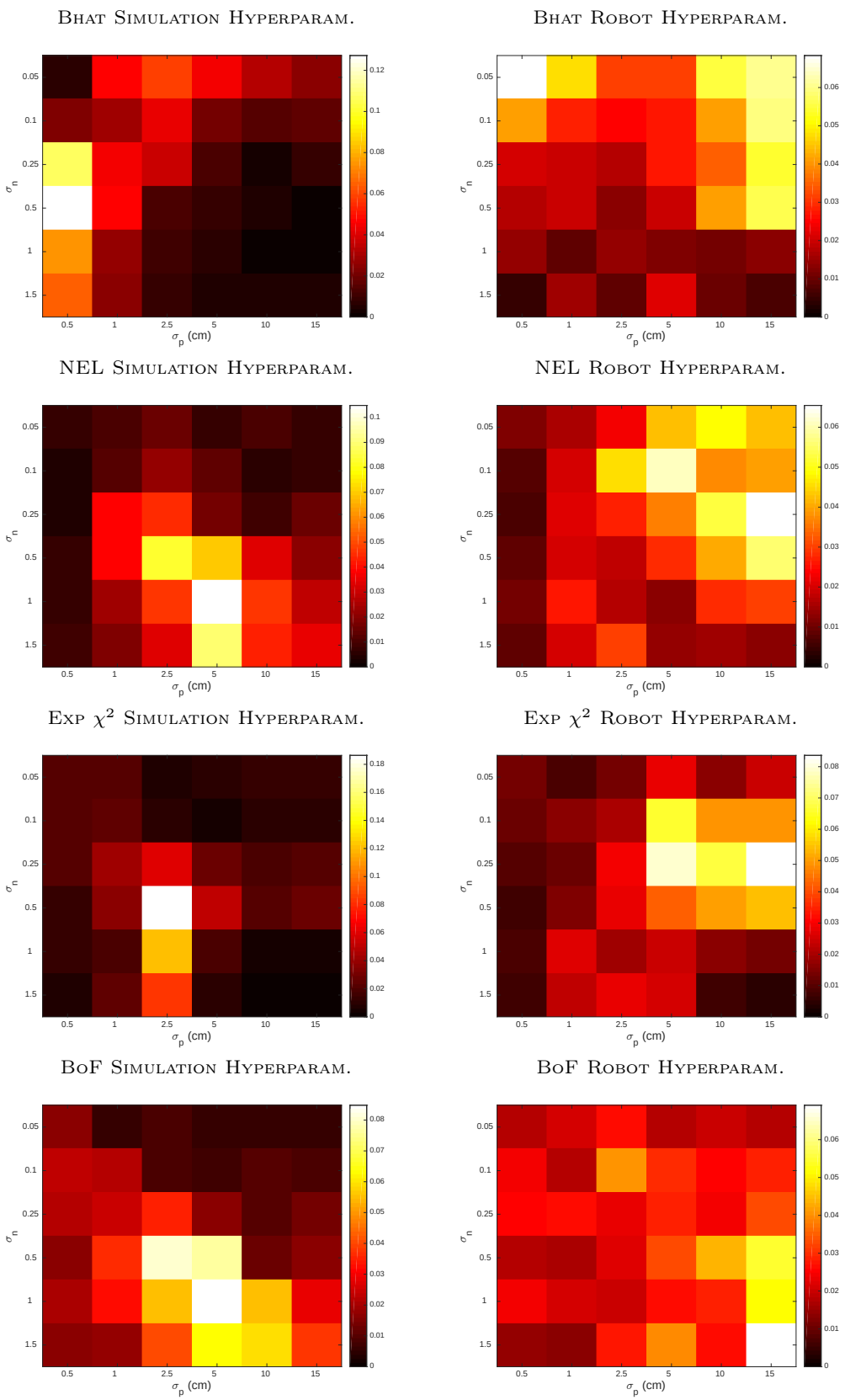


Fig. 4 The plots show the distributions over the hyperparameter settings for the evaluations using 10 to 100 samples. The left and right columns correspond to the simulated and real robot grasping experiments respectively. The four rows correspond to the Bhattacharyya kernel (Bhat), the normalized expected likelihood kernel (NEL), the exponential χ^2 kernel (Exp χ^2), and the bag-of-features approach (BoF).

the tactile sensors' readings at $\tau = 15$, as described in Section 2.1. If a taxel was considered to be in contact, its position and normal estimates were computed using the robot's forward kinematics. As part of a blind grasping framework, the contacts were defined relative to the robot's palm frame. In addition to the position and normal contact representation, we also explored using a position and force representation. The dimensionality is $d = 6$ in both cases. For the force representation, we scaled the normal vectors according to the taxels' output value. The sensor values were scaled by the inverse of their standard deviation and offset to get a median value of one. The taxels do not measure tangential forces and the extracted normal and force vectors are aligned. Including both the normals and forces would therefore be redundant.

The robot grasping data was evaluated in a similar manner to the simulated data. The five-fold cross validation was performed with 40 samples in each test set, and 160 samples in the training set pool. We evaluated 25 classifiers for each test set and training set size. The hyperparameter grid search was performed using $\sigma_p \in \{0.5, 1.0, 2.5, 5.0, 10.0, 15.0\}$ cm for the positions and $\sigma_n \in \{0.05, 0.1, 0.25, 0.5, 1.0, 1.5\}$ for the normals and scaled force estimates. The plots on the right of fig. 4 show the distribution over the hyperparameter values for training set sizes from 10 to 100 for the position and normals contact data. The learning curves for the robot's grasping data are shown in Fig. 6.

The kernel functions require different amounts of time to compute. For comparison, we recorded the times required to compute the values for 64 samples, i.e., the same number of samples as the bag-of-features representation has features. We recorded the time from receiving the contact vectors $\boldsymbol{x}_{ij}$ to computing the kernel or feature values. For the BoF and Expχ^2 approaches, we assumed that the dictionary was given and did not include the k-means clustering in the computation. The computations were performed in Matlab on a 1.8GHz Intel Core i7 MacBook Air with 4GB of memory. We averaged the times over the 36 different hyperparameter settings. The times for evaluating 64 samples were: Bhat 0.187s, NEL 0.215s, Expχ^2 0.050s, and BoF 0.029s. The Bag-of-Features approach is therefore the fastest, while the Bhattacharyya kernel is the second slowest after the NEL kernel. If we double the number of samples to 128, then the computation times become Bhat 0.693s, NEL 0.837s, Expχ^2 0.142s, and BoF 0.056s. As one would expect, the Bag-of-Features approach scales the best with more samples and may therefore be more suitable for very large amounts of data. However, even the longest time of 0.837s for 128 samples with the NEL kernel corresponds to 0.0065s for comparing one sample to 128 samples. This duration is sufficiently short for most applications.

