A Comparison of Contact Distribution Representations for Learning to Predict Object Interactions

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

***Abstract*— Different contacts between objects afford different interactions between them. For example, while contacts below an object can provide support, contacts on opposing sides can be used for pinching. Hence, a robot can learn to predict which interactions are currently afforded based on the set of contacts.**

However, representing sets of contacts is not trivial, as the number of contacts is not fixed nor are the contacts ordered. In this paper, we compare different methods for representing contacts, including bag-of-features, probability product kernels, and random forests. These approaches model the distribution over the contacts without relying on task-specific features. The methods were evaluated on both simulated grasping data, as well as real robot grasps. The random forest and the normalized expected likelihood kernel approaches achieved the highest accuracies for the simulated experiments. In the case of the real robot data, the more robust exponential χ^2 and Bhattacharyya kernels achieved higher accuracies.

I. Introduction

In order to perform a wide range of manipulations, a robot needs to be able to learn when certain interactions between objects occur. For example, the robot may need to learn when its hand is placed such that it can push an object, or when one object is being supported by another. Many interactions between objects are based on direct physical contacts. The set of afforded interactions between the two objects is determined by these contacts. A robot can learn to predict the potential interactions between objects based on their contacts.

However, representing sets of contacts is not trivial. Contacts are not ordered, nor is the number of contacts constant across different situations. Therefore, it is difficult to establish a one-to-one relationship between contacts across different situations. Instead of considering individual contacts, one could alternatively consider the *distribution* over the contacts. In this manner, the robot could first learn a model of the contact distribution, and then employ machine learning methods for classifying the distributions in order to predict object interactions.

In this paper, we present a comparison of different methods for predicting object interactions from contact distributions. We formalize the problem of predicting interactions between objects as classifying sets of contacts. The methods were evaluated in a simulated grasping experiment, similar to the one proposed in [7], as well as a real robot grasping experiment, in which the robot tries to predict the success of the specific interaction of grasping and lifting an object. Three different types of methods for representing the grasps were compared in our evaluations.

All authors are members of the Technische Universitaet Darmstadt, Germany. Oliver is also a member of the University of Southern California. Jan Peters is also a member of the MPI for Intelligent Systems.
{simon.leischnig}@stud.tu-darmstadt.de
{stefan.luettgen}@stud.tu-darmstadt.de
{oli, jan}@robot-learning.de

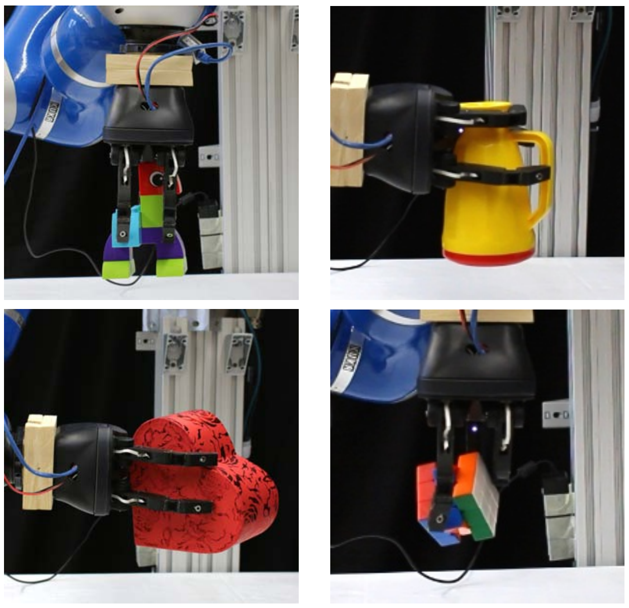
Fig. 1. Example grasps used to create the robot grasp dataset. The robot is equipped with a ReFlex hand, with tactile sensors along the finger pads and palm [26]. The robot uses the tactile data to estimate the set of contact points and normals before lifting, in order to predict the quality of the grasp.

The first type is a bag-of-features approach, which was inspired by the work of Dang and Allen [7]. In this approach, the contact distribution is represented using a dictionary of prototypical contacts, which is learned using k-means clustering. The resulting representation has a similar form to a histogram. We therefore also evaluated using an exponential χ^2 kernel between the histograms.

The second type of approach is based on probability product kernels [14], and was inspired by the work of Kroemer and Peters [18]. This approach uses a continuous density estimate to represent the contact distribution, and then computes a kernel between these densities. We evaluated two kernels for representing the contacts: the Bhattacharyya kernel and the normalized expected likelihood kernel.

The third type of approach is based on random forests [4]. Random trees learn to partition the input space and classify samples accordingly. In this manner, random trees directly combine feature learning with classification. We employ a version of the random forests that can handle sets of contacts, by using tests with two stges. The first stage selects a region of the space of contacts, and the second stage tests the proportion of contacts that fall within this region.

