Learning Robot Tactile Sensing for Object Manipulation

Lernen taktiler Robotersensorik für Objektmanipulation

Master-Thesis von Yevgen Chebotar

Februar 2014

TECHNISCHE UNIVERSITÄT DARMSTADT

Learning Robot Tactile Sensing for Object Manipulation
Lernen taktiler Robotersensorik für Objektmanipulation

Vorgelegte Master-Thesis von Yevgen Chebotar

1. Gutachten: Jan Peters
2. Gutachten: Oliver Kroemer

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 17. Februar 2014

(Yevgen Chebotar)

Abstract

Tactile sensing is a fundamental component of object manipulation and tool handling skills. With robots entering unstructured environments, tactile feedback also becomes an important ability for robot manipulation and use of tools.

In this work, we explore how a robot can learn to use tactile sensing in object manipulation tasks. We first address the problem of in-hand object localization and adapt three pose estimation algorithms from computer vision. Second, we employ dynamic motor primitives to learn robot movements from human demonstrations and record desired tactile signal trajectories. Then, we add tactile feedback to the control loop and apply relative entropy policy search to learn the parameters of the tactile coupling. Additionally, we show how the learning of tactile feedback can be performed more efficiently by reducing the dimensionality of the tactile information through spectral clustering and principal component analysis. We build a robot hand equipped with tactile sensors and implement our approach on a real robot, which learns to perform a scraping task with a spatula in an altered environment.

Contents

1 Introduction

Manipulation of objects and use of tools are among the most impressive abilities of humans. The sense of touch plays a crucial role in these tasks. Moreover, impairment of the tactile sensibility leads to a significant loss of manipulation skills as tactile information is needed for performing the sensorimotor control [JF09]. Thus, in order to be able to use tools and assist humans in unstructured environments, it is important for robots to make use of tactile feedback. When visual information is unavailable, especially due to the in-hand occlusion, tactile information also provides additional cues about the object state, such as its position and orientation. Some of the object properties, e.g. its material or internal physical state, are often only accessible through the use of tactile perception [CSPB11].

Human manipulation control is based on the prediction of sensory information and reactions to deviations from these predictions [JF09]. Human hands contain four types of tactile afferent neurons: fast-adapting (FA-I, FA-II) and slow-adapting (SA-I, SA-II). The fast-adapting afferents react to the high-frequency changes of skin deformation, e.g. during an initial contact of an object with the hand or contact of an object inside the hand with another object. During a manipulation task, the fast-adapting afferents convey changes of the task phase. The slow-adapting afferents are sensitive to low-frequency stimuli and provide cues about the object position and static forces acting on it.

In this work, we use two tactile matrix arrays attached to a parallel gripper in order to acquire tactile information of grasped objects. We divide the process of using this information for manipulation into several steps. First, we localize the object inside the hand, which is important for a correct interpretation of the tactile signal during a manipulation task. Second, we learn a robot movement and collect tactile information associated with this movement. In particular, we employ dynamic motor primitives (DMPs) [INS02] for learning a movement from human demonstration by kinesthetic teach-in. The tactile information is encoded as a desired tactile trajectory and tactile feedback is added to the system through perceptual coupling [KMP08]. Third, the parameters of tactile feedback are learned with the relative entropy policy search (REPS) reinforcement learning algorithm [PMA10]. We face the problem of high dimensionality of the tactile data and therefore perform dimensionality reduction with spectral clustering [Lux07] and principal component analysis [Jol86]. We evaluate our method by performing a gentle scraping task using a spatula as shown in Figure 1.1. Tactile feedback is used to adapt to the changes of task conditions.

1.1 Related work

Although tactile sensing has been an object of study for a long time, there has been little work on its use for manipulation tasks. A range of works has focused on recognizing various object properties. Object materials could be classified more accurately in [KLP11] by learning a lower-dimensional representation of the tactile data with the help of visual information. Additionally to material properties, Chitta et al. [CSPB11] used tactile matrix arrays attached to robot fingers to derive high-frequency tactile features for determining internal states of an object, such as detection of movements and presence of a liquid inside a container. A number of works was concerned with using tactile information for slip detection of objects inside the robot hand, e.g. [ZL12], [TKI$^+$04] and [Mel00]. The information about slip conditions was used for optimizing grasps and adjusting gripping force.