Fig. 5 The objects used to create the grasping dataset. The RightHand robotics ReFlex hand that was used in the experiment is shown on the left. The hand is equipped with tactile sensor arrays to detect contacts.

3.1.4 Discussion of the Grasping Experiment

The results of the simulation experiment show that the proposed Bhattacharyya kernel achieved the highest accuracy. The difference in performance is most notable when the robot is given only a limited number of training samples. For 30 training samples, the Bhattacharyya kernel achieves an accuracy of 71.9% while the NEL kernel achieved the next highest accuracy of 65.6%. The bag-of-features approach, which is the fastest method to compute, resulted in the lowest accuracy of 62.5%. This result indicates that the robot should use the Bhattacharyya kernel when the number of training samples is limited.

Although the Bhattacharyya kernel obtained the highest accuracy for the real robot data, the performance increase over the other kernels is not significant. The Bhattacharyya kernel achieved an accuracy of 74.1% for 100 real robot training samples. The bag-of-features approach again achieved the lowest accuracy of 67.5% for 100 real robot training samples. Some of the grasps failed during the experiment because they were too far from the objects' center of masses, which resulted in large torques during the lifting. A perfect prediction accuracy based on the palm-relative contact information is therefore not possible. The latent mass distribution issue is a challenge inherent to the blind grasping task. The proposed representation does not take into account the differences in friction properties of the objects being grasped, which additionally limits the maximum achievable accuracy for this task. The

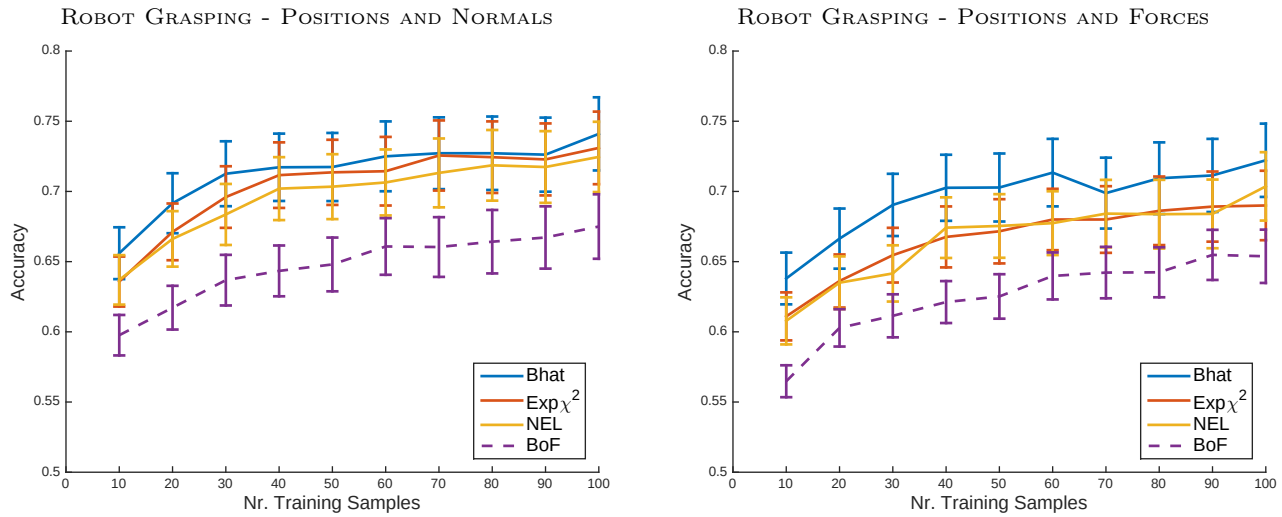


Fig. 6 The figure shows the classification accuracies for different training set size for the real robot experiment. (Left) The plot shows the results when using the contacts' positions and normals. (Right) The plot shows the results when using the contacts' positions and sensed forces before lifting the object. Error bars indicate one standard error.

contact point representations also do not take into account the robot's action space, e.g., whether the robot can apply a certain force given the current configuration.

Using force estimates instead of the contact normals resulted in a slight drop in performance for all four approaches. Force estimates allow the robot to disambiguate between certain interactions, e.g., a hand resting versus pushing against a surface. The force estimates for predicting the quality of the grasp were recorded before the object was lifted from the table. These estimates therefore do not provide information regarding the mass distribution of the object. They may however include additional irrelevant forces from the robot pushing down on the object and table, which could explain the decreased performance.

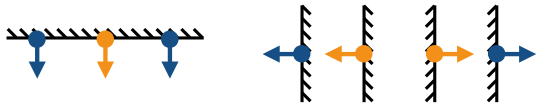
The distribution over the hyperparameter values selected by each of the kernels are shown in fig. 4. The plots reveal that the robot tended to select smaller values for the hyperparameters, especially σ_p, when using the Bhattacharyya kernel. The other approaches tend to use the larger length scales to generalize between the samples. The Bhattacharyya kernel can use a smaller length scale as its uni-modal distribution already spans a larger region of the contact space. The real robot data tended to result in larger position hyperparameter values σ_p. The normal hyperparameters σ_n for the kernel values also tended to increase. This indicates that the position information was less discriminative and the robot relied more on the normal information to differentiate between successful and unsuccessful grasps.

While the Bhattacharyya kernel is based on uni-modal Gaussian distributions, the other kernels are based on histograms and parzen windows, which can represent multi-modal distributions. Contacts for grasping often have multi-modal distributions, e.g., a pinch grasp includes two distinct regions of contacts on opposite sides of the object. One would therefore expect the multi-modal kernels to be more suitable for representing contacts. However, the results of the experiment suggest that the Bhattacharyya kernel may be better at representing the interactions afforded by these sets of points.