The remainder of this section formalizes the problem of predicting interactions from contacts and discusses related work. The different methods are described in Section II. The simulated and real robot evaluations, are given in Section III.

A. *Problem Statement*

From a machine learning perspective, we can consider 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. We refer to a group of elements as a sample. The ith sample $\boldsymbol{X}_i$ consists 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 correspond to grasps. In our experiments, we represent contacts using a $d = 6$ dimensional feature vector that includes the 3D position and the surface normal of the contact. The vector could potentially be extended with additional information, such as forces and friction coefficients. However, this information is often not readily available to the robot. The label is positive $Y_i = 1$ if the interaction is occurring between the objects for the corresponding sample, and zero otherwise. As object interactions are generally not mutually exclusive, the robot can learn to predict multiple interactions between objects by learning a separate classifier for each one.

B. *Related Work*

Learning object interactions is an important ability for robots to define symbolic states of objects [30] [31]. For example, Kulick et al [20] proposed an active learning approach for determining relational symbols between objects based on their relative positions and geometric properties. Rosman and Ramamoorthy [28] learn spatial relations between objects, e.g. on and adjacent, using k-means classifiers based on the normals of the contact points. Sjoo et al. [29] learn to select relevant features for determining spatial relations between objects using an automatic relevance determination approach.

Learning object interactions can also be used to predict whether the environment affords a certain manipulation. Jiang et al. [17] proposed a method for classifying locations to place an object based on local geometric features of the scene. Hermans et. al [11] learned locations on objects for pushing based on their overall shape and the local geometry around the pushing point. A considerable amount of work has focused on learning grasping interactions. The quality of a grasp depends on the contacts made between the hand and the object [3], [27]. Suitable contact points are often learned implicitly based on the local shape of the object [8], [12], [19] or tactile data [22], [2]. The robot can also attempt to explicitly model the locations of the contact points [7], [18].

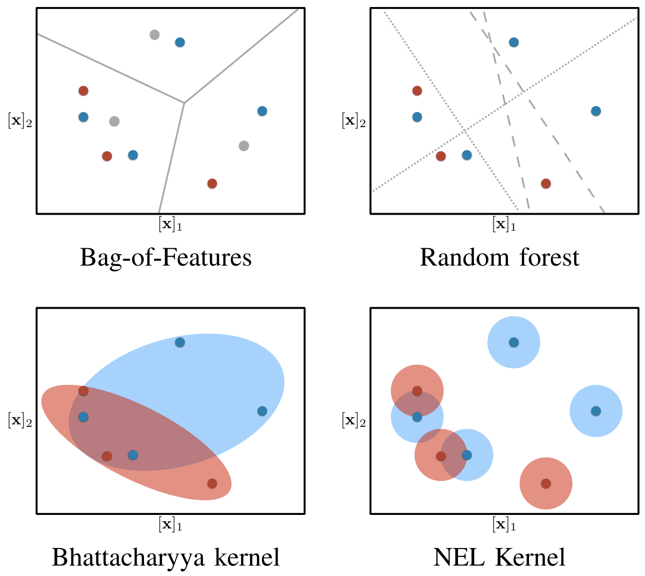

Fig. 2. The figure illustrates the different types of contact distribution representations used by the compared methods. The red and blue dots represent contact elements from two different grasps. The gray lines represent partitions of the space of contacts into discrete parts. The gray dots represent cluster centers. The ellipses represent Gaussian distributions. The red and blue dots on the left side are closer together than on the right. As a result, these contacts tend to cause the distributions to overlap more, which increases the similarity between the samples.

Thus, the robot ignores changes in the object's shape or the tactile readings that do not change the contacts.

The problem of classifying sets of contact points is closely related to multi-instance classification [1] and set classification [25] problems. However, these problem formulations generally assume that each element is associated with a class label, and the sample's label is defined by the proportions of the element's labels. This assumption does not hold for classifying sets of contacts, as the interactions often rely on the relationships between different contacts. Hence, many multi-instance and set classification methods are not applicable to this problem.

Many of the representations evaluated in this paper are based on probability density estimates, e.g., histograms, Gaussians, and kernel density estimates. These similarities are a result of modeling the proportion of contacts in different regions of the element space $\mathbb{R}^d$, and should generally not be interpreted as actual probability distributions.

II. Contact Distribution Representations

In this section, we present three different types of approaches for representing contact distributions. In Sections II-A and II-B, we present methods based on bag-of-feature and probability product kernels respectively. These representations are used together with classifiers, such as logistic regression, to predict interactions between sets of contacts. In Section II-C, we explain the random forest approach, which automatically creates features for representing sets of contacts. An overview of the different representations is shown in Fig. 2.

A. *Bag-of-Features*

The first approach is based on the Bag-of-Features method. This approach involves computing a histogram over the elements of a sample, and then using this histogram to classify the sample.