To illustrate this point, we can consider the sets of orange and blue contacts shown in Fig. 7. Using a uni-modal contact distribution increases the similarity of the orange and blue contacts in both scenarios we have $k_{\text{Bhat}}(\boldsymbol{X}_A, \boldsymbol{X}_B) > k_{\text{NEL}}(\boldsymbol{X}_A, \boldsymbol{X}_B)$, although both kernels can still differentiate between the sets of contacts $k(\boldsymbol{X}_A, \boldsymbol{X}_B) < 1$. For the scenario on the left (i), the two blue contacts can apply similar forces to the surface as the orange contact between them. For the second scenario (ii), both sets of contacts can pinch the object about the same point. Hence, even though the orange and blue contacts are different, they afford similar interactions. The Bhattacharyya kernel is better at capturing these similarities between sets of contacts. The kernel's ability to generalize between these types of variations in contacts could explain its higher accuracy for small training sets.

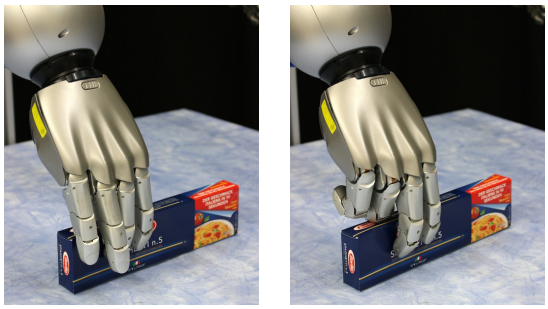
3.2 Lifting an Elongated Box

The second experiment involved predicting whether a given grasp could be used to steadily lift an elongated object. In this experiment, we explore the effects of us-



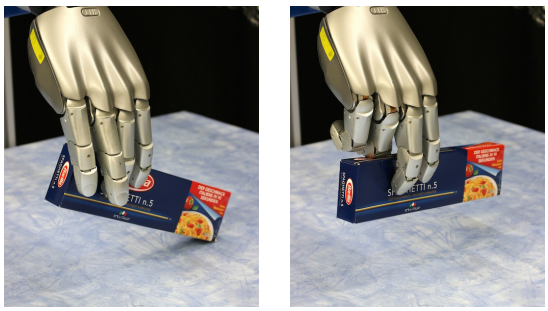
$k_{\text{Bhat}}(X_A, X_B) > k_{\text{NEL}}(X_A, X_B)$

Fig. 7 The illustrations show sets of orange and blue contact points and their corresponding normals. The hashed black lines indicate surfaces of objects. The kernel value between the contacts is greater when using the Bhattacharyya kernel, which is based on a unimodal distribution, than the NEL kernel, which is based on a multi-modal distribution.



3-Fingered Grasp 4-Fingered Grasp

Fig. 8 The two types of grasps that were used during the lifting experiment. The three-fingered grasp uses the tips of the thumb, middle, and index fingers to pinch the object. The ring and little finger are not touching the box. The four-fingered grasp additionally uses the back of the ring finger on the top of the box to provide additional support.



Failed Lift Successful Lift

Fig. 9 Examples of failed and successful lifts. A lift was considered a failure if the object was still touching the table at the end of the trial.

ing object-relative or hand-relative interaction frames, as well as the influence of incorporating the contact normal estimates. This systematic evaluation, in contrast to our previous work (Kroemer and Peters, 2014), allows us to observe the effect of the interaction frame on the contact normal's influence.

3.2.1 Experimental Setup

The robot performed 60 randomly selected grasps along the length of a spaghetti box. We specifically selected an object with a simple extruded shape such that the robot has a continuum of potential grasp locations, but only some of them will result in successful lifts (Laaksonen et al, 2012). The first 30 grasps were performed with a three-fingered grasp and the other 30 were executed with a four-fingered grasp, as shown in Fig. 8. The four-fingered grasp used the back of the ring finger for additional support. The robot subsequently tried to lift the box 13cm above the table. Lifting the box was considered successful if the object was no longer in contact with the table, and a failure otherwise, as shown in Fig. 9. The 13cm lift allows the robot to clearly differentiate between the large and small rotations observed in the left and right pictures of Fig. 9 respectively. The small rotation of the box in the picture on the right is due to the weight of the box and the compliance of the robot's fingers.

Before lifting the box, the robot recorded the state of the scene and estimated the distribution of contacts between the hand and the box. Since the robot does not have tactile sensors, the contacts were extracted from 3D point cloud models of the object and the robot's hand. The logistic regression contact detector was trained on ten manually labeled points from one scene. The normals of the points were computed based on the local neighborhood in the point cloud (Rusu and Cousins, 2011).

We evaluated representing contact points using only positions $d = 3$ (Pos) as well as both positions and normals $d = 6$ (PosNorm). We also evaluated using hand-relative and object-relative interaction frames for this experiment. The hand-relative frame is a fixed coordinate frame located at the robot's wrist. The object-relative frame is located at the mean position of the object model's points, and the first axis is in the vertical direction of gravity. The other two axes are computed as explained in Section 2.2.

The lifting performance was evaluated using 10-fold cross validation, with six samples in each test set and training set pools of 54 samples. The robot evaluated 25 classifiers for each test set and contact representation. The grid searches over the hyperparameters, with five-fold cross validation on the training sets, were performed using $\sigma_p \in \{0.5, 1.0, 2.5, 5.0, 10.0, 15.0\}$ cm and $\sigma_n \in \{0.05, 0.1, 0.25, 0.5, 1.0, 1.5\}$. The results of the evaluation are shown in Fig. 10.