In order to use a bag-of-features representation, we first compute a dictionary $\boldsymbol{D}$ of $\tilde{n}$ prototypical elements $\boldsymbol{D} = \{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_{\tilde{n}}\}$, $\tilde{\boldsymbol{x}} \in \mathbb{R}^d$. These prototypes $\tilde{\boldsymbol{x}}$ represent the bins of the histogram according to a Voronoi decomposition of the space of elements $\mathbb{R}^d$. The prototype elements are often computed by performing k-means clustering to the set of all elements in the training set. In our experiments, we computed the set of prototype elements using k-means clustering.

Assigning an element to the nearest neighboring prototype can result in discretization effects, wherein small amounts of noise can drastically alter the assignment. In order to reduce the discretization effects, we employ a soft assignment of the elements to the bins. The weight of the jth bin for representing the ith sample is thus given by

$$P_j(\boldsymbol{X}_i) = \frac{1}{n_i} \sum_{m=1}^{n_i} \frac{\exp(-(\tilde{\boldsymbol{x}}_j - \boldsymbol{x}_{im})^T \boldsymbol{\Sigma}_0^{-1} (\tilde{\boldsymbol{x}}_j - \boldsymbol{x}_{im}))}{\sum_{l=1}^{\tilde{n}} \exp(-(\tilde{\boldsymbol{x}}_l - \boldsymbol{x}_{im})^T \boldsymbol{\Sigma}_0^{-1} (\tilde{\boldsymbol{x}}_l - \boldsymbol{x}_{im}))},$$

where $\boldsymbol{\Sigma}_0$ is a diagonal matrix of d squared length scales corresponding to the dimensions of the elements, and the assigned weights are inversely proportional to the exponential of the distance to the respective prototype. The distribution over elements in $\boldsymbol{X}_i$ can then be described by a feature vector $\boldsymbol{\phi}(\boldsymbol{X}_i) \in \mathbb{R}^{\tilde{n}+1}$, where the jth component of the vector is given by $P_j(\boldsymbol{X}_i)$, and the last component is a constant bias term of 1.

Rather than directly using the histogram weights as features, the robot can also use a exponential χ^2 kernel [13] for comparing the histograms of different samples. The kernel is given by

$$k_\chi(X_i, X_j) = \exp\left(-\sum_{l=1}^{\tilde{n}} \frac{(P_l(\boldsymbol{X}_i) - P_l(\boldsymbol{X}_j))^2}{0.5(P_l(\boldsymbol{X}_i) + P_l(\boldsymbol{X}_j))} \right)$$

This kernel is often used to compare histograms in computer vision [30], and has also been used to learn locations on objects for pushing [11].

Having defined a feature vector and a kernel for representing the contact distributions, the robot can now use them to classify interactions from contacts using standard classifiers. In our experiments, we use logistic regression and kernel logistic regression as basic classifiers.

B. *Probability Product Kernels*

Instead of a histogram, the distribution over elements in a sample can also be represented by a kernel density estimate or a Gaussian distribution. These distributions can then be used to compute probability product kernels (PPKs) between samples [14]. By directly comparing the distributions, these approaches avoid the need for computing a dictionary of prototypical elements.

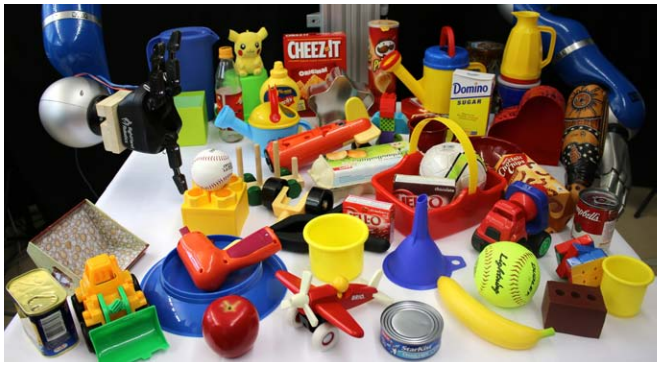

Fig. 3. The set of objects used for the real robot experiment, together with the ReFlex hand for comparison [26]. The object set was selected to contain objects with primitive and complex shapes.

The first PPK is based on a Gaussian representation of the distribution of elements within a sample. Hence, each sample $\boldsymbol{X}_i$ is represented by a mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and a covariance matrix $\boldsymbol{\Sigma}_i \in \mathbb{R}^{d \times d}$. We used the maximum-likelihood estimate to compute the mean $\boldsymbol{\mu}_i$. The covariance matrix $\boldsymbol{\Sigma}_i$ is given by the maximum likelihood estimate of the covariance of the elements in the sample $\boldsymbol{X}_i$, plus the diagonal length scale matrix $\boldsymbol{\Sigma}_0$. In this manner, the matrix $\boldsymbol{\Sigma}_0$ defines the similarity between individual samples.