3.2.2 Discussion of the Lifting Experiment

The results of the experiment show that the robot's predictions are more accurate for the lifting task when using an object-relative contact representation. The object's extruded shape makes the task more challenging

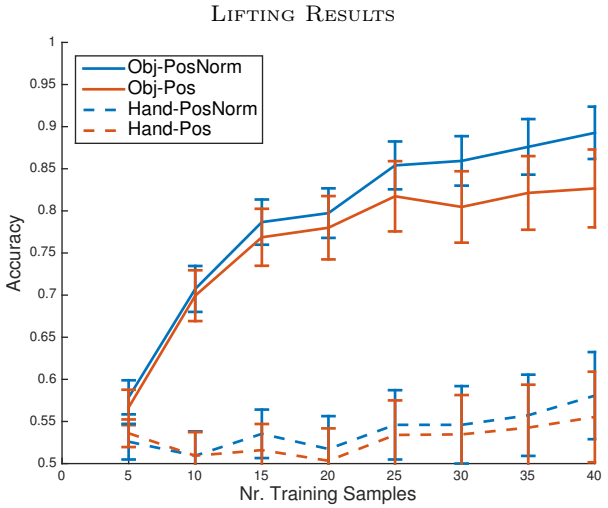
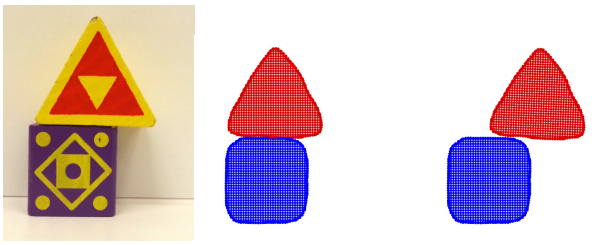


Fig. 10 The expected accuracies for the lifting task. We evaluated hand (Hand-) and object (Obj-) relative contact representations. We also investigated representing contacts using only positions (Pos) or using both positions and normals (PosNorm). The error bars indicate one standard error.

for the hand-relative representation, as the contacts appear similar regardless of the position along the box's length.

This result suggests that the accuracies for the grasping experiment in Section 3.1 could be increased by using an object-relative contact representation. However, estimating the center of the object is not trivial and would require multiple grasp attempts in a blind grasping scenario (Laaksonen et al, 2012). The objects in the grasping experiment also have a variety of unobserved masses and material properties which may also limit the accuracy of the classifier. The lifting experiment benefits from all of the samples having the same material properties. The robot could potentially include the mass and material properties explicitly by creating additional kernels for them.

Including the contact normals increased the accuracy of the classifier from 82.7% to 89.3% when using 40 training samples. The normals help the robot differentiate between the pinching contacts on the sides of the object and the ring finger's contacts on the top of the box. The contacts on the top of the box provide additional support for grasps on one half of the box, i.e., when the top contacts and the object's center of mass are on opposite sides of the pinching point. The increase in performance is smaller when using the hand-relative interaction frame, as the robot cannot determine if the center of mass is on the opposite side of the hand. The contact representation thus allows the robot to predict the success of grasps that use leverage to support the object.



Positive Example Negative Example

Fig. 11 Point cloud examples of a stable and an unstable stacking of blocks

3.3 Learning to Stack Objects

In the third experiment, the robot was given the task of classifying whether one object would support another object. We extend our previous work (Kroemer and Peters, 2014) to investigate the performances of three different structures for the Bhattacharyya kernel's hyperparameter matrix $\boldsymbol{\Sigma}_0$. The robot ultimately uses a kernel logistic regression classifier with the proposed kernel to predict placements for stacking assorted blocks into towers. Unlike the previous experiments, this experiment uses object-object contacts instead of hand-object contacts. Stacking thus demonstrates how the proposed framework can be applied to interactions where the shapes of both objects vary (Ugur and Piater, 2015; Kulick et al, 2013).

3.3.1 Classifying Stable Block Placements

The stacking dataset consists of 60 example scenes, each containing two interacting toy blocks, such as the ones shown in Fig. 11. For the 30 negative examples, physically impossible static scenes were manually created. The extruded point cloud models of the blocks were acquired using a turn table setup and a Kinect. The turn table provided additional views and more details of the relatively small blocks, which resulted in more accurate point cloud models. The positions of the interaction frames $\boldsymbol{\rho}_i$ were defined by the means of the points in the point cloud models, and the first axes $\boldsymbol{a}_i^x$ of the interaction frames were aligned with the direction of gravity. To train the contact point classifier for the 3D models, as explained in Section 2.1, a logistic regression classifier was trained on ten hand-labeled points from one scene.

In this experiment, we explored three different structures for the Bhattacharyya kernel's hyperparameter matrix $\boldsymbol{\Sigma}_0$. The Zero approach sets all of the parameters to zero. The ISO approach uses one value for all of the position dimensions and another value for all of the normal dimensions. The ARD approach allows for a separate value for each of the $d = 6$ dimensions. A

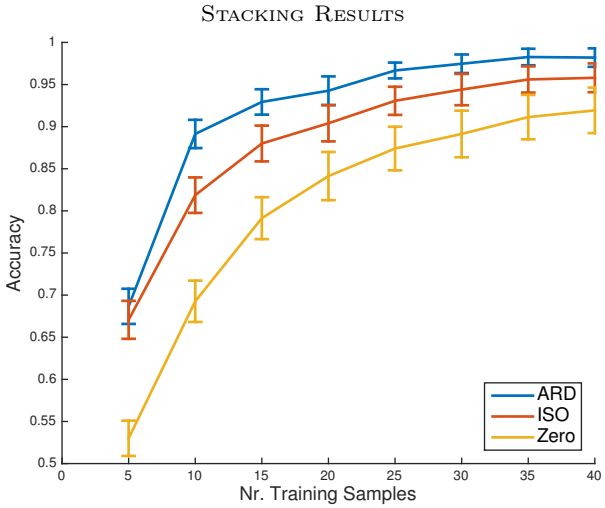


Fig. 12 The accuracy for the block stacking task. The blue line indicates the performance when using separate hyperparameters for each dimension of the contact vectors. The red line shows the performance when using one hyperparameter for all three position dimensions and another hyperparameter for the normal dimensions. The yellow line shows the accuracy when setting all of the hyperparameters to zero. The error bars indicate one standard deviation.

dimension with a relatively large hyperparameter value is considered to be irrelevant, as small variations in the contacts along this dimension will have a negligible effect on the kernel value. The hyperparameter tuning for the ARD approach thus selects the relevance of the dimensions.

Similar to the lifting experiment, the stacking performance was computed using 10-fold cross validation with six samples per test set, and training pools of 54 samples. The robot evaluated 25 kernel logistic regression classifiers for each test set, training sample size, and hyperparameter structure. The hyperparameter grid search was performed over a set of position values $\sigma_p \in \{0.0, 0.5, 1.0, 2.5, 5.0, 10.0\}$ cm and normal values $\sigma_n \in \{0.0, 0.05, 0.1, 0.25, 0.5, 1.0\}$ for the ISO approach and $\sigma_p \in \{0.0, 10.0\}$ cm and $\sigma_n \in \{0.0, 1.0\}$ for the ARD approach. Thus, the robot evaluated $6^2 = 36$ sets of hyperparameters for ISO and $2^6 = 64$ sets for ARD. The results are shown in Fig. 12.

3.3.2 Discussion of the Stacking Experiment

The experiment shows that the more flexible hyperparameter structures result in higher accuracies. Setting the hyperparameters to zero achieved the worst performance, as it implies that a small change in the contacts can greatly alter the interaction between the objects. The generalization performance is therefore limited, which leads to worse performance.

Fig. 13 examples of block towers constructed by the robot using a kernel logistic regression classifier with a Bhattacharyya kernel and the kernel's hyperparameters set to zero. The classifier was trained on 60 samples of object pairs.

Using the ARD approach leads to a higher accuracy than the ISO structure. For the stacking task, the horizontal position of the contacts is more important than the vertical position. The ARD approach results in the robot selecting larger hyperparameters for the vertical positions than the horizontal positions. By contrast, the ISO approach must find single hyperparameter settings that are suitable for all three directions.

The improved performance of the hyperparameter tuning requires more computation time, as the grid search evaluates an exponential number of parameter settings. We therefore used a smaller parameter pool for the ARD approach to keep the computation times comparable. Other methods, such as hill-climbing, could alternatively be used to perform the hyperparameter search (Kroemer and Peters, 2014). However, the grid search allows us to specify all of the evaluated hyperparameter settings for our evaluations.

Smaller hyperparameters for the position dimensions will result in lower kernel values when comparing the contacts on objects with very different sizes, e.g., a small toy block and a large toolbox. The kernel's ability to generalize between different scales is therefore limited. The stacking experiment uses blocks of similar size. Discriminating between different scales is however important for some manipulations. For example, the robot may be able to screw in a large screw by hand but not a small screw, as the larger screw allows for more torque.

The stacking task demonstrates that the proposed method can be used for manipulation tasks where the shapes of both objects vary between scenarios, e.g., tool usage. Taking into consideration the shape of both objects allows the method to be efficiently used for a wider range of interactions between objects and their environment.

3.3.3 Learning to Build Block Towers

In the final part of the experiment, the robot used a classifier to stack assorted blocks into towers. This experiment is the same as the one presented in our pre-

Fig. 14 The 25 example scenes used for the clustering experiment. The poses of the objects were estimated using detachable ARTags (removed for clarity). In each scene, we are focusing on the interaction of the object that is not in contact with the table with the object on the table.

vious work (Kroemer and Peters, 2014). We include it here for completeness and as an example of how the proposed method can be used to select object poses. The kernel logistic regression classifier with the Bhattacharyya kernel was trained using all 60 samples from the stacking experiment, and the hyperparameters $\boldsymbol{\Sigma}_0$ were set to zero. The robot was provided with a small wooden board, on which to stack the blocks. In order to avoid all of the blocks being placed directly on the board, the placing of the blocks was limited to a single strip along the middle of the board. The sequence of blocks was predefined. For every block, the robot observed the current scene using a Kinect. It used the resulting point cloud as the secondary object in the interaction. The robot does not consider the interactions between blocks further down in the stack. The point cloud of the current scene is noisy and partial, as it does not include the sides or backs of the tower's blocks. The partial point cloud is suitable for stacking, as the robot

only needs to consider contacts with the top regions of the current tower.

The robot determined a suitable placement for the current block by sampling different positions in the scene. For each sample, the contact points were estimated and the probability of the block being supported was computed using the Bhattacharyya kernel classifier. The robot subsequently attempted to place the block at the position with the highest probability.

Random sampling of block positions led to poor performance. The robot would often attempt to place the object into a partially occluded region, resulting in a collision. The robot can avoid these collisions by considering the path of the block as it is being placed. Hence, we implemented a sampling approach that mimics the movement of the block when it is being placed. The robot sampled 20 horizontal positions at 7.5mm increments across the width of the board. For each horizontal position, the robot sampled vertical placements at 5mm increments in a top-down manner until contact was detected between the block and the stack. This sampling approach improved the stacking framework's robustness. The occlusion issue could potentially also be resolved by completing the point cloud (Bohg et al, 2011; Kroemer et al, 2012a).

The robot was given the task of creating five towers consisting of five blocks each. Using the improved sampling approach, the robot successfully placed 24 of the 25 blocks without knocking any blocks down. Only one block was misplaced by a few millimeters and fell down. The robustness of the system could be further improved by also considering the success probability of neighboring positions (Boularias et al, 2011).

Examples of block towers created using the proposed method are shown in Fig. 13. The robot placed the pink arched block across gaps, as can be seen in the picture on the right. This result illustrates how the robot adapts the block's placement to both the shape of the block and the scene.

3.4 Clustering Interactions

In the final experiment, we explored using the Bhattacharyya kernel for clustering different interactions. This experiment explored object-object interactions and used 3D point cloud models to extract the contact points.

3.4.1 Experimental Setup

For this experiment, we provided the robot with 25 samples from five different types of manipulations: placing, inserting, cutting, wiping, and pushing. The scenes are shown in Fig. 14. In each scene, we identify a primary object that is not touching the table, and a secondary object that is resting on the table. Since the interactions are between objects, we provide the robot with 3D point cloud models of the objects and use the model-based approach to extract the contact points. The models were generated by placing the objects individually on a table, taking a single depth image of the object using a Microsoft Kinect, and then completing the 3D models by fitting a linearly or rotationally extruded shape (Kroemer et al, 2012a). The process results in coarse 3D models of the objects and some of the details are lost, as shown in Fig. 2. The proposed approach works well even with these coarse models.

The positions of the interaction frames are defined by the mean positions of the primary objects' points. The first axis of the interaction frame is aligned with gravity and the other axes are computed according to the contact points as explained in Section 2.2. Once the contact points have been extracted, we use the Bhattacharyya kernel to perform normalized spectral clustering as detailed in Section 2.5. We evaluated 1500 clusterings with the number of clusters κ sampled uniformly between two and ten.