After representing the samples by Gaussian distributions, the robot can compute the Bhattacharyya kernel [15] between the samples

$$k_B(X_i, X_j) = \int \sqrt{\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)} \sqrt{\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \mathrm{d}\boldsymbol{x}.$$

This kernel has a value of 1 if and only if the two distributions are equal, and it tends towards zero as the overlap between the distribution decreases. The kernel value can be computed in closed-form as

$$k_B(X_i, X_j) = C \exp\left(-M/4\right),$$

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 second PPK is based on a kernel density estimate of the distribution of elements. This approach is computationally more expensive, but allows the robot to capture more details of the distribution than a single Gaussian can. The distribution over elements for the ith sample is therefore given by

$$p_i(\boldsymbol{x}) = n_i^{-1} \sum_{j=1}^{n_i} z^{-1} \exp(-0.5(\boldsymbol{x} - \boldsymbol{x}_{ij})^T \boldsymbol{\Sigma}_0^{-1} (\boldsymbol{x} - \boldsymbol{x}_{ij})),$$

where z is a normalization constant, and $\boldsymbol{\Sigma}_0$ is again a diagonal matrix of d squared length scales corresponding to

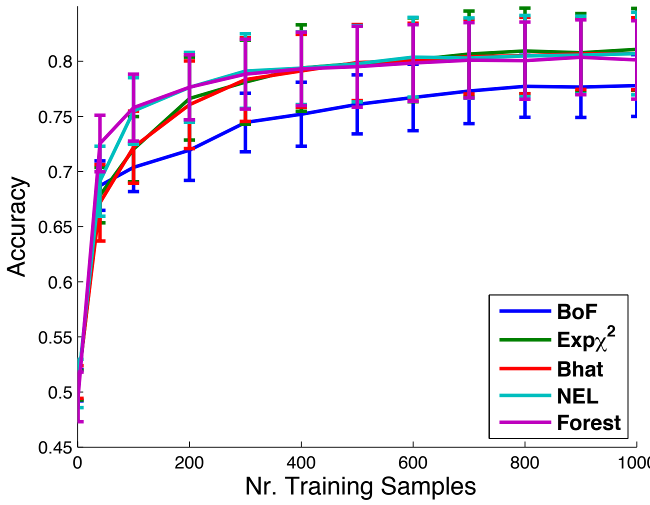

Fig. 4. The results of the simulation experiments with *large* training sets. The comparison includes bag-of-features (BoF), the exponential chi-squared (Expχ^2), the Bhattacharyya kernel (Bhatt), the normalized expected likelihood kernel (NEL), and the random forest (Forest) approaches. The error bars indicate one standard deviation.

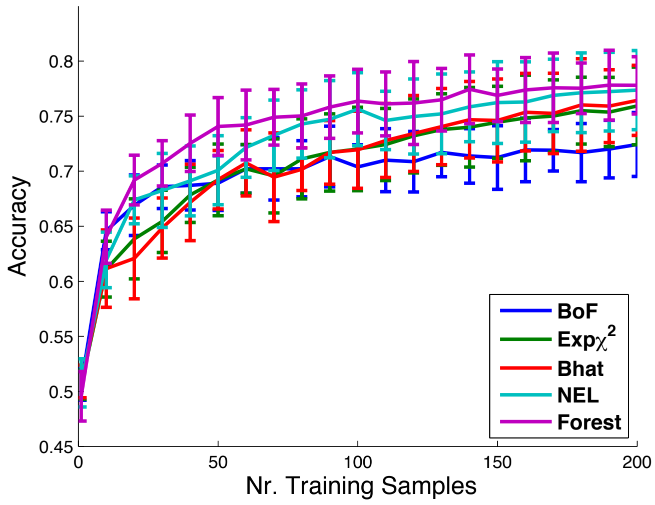

Fig. 5. The results of the simulation experiments with *small* training sets. The comparison includes bag-of-features (BoF), the exponential chi-squared (Expχ^2), the Bhattacharyya kernel (Bhatt), the normalized expected likelihood kernel (NEL), and the random forest (Forest) approaches. The error bars indicate one standard deviation.

the dimensions of the elements. We then compute a kernel between two samples as

$$k_C(X_i, X_j) = \frac{\int p_i(\boldsymbol{x}) p_j(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}}{\sqrt{\int p_i(\boldsymbol{x}) p_i(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}} \sqrt{\int p_j(\boldsymbol{x}) p_j(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}}}.$$

This kernel is closely related to the Cauchy-Schwarz divergence between probability distributions [16], and also has a value between 0 and 1. The integral $\int p_i(\boldsymbol{x}) p_j(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$ is the expected likelihood kernel [15], which we compute as

$$\sum_{k=1}^{n_i} \sum_{l=1}^{n_j} \tilde{z}^{-1} \exp(-0.25(\boldsymbol{x}_{ik} - \boldsymbol{x}_{jl})^T \Sigma_0^{-1} (\boldsymbol{x}_{ik} - \boldsymbol{x}_{jl})).$$

We therefore refer to this kernel k_C as the normalized expected likelihood kernel. The normalization factors $\tilde{z}$ do not need to be computed, as the numerator and denominator values cancel each other out when computing the kernel function.