The clustering of the samples depends on the kernel's hyperparameters σ_p and σ_n. Using larger values will generally result in fewer clusters. We selected the values $\sigma_p = 3.5$cm and $\sigma_n = 0.35$, which correspond to midrange values from the previous experiments' grid search. We also evaluated reducing the normal hyperparameter to $\sigma_n = 0.035$. The results of the clustering are shown in Fig. 15. The plots on the left show the kernel values between the individual samples. The plots on the right show the resulting clustering, with nonzero elements indicating the pairs of samples that are assigned to the same cluster. The robot extracted three and four clusters for the $\sigma_n = 0.35$ and $\sigma_n = 0.035$ values respectively.

3.4.2 Discussion of the Clustering Experiment

The robot was not given the labels of the five different types of manipulations presented in the scenes. However, it still managed to cluster the samples from the same types of manipulations into the same clusters for both hyperparameter settings. Some of the samples from the same interaction type would be divided into different clusters as the position hyperparameter value σ_p tends to zero. For example, if the value is close to zero, then the kernel will differentiate between contacts at different locations along a knife's edge.

When the normal hyperparameter is set to $\sigma_n = 0.35$, the robot found three clusters of interactions. The insertion and pushing samples received their own clus-

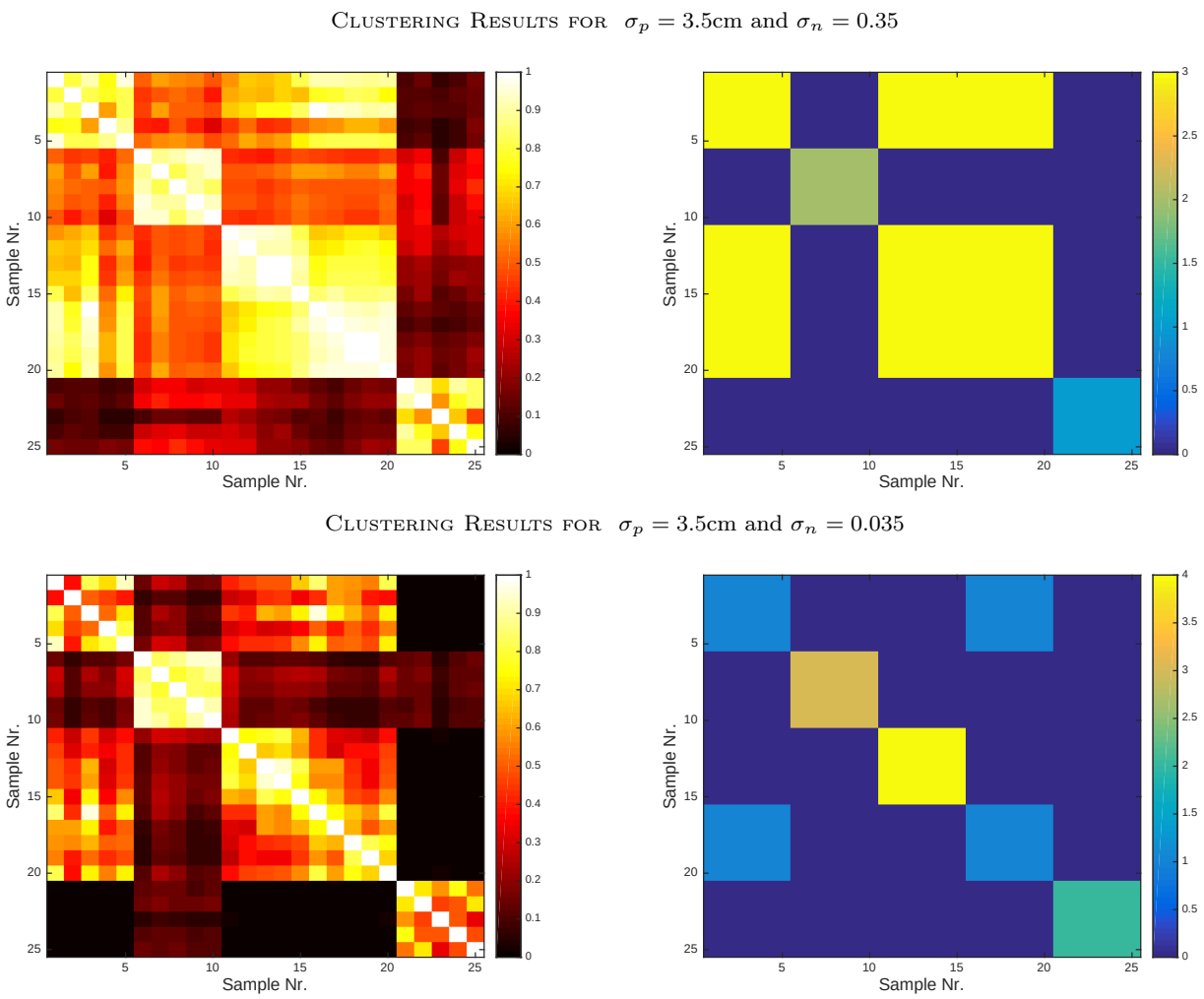


Fig. 15 The plots show the results of the clustering experiments for two sets of hyperparameter values (top: $\sigma_p = 3.5$cm and $\sigma_n = 0.35$, bottom: $\sigma_p = 3.5$cm and $\sigma_n = 0.035$). Each of the 25 rows and 25 columns of the plots correspond to a sample, as shown in Fig. 14. The plots on the left show the Bhattacharyya kernel values for the pairs of samples. The plots on the right show the resulting clusters from the normalized spectral clustering. The non-zero elements indicate pairs of samples that are assigned to the same cluster.

ters. The placing, cutting, and wiping samples were all clustered together as they all involve the primary object being on top of the secondary object. If we reduce the normal hyperparameter value to $\sigma_n = 0.035$, then the average kernel value between the cutting and placing samples is reduced from 0.63 to 0.44 and the cutting samples are given their own cluster. The smaller hyperparameter value allows the kernel to capture the higher variance in the normals around the edge of the knife and differentiate it from the large flat contacts from the placing and wiping samples.