This kernel allows the robot to directly compare the different elements between different samples, but is therefore also more computationally expensive. In our experiments, we again used kernel logistic regression to learn classifiers with the probability product kernels.

C. Random Forests

The third type of approach is to use a variant of random forests to classify the samples. Random forests employ an ensemble learning approach, and consist of multiple random trees. Each random tree is a separate classifier that provides a vote for the label of an input sample. The label with the majority of votes is then assigned to the sample. The trees are trained by randomly creating sets of tests for splitting the samples.

Each tree starts with a single node, i.e. the root node, which contains all of the training samples. In order to split the node, κ random tests are generated and evaluated. In order to handle sets of elements, we use a two-stage test for splitting the samples. The first stage is the selection stage. The robot first selects a random direction $\boldsymbol{\delta} \in \mathbb{R}^d$ and projects all of the elements in the node into this dimension. The robot then uniformly samples a threshold τ in the range of the projected elements. In this manner, the robot separates the space of elements into two halves. The sample $\boldsymbol{X}_i$ now has n_{i0} elements above the threshold and n_{i1} elements below it, such that $n_i = n_{i0} + n_{i1}$. Each sample can therefore be described by the proportion of samples above the hyperplane n_{i1}/n_i. The robot then uniformly samples a second threshold ν in the range of these values. The test is considered positive for sample X_i if $n_{i1}/n_i > \nu$, and negative otherwise.

Once the κ tests have been generated and the samples evaluated, the robot selects the test that best divides the data. There are various criteria that one can use to select the test [5]. In our experiments, we used the variance reduction. Given that the test divided the samples into a positive set of samples η^+ and a negative set η^-, then the test's score is

$$\text{Score}(\boldsymbol{\delta}, \tau, \varepsilon, \nu) = -\left|\eta^+\right| \text{Var}_{i \in \eta^+}(Y_i) - \left|\eta^-\right| \text{Var}_{i \in \eta^-}(Y_i)$$

where $|\eta|$ is the cardinality of the set η, and $\text{Var}_{i \in \eta}(Y_i)$ is the variance of the labels Y of the set. This score encourages the robot to use tests that divide the samples into subsets that have similar labels.

After selecting the test with the highest score, two child nodes are created. The first child node receives the positive samples of the test, and the second node receives the negative samples. A child node is split further if it contains more than α samples. If the new node contains less than α samples, then it is considered a leaf node. A leaf node is assigned positive

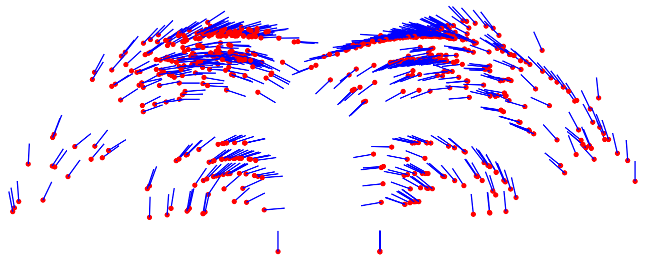

Fig. 6. A side view of all the estimated contacts that were gathered in the robot experiment. The thumb of the hand is on the left, the index and middle finger on the right. The contact positions are shown in red, and the normals are indicated by the corresponding blue lines. The contacts are defined in the hand's reference frame.

value $Y = 1$ if at least half of the samples within it are positive, and negative otherwise.

Rather than learning a single decision trees, a random forest learns a set of ρ trees. In order to classify a new sample, each tree first classifies the sample separately. Starting at the root node, the sample is evaluated by the node's test and is passed onto the first child node if the test is positive, and the second child node if it is negative. The sample is passed on until it reaches a leaf node. The label of the leaf node Y indicates the tree's vote for that tree. The random forest then assigns the label with the majority of votes to the sample.

III. Experiments

The methods presented in Section II were evaluated on both a simulated and a real robot grasping task. Grasping is a challenging benchmarking task, as the success of a grasp depends on the interactions between multiple contacts between the hand and an object. In both experiments, the robot had to predict whether a grasp was successful based on the detected contacts. The simulation and robot experiments are detailed in Sections III-A and III-B respectively, followed by a discussion of all of the results in Section III-C.

A. Simulated Grasping Experiment

The simulated grasps were obtained using GraspIt [23] and the Columbia Grasp Database [10]. This experiment was strongly inspired by the work of Dang and Allen in the context of learning stable grasps [7]. We used a simulated Barrett hand to collect 2000 grasps from 19 classes of everyday objects, including mugs, knives and other household objects. The epsilon ϵ and volume v grasp metrics were computed for each grasp [24], [9], [21]. A grasp was considered successful if and only if it achieved both $\epsilon > 0.07$ and $v > 0.1$. The task can therefore be interpreted as predicting whether a set of contacts will result in high grasp metric values. This success criterion resulted in 900 successful and 1100 failed grasps.

The robot was provided with the 3D positions and normals of all of the detected contacts between the object and the hand, resulting in $d = 6$ dimensional contact elements. For the length scale of the contacts we used $\sigma_p = 36.45$ for the position, as suggested in [7]. For the normals, we set the length scale to $\sigma_n = \sqrt{0.5}$, such that orthogonal normals are $2\sigma_n$ apart. For the random forest approach (Forest), the contact data was scaled according to these values. We also used $\rho = 30$ trees, $\kappa = 20$ random tests per split, and $\alpha = 4$ or less samples per leaf node. For the evaluation with 100 to 1000 training samples, we set the prepruning parameter α to five percent of the training sample size. For the bag-of-features (BoF) and the exponential chi-squared (exp χ^2) approaches, we used the length scales for the soft assignment of the contacts to the dictionaries, with a Gaussian weighting function. In order to create the dictionary, we used k-means clustering, with $k = 64$, applied to all of the collected contacts. For the bag-of-features representation, we also normalized all of the features as part of the preprocessing. For the Bhattacharyya (Bhat) and normalized expected likelihood (NEL) kernels, we used the length scales to define the variances $\mathbf{\Sigma}_0$ of the individual contacts.

The methods were evaluated using ten-fold cross-validation, with each fold having a test set of 200 grasps. The training sets were then randomly sampled from the remaining 1800 samples. For each fold, 10 separate classifiers were trained using different training sets, and their results were averaged. In this manner, the effects of the training set sampling are reduced. For the dictionary-based methods, a new dictionary was also computed for each set of these 10 classifiers.

The results of the simulated experiment are shown in Fig. 4 and Fig. 5. The figure shows the effects of the size of the training set on the accuracy of the different classifiers. When using 1000 training samples, the best results were obtained by the three kernel methods, which all achieved an accuracy of 81%. These were followed closely by the random forest approach, with an accuracy of 80%, and then the bag-of-features approach, which obtained a 78% accuracy. These results are comparable to those of Dang and Allen [7], which achieved an overall accuracy of 81.0% based on 24640 training examples, using a similar set of objects and the same success criterion.

B. Robot Grasping Experiment

The second experiment evaluated the methods using a set of 200 grasps collected with a real robot. The grasp attempts were taken from 50 different objects, shown in Fig. 3. The object set consists of objects with primitive and complex shapes. Many of the objects are from the YCB object database [6].

The robot's arm consists of a Kuka lightweight robot arm, and a three-fingered Right Hand Robotics ReFlex hand [26]. Both the arm and the hand are compliant. The hand is equipped with TakkTile sensors in the palm and along the fingers. Contacts were detected by thresholding the sensor values, and the contact's positions and normals were estimated using the forward kinematics of the hand. A cylindrical preshape was employed for all of the grasp attempts. The robot used a guarded motion to close the fingers. A side view of the detected contacts is shown in Fig. 6. Incorporating the normals of the contacts allows the

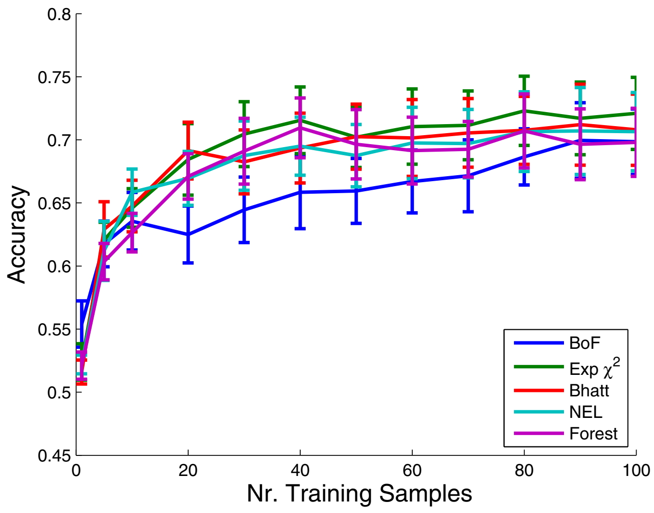

Fig. 7. The results of the real robot experiments. The comparison includes bag-of-features (BoF), the exponential chi-squared (Exp χ^2), the Bhattacharyya kernel (Bhatt), the normalized expected likelihood kernel (NEL), and the random forest (Forest) approaches. The error bars indicate one standard deviation.