The placing and horizontal wiping samples were clustered together for both hyperparameter settings. The contact distributions are very similar for these two manipulations. The key difference between them is the motion of the object. Hence, one could consider including the velocities of the points in the contact vectors to differentiate between these kinds of interactions. It should also be noted that wiping a vertical surface would be considered a different interaction from wiping a horizontal surface. The robot could generalize wiping over different surface orientations by defining the interaction frame according to the surface normal. However, for clustering we need a common interaction frame for all of the samples and interaction types.

Similar to the grasping experiment, the clustering does not take into account object properties such as friction or material, which could influence the types of interactions between the objects. However, the experiment has shown that the Bhattacharyya kernel can be used to cluster some different types of manipulations using spectral clustering.

4 Future Work

The proposed kernel approach allows the robot to represent and compare the geometric contact distributions between objects. In the future, we plan on extending the framework by incorporating additional information into the kernel representation. The robot may include material properties, e.g. masses and friction coefficients, or action related information, e.g., the velocities of the contact points. Information related to individual contact points could be incorporated by extending the contact vectors $\boldsymbol{x}_{ij}$. Additional object information, e.g., masses, can be captured using a separate kernel and then combined with the proposed kernel to create a new kernel (Schölkopf and Smola, 2001). Ultimately, the type of information that the robot can include will depend on the scenario

We plan to also explore using the proposed framework in different situations. For example, the robot may be able to better predict the outcome of a lift using the position and force information if it has already lifted the object slightly from the table (Chebotar et al, 2016). By combining the kernels of multiple-pairwise interactions, the robot can also use the proposed approach to represent interactions between three or more objects.

The proposed method relies on estimating contacts between objects. It thus emphasizes the need for robust contact detection and verification methods. Tactile sensors are well suited for detecting hand-object contacts, but object-object contacts are generally more difficult to detect. Verifying that two objects are in contact, and not just in close proximity, is difficult using only vision. We will therefore explore methods for detecting object-object contacts on held objects using tactile sensing (Molchanov et al, 2016), as well as acoustic cues (Griffith et al, 2012).

5 Conclusion

In this paper, we explored the problem of recognizing interactions between objects based on their contact distributions. The robot first extracts a set of contact points using either 3D point cloud models or tactile sensors. The extracted contact points are then used to compute a contact distribution, which forms the basis for a kernel function. The kernel allows the robot to compare different sets of contacts to classify the interactions.

The proposed kernel was evaluated on simulated and real grasping data, and it achieved higher accuracies than the three benchmark kernels. The robot also successfully evaluated the proposed method on lifting and stacking tasks, as well as a clustering experiment. These experiments' results show the importance of using an interaction frame that is well suited to the task and a flexible hyperparameter structure. The stacking task also showed that the proposed method can generalize between scenarios where the shapes of both interacting objects vary.

References

Abdo N, Kretzschmar H, Spinello L, Stachniss C (2013) Learning manipulation actions from a few demonstrations. In: International Conference on Robotics and Automation (ICRA)

Amores J (2013) Multiple instance classification: Review, taxonomy and comparative study. Artificial Intelligence

Bekiroglu Y, Detry R, Kragic D (2011) Learning tactile characterizations of object- and pose-specific grasps. In: International Conference on Intelligent Robots and Systems (IROS)

Ben Amor H, Kroemer O, Hillenbrand U, Neumann G, Peters J (2012) Generalization of human grasping for multi-fingered robot hands. In: International Conference on Intelligent Robots and Systems (IROS)

Bicchi A, Kumar V (2000) Robotic grasping and contact: A review. In: International Conference on Robotics and Automation (ICRA)

Bohg J, Johnson-Roberson M, León B, Felip J, Gratal X, Bergström N, Kragic D, Morales A (2011) Mind the gap - robotic grasping under incomplete observation. In: International Conference on Robotics and Automation (ICRA)

Bohg J, Morales A, Asfour T, Kragic D (2014) Data-driven grasp synthesis - a survey. IEEE Transactions on Robotics (TRo)

Boularias A, Kroemer O, Peters J (2011) Learning robot grasping from 3d images with markov random fields. In: International Conference on Intelligent Robot Systems (IROS)

Chebotar Y, Hausman K, Kroemer O, Sukhatme G, Schaal S (2016) Generalizing regrasping with supervised policy learning. In: International Symposium on Experimental Robotics (ISER)

Chen Z, Lii NY, Wimboeck T, Fan S, Jin M, Borst C, Liu H (2010) Experimental study on impedance control for the five-finger dexterous robot hand dlr-hit ii. In: International Conference on Intelligent Robots and Systems (IROS)

Dang H, Allen PK (2012) Learning grasp stability. In: International Conference on Robotics and Automation (ICRA)

Detry R, Ek CH, Madry M, Piater J, Kragic D (2012) Generalizing grasps across partly similar objects. In:

International Conference on Robotics and Automation (ICRA)

Eppner C, Brock O (2013) Grasping unknown objects by exploiting shape adaptability and environmental constraints. In: International Conference on Intelligent Robots and Systems (IROS)

Ferrari C, Canny J (1992) Planning optimal grasps. International Conference on Robotics and Automation (ICRA) pp 2290–2295

Gibson JJ (1986) The Ecological Approach To Visual Perception. Lawrence Erlbaum Associates

Goldfeder C, Ciocarlie M, Dang H, Allen PK (2009) The Columbia Grasp Database. In: International Conference on Robotics and Automation (ICRA)

Griffith S, Sinapov J, Sukhoy V, Stoytchev A (2012) A behavior-grounded approach to forming object categories: Separating containers from noncontainers. IEEE Transactions on Autonomous Mental Development 4(1):54–69

Hermans T, Li F, Rehg JM, Bobick AF (2013) Learning contact locations for pushing and orienting unknown objects. In: International Conference on Humanoid Robots

Herzog A, Pastor P, Kalakrishnan M, Righetti L, Bohg J, Asfour T, Schaal S (2013) Learning of grasp selection based on shape-templates. Autonomous Robots

Hofmann T, Schölkopf B, Smola AJ (2008) Kernel methods in machine learning. Annals of Statistics

Jebara T, Kondor R (2003) Bhattacharyya and expected likelihood kernels. In: Conference on Learning Theory (COLT)