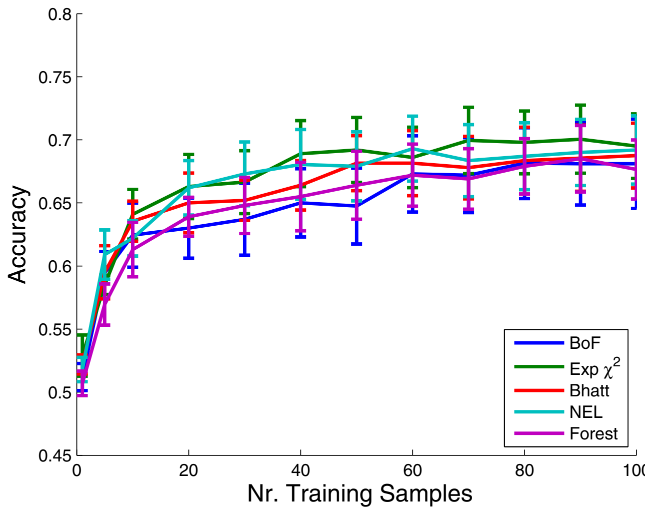

Fig. 8. The results of the real robot experiments using only the position information. The comparison includes bag-of-features (BoF), the exponential chi-squared (Exp χ^2), the Bhattacharyya kernel (Bhatt), the normalized expected likelihood kernel (NEL), and the random forest (Forest) approaches. The error bars indicate one standard deviation.

robot to more easily distinguish contacts on the thumb from those on the opposing fingers.

The contacts were recorded after the robot attempted to grasp the object, but before trying to lift it. A grasp was considered a success if it could be used to firmly lift the object from the table. If the object slid out of the hand, or turned by more than 30 degrees during the lifting process, it was considered a failed grasp. The objects were placed at different positions and orientations on the table by the human operator. Grasp attempts that failed to make contact with the object were ignored as they would be trivial to label. This approach resulted in a total of 119 successful and 81 failed grasps. The robot was not provided with any information regarding the objects' shapes, masses, or material properties.

The position length scale was set to $\sigma_p = 4.21$cm. This value corresponds to the length used in the simulated experiment, scaled according to the finger spans of the simulated and real robots. The length scale of the normals was kept the same $\sigma_n = \sqrt{0.5}$. The dictionary size was kept the same $k = 64$, as in the simulation experiment. For the random trees, we set the parameters again to $\rho = 30$, $\kappa = 20$, and $\alpha = 4$. The 10-fold cross-validation was again performed in the same manner as in the simulated experiments, except with a test set of 20 samples for each fold. The results of the experiment are shown in Fig. 7. In Fig., 8, we show the results of using only the position, and not the normal information.

C. Discussion

When we take the results of both the simulated and the real robot experiments into account, we can identify certain trends in the methods' performances. The random forests and the NEL kernel approaches achieved high accuracies in the simulated experiment, using relatively few training samples. With the adaptive basis functions of the random forest, and the flexible KDE representation of the NEL kernel, these methods can both capture a lot of details of the contact distributions. These details allow the methods to differentiate between successful and failed grasps more accurately when using the exact contact information from the simulation. For the real robot experiments, these two methods also achieved high accuracies given a small training set, but did not surpass the other methods as in the simulated experiments. This suggests that these methods may be overfitting slightly to the more noisy real robot evaluations.

In comparison, the exponential χ^2 kernel and the Bhattacharyya kernel achieved the best results for the real robot experiments. Both of these kernel methods use more coarse approximations of the contact distributions, i.e. a dictionary of elements and a single Gaussian respectively. These coarser representations may be more robust to the noise in the real robot data. The bag-of-features representation tended to learn the slowest overall, and it achieved the worst performance on the simulated experiments. This method is however the fastest to compute, and therefore presents a tradeoff between computation time and accuracy.

Overall, the differences in performance between all of the methods were minor. The performances of some of the methods could potentially be increased by tuning their respective hyperparameters. However, given that the accuracies in the simulated experiment are close to these reported by Dang and Allen [7] with tens of thousands of grasps, the robot may also be reaching a performance limit for this type of representation, i.e., the distribution of contacts.

As one would expect, the accuracies for the real robot experiments are lower than for the simulated grasps. However, considering the variety of shapes, masses, and material properties of the grasped objects, a 5% reduction in accuracy is fairly small. In order to achieve higher accuracies, the robot should take into account additional object attributes, such as the material properties, and the object's mass distribution. For improving the overall performance, evaluating the hyperparameters of the different methods also seems promising and could show a more varied performance between the kernel-based and random forest methods.

When comparing Fig. 7 to Fig. 8, we can see a drop in performance of approximately two percent when excluding the contact normals. Including the the normal information allows the representations to distinguish better between the thumb's contacts and contacts on the opposing fingers, as can be seen in Fig. 6. These additional details seem to help the robot to classify the grasps more accurately.

IV. Conclusion

In this paper, we compared different representations of contact distributions for predicting interactions, such as stable grasps. We presented methods based on bag-of-features, probability product kernels, and random forests.