Jebara T, Kondor R, Howard A (2004) Probability product kernels. Journal of Machine Learning Research (JMLR) 5:819–844

Jenssen R, Principe JC, Erdogmus D, Eltoft T (2006) The cauchy-schwarz divergence and parzen windowing: Connections to graph theory and mercer kernels. Journal of the Franklin Institute 343(6):614–629

Jentoft LP, Tenzer Y, Vogt D, Liu J, Wood RJ, Howe RD (2013) Flexible, stretchable tactile arrays from mems barometers. In: International Conference on Advanced Robotics (ICAR)

Jiang Y, Lim M, Zheng C, Saxena A (2012) Learning to place new objects in a scene. International Journal of Robotic Research (IJRR) 31(9):1021–1043

Kappler D, Bohg B, Schaal S (2015) Leveraging big data for grasp planning. In: IEEE International Conference on Robotics and Automation (ICRA)

Karayiannidis Y, Smith C, Vina FE, Kragic D (2014) Online contact point estimation for uncalibrated tool use. In: International Conference on Robotics and Automation (ICRA)

Kopicki M, Detry R, Adjigble M, Stolkin R, Leonardis A, Wyatt JL (2015) One shot learning and generation of dexterous grasps for novel objects. The International Journal of Robotics Research (IJRR)

Kopicki MS, Zurek S, Stolkin R, Morwald T, Wyatt JL (2011) Learning to predict how rigid objects behave under simple manipulation. In: International Conference on Robotics and Automation (ICRA)

Kroemer O, Peters J (2014) Predicting object interactions from contact distributions. In: International Conference on Intelligent Robots and Systems

Kroemer O, Ben Amor H, Ewerton M, Peters J (2012a) Point cloud completion using extrusions. In: the International Conference on Humanoid Robots

Kroemer O, Ugur E, Oztop E, Peters J (2012b) A kernel-based approach to direct action perception. In: International Conference on Robotics and Automation (ICRA)

Kroemer O, Daniel C, Neumann G, van Hoof H, Peters J (2015) Towards learning hierarchical skills for multi-phase manipulation tasks. In: International Conference on Robotics and Automation (ICRA)

Kulick J, Lang T, Toussaint M, Lopes M (2013) Active Learning for Teaching a Robot Grounded Relational Symbols. In: International Joint Conference on Artificial Intelligence (IJCAI)

Laaksonen J, Nikandrova E, Kyrki V (2012) Probabilistic sensor-based grasping. In: International Conference on Intelligent Robots and Systems (IROS)

Leischnig S, Luettgen S, Kroemer O, Peters J (2015) A comparison of contact distribution representations for learning to predict object interactions. In: International Conference on Humanoid Robots

Lenz I, Lee H, Saxena A (2013) Deep learning for detecting robotic grasps. In: Robotics: Science and Systems (RSS)

Li Q, Schürmann C, Haschke R, Ritter HJ (2013) A control framework for tactile servoing. In: Robotics: Science and Systems (R:SS)

Li Z, Sastry SS (1988) Task-oriented optimal grasping by multifingered robot hands. Journal of Robotics and Automation 4(1):32–44

Luxburg U (2007) A tutorial on spectral clustering. Statistics and Computing 17(4):395–416

Madry M, Bo L, Kragic D, Fox D (2014) ST-HMP: Unsupervised Spatio-Temporal Feature Learning for Tactile Data. In: International Conference on Robotics and Automation (ICRA)

Miller A, Allen P (1999) Examples of 3d grasp quality computations. In: International Conference on Robotics and Automation (ICRA)

Miller A, Allen P (2004) Graspit!: A versatile simulator for robotic grasping. IEEE Robotics and Automation

Magazine 11:110–122
Molchanov A, Kroemer O, Su Z, Sukhatme GS (2016) Contact localization on grasped objects using tactile sensing. In: International Conference on Intelligent Robots and Systems (IROS)
Montesano L, Lopes M, Bernardino A, Santos-Victor J (2007) Modeling affordances using bayesian networks. In: International Conference on Intelligent Robot Systems (IROS)
Ning X, Karypis G (2008) The set classification problem and solution methods. In: International Conference on Data Mining Workshops
Odhner LU, Jentoft LP, Claffee MR, Corson N, Tenzer Y, Ma RR, Buehler M, Kohout R, Howe RD, Dollar AM (2014) A compliant, underactuated hand for robust manipulation. The International Journal of Robotics Research (IJRR) 33(5):736–752
ten Pa A, Platt R (2015) Using geometry to detect grasp poses in 3d point clouds. In: International Symposium on Robotics Research (ISRR)
Roa MA, Suàrez R (2015) Grasp quality measures: review and performance. Autonomous Robots (AuRo)
Rosman B, Ramamoorthy S (2011) Learning spatial relationships between objects. The International Journal of Robotics Research (IJRR)
Rusu RB, Cousins S (2011) 3D is here: Point Cloud Library (PCL). In: International Conference on Robotics and Automation (ICRA)
Sahin E, Cakmak M, Dogar MR, Ugur E, Ucoluk G (2007) To Afford or Not to Afford: A New Formalization of Affordances Toward Affordance-Based Robot Control. Adaptive Behavior (4):447–472
Schölkopf B, Smola AJ (2001) Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, 1st edn. The MIT Press
Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans on Pattern Analysis and Machine Intelligence 22(8):888–905
Sjoo K, Jensfelt P (2011) Learning spatial relations from functional simulation. In: International Conference on Intelligent Robots and Systems (IROS)
Trinkle J, Paul RP (1990) Planning for dextrous manipulation with sliding contacts. International Journal of Robotics Research 9(3):24–48
Ugur E, Piater J (2015) Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning. In: International Conference on Robotics and Automation (ICRA), pp 2627–2633
Vedaldi A, Gulshan V, Varma M, Zisserman A (2009) Multiple kernels for object detection. In: International Conference on Computer Vision (ICCV)
Veiga F, van Hoof H, Peters J, Hermans T (2015) Stabilizing novel objects by learning to predict tactile slip. In: International Conference on Intelligent Robots and Systems (IROS)
Will PM, Grossman DD (1975) An experimental system for computer controlled mechanical assembly. IEEE Trans Comput 24(9):879–888