These methods were compared on both simulated and real robot grasping tasks. The experiments showed that the random forests and NEL kernels could capture finer details of the contact distributions, while the Bhattacharya and exponential χ^2 kernels' coarser representations were slightly more robust to noise. The bag-of-features approach was the slowest to learn, but was the fastest to compute, and thus presents a tradeoff between accuracy and computation time.

The experiments also suggest that the robot should additionally take into consideration other object properties, such as the material and center of mass, in order to further increase the accuracy of these methods.

V. Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7-ICT-2013-10) under grant agreements 610878 (3rdHand) and 610967 (TACMAN).

References

[1] J. Amores. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, 2013.

[2] Y. Bekiroglu, R. Detry, and D. Kragic. Learning tactile characterizations of object- and pose-specific grasps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[3] A. Bicchi and V. Kumar. Robotic grasping and contact: A review. In *IEEE International Conference on Robotics and Automation*, 2000.

[4] L. Breiman. Random forests. *Machine Learning*, 2001.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[6] Berk Çalli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols. *Computing Research Repository*, abs/1502.03143, 2015.

[7] H. Dang and P. K. Allen. Learning grasp stability. In *International Conference on Robotics and Automation*, 2012.

[8] R. Detry, C. Henrik Ek, M. Madry, J. Piater, and D. Kragic. Generalizing grasps across partly similar objects. In *IEEE International Conference on Robotics and Automation*, 2012.

[9] C. Ferrari and J. Canny. Planning optimal grasps. *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2290–2295, 1992.

[10] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen. The Columbia Grasp Database. In *IEEE International Conference on Robotics and Automation*, 2009.

[11] T. Hermans, F. Li, J. M. Rehg, and A. F. Bobick. Learning contact locations for pushing and orienting unknown objects. In *IEEE-RAS International Conference on Humanoid Robots*, 2013.

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

[13] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 2008.

[14] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research*, 2004.

[15] T. Jebara and Risi Kondor. Bhattacharyya expected likelihood kernels. In *Conference on Learning Theory and 7th Kernel Workshop*, 2003.

[16] R. Jenssen, J. C. Principe, D. Erdogmus, and T. Eltoft. The cauchy-schwarz divergence and parzen windowing: Connections to graph theory and mercer kernels. *Journal of the Franklin Institute*, 2006.

[17] Y. Jiang, M. Lim, C. Zheng, and A. Saxena. Learning to place new objects in a scene. *The International Journal of Robotics Research*, 2012.

[18] O. Kroemer and J. Peters. Predicting object interactions from contact distributions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[19] O. Kroemer, E. Ugur, E. Oztop, and J. Peters. A kernel-based approach to direct action perception. In *IEEE International Conference on Robotics and Automation*, 2012.

[20] J. Kulick, T. Lang, M. Toussaint, and M. Lopes. Active Learning for Teaching a Robot Grounded Relational Symbols. In *International Joint Conferences on Artificial Intelligence*, 2013.

[21] Zexiang Li and S. Shankar Sastry. Task-oriented optimal grasping by multifingered robot hands. *IEEE Journal of Robotics and Automation*, 4(1):32–44, 1988.

[22] M. Madry, L. Bo, D. Kragic, and D. Fox. ST-HMP: Unsupervised Spatio-Temporal Feature Learning for Tactile Data. In *IEEE International Conference on Robotics and Automation*, 2014.

[23] Andrew T Miller and Peter K Allen. Graspit! a versatile simulator for robotic grasping. *Robotics & Automation Magazine, IEEE*, 11(4):110–122, 2004.

[24] A.T. Miller and P.K. Allen. Examples of 3d grasp quality computations. In *IEEE International Conference on Robotics and Automation*, 1999.

[25] X. Ning and G. Karypis. The set classification problem and solution methods. In *Eighth IEEE International Conference on Data Mining Workshops*, 2008.

[26] Lael U. Odhner, Leif P. Jentoft, Mark R. Claffee, Nicholas Corson, Yaroslav Tenzer, Raymond R. Ma, Martin Buehler, Robert Kohout, Robert D. Howe, and Aaron M. Dollar. A compliant, underactuated hand for robust manipulation. *International Journal of Robotics Research*, 33(5):736–752, 2014.

[27] M. A. Roa and R. Suàrez. Grasp quality measures: review and performance. *Autonomous Robots*, 2015.

[28] B. Rosman and S. Ramamoorthy. Learning spatial relationships between objects. *The International Journal of Robotics Research*, 2011.

[29] K. Sjoo and P. Jensfelt. Learning spatial relations from functional simulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[30] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *International Conference on Computer Vision*, 2009.

[31] Konstantinos Zampogiannis, Yezhou Yang, Cornelia Fermuller, and Yiannis Aloimonos. Learning the spatial semantics of manipulation actions through preposition grounding. In *International Conference on Robotics and Automation (ICRA)*, 2015.