Low-resolution tactile matrix arrays have been used for object identification by Schneider et al. [SSS$^+$09]. A histogram codebook of appearances of different object parts was created for identifying objects based on similar part appearances. In this work, we use local features of object parts, such as intensity patches of tactile images, for the in-hand localization. Touch based perception has been studied by Petrovskaya et al. [PKTN07]. The authors created 3D object models and used readings of a touch sensor for Bayesian estimation of the object pose. They used the pose information for manipulating a

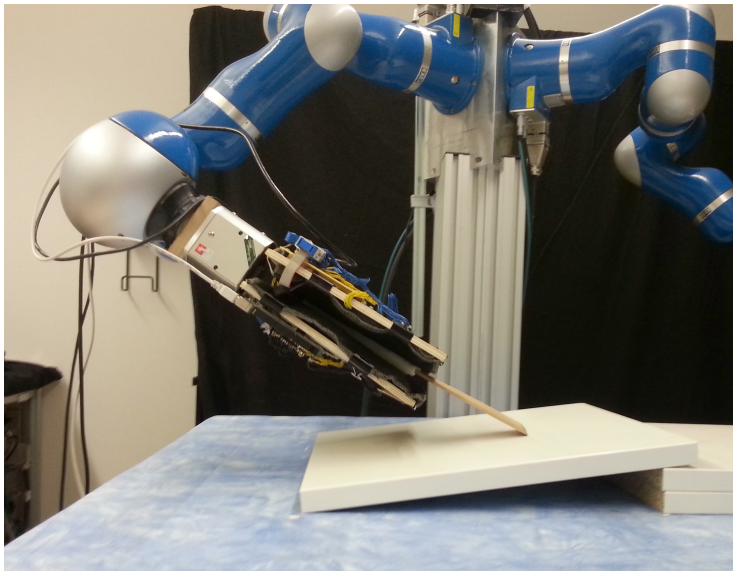

Figure 1.1: Robot performing a gentle scraping task using tactile feedback.

box and operating a door handle. In our work, besides localization, we focus on the actual dynamic tactile feedback during the task execution.

Before incorporating tactile feedback, we employ imitation learning for initializing robot movements. Imitation learning and DMPs have been extensively used in the past for performing complex movement tasks such as table tennis [MKP10], T-ball batting [PS06], grasping [KDPP10], etc. The introduction of perceptual feedback by coupling of external sensor variables to control has led to improved task performance in difficult setups, e.g. in a ball-in-a-cup task [KMP08]. In contrast to defining reward as a deviation from a static goal, such as the distance between a ball and a cup, we look at the deviation from a desired sensory trajectory. Matching of demonstrated and robot movement trajectories through reinforcement learning has shown robustness to changes of task conditions [EPDP13]. In our work, we perform matching to the desired tactile trajectory and use deviations from it to correct robot movements. The idea is also reflected in associative skill memories [PKRS12] that proposed to record stereotypical sensor signals along with the movements and use them to compute deviations from the desired behavior. However, in contrast to predefined feedback parameters, we learn optimal parameters through trial and error by applying reinforcement learning.

1.2 Structure of the thesis

The thesis is organized as follows. In Chapter 2, we describe three localization algorithms for the in-hand pose estimation. In Chapter 3, we introduce our approach for adding tactile feedback to DMPs, dimensionality reduction of tactile data and optimization of feedback parameters. In Chapter 4, we present results collected during robot experiments.

2 In-Hand Localization

In this section we describe three pose estimation algorithms for localizing objects inside the robot hand: probabilistic hierarchical object representation, iterative closest point and voting scheme. Pose estimation consists of two phases. In the first phase, a model of an object is learned from the collected tactile data. In the second phase, pose inference is performed with the help of the learned model to estimate the object pose from the new observations. Although originally developed for visual data, the presented algorithms can be used for tactile images acquired from tactile matrix arrays by performing grasps of an object. In this work, we use low-resolution tactile sensors, which imposes an additional challenge for the localization. Figure 2.1 shows an example of a tactile image of a screwdriver handle (left) and a tactile matrix array (right). In all localization algorithms, we use intensity value vectors of tactile image patches as features of the object appearance.

2.1 Probabilistic Hierarchical Object Representation

Part-based object models have been extensively used to represent objects in 2D (i.e. [FGMR10], [MRY11]) and 3D (i.e. [SSSL09], [STFW06]). Probabilistic hierarchical object representation (PHOR) is an approach developed by Detry et al. [DPP09] for creating a part-based hierarchical model of an object. The model is created iteratively beginning with primitive appearance features and propagating relative pose information to higher levels of the hierarchy. The inference is performed by using nonparametric belief propagation (NBP) [SII$^+$10]. In the following, the process is described in detail.

2.1.1 Model building

An object model is a Markov tree, in which each node corresponds to a specific part of an object. The edges of the tree encode relative transformations between parent nodes and their children. They are represented as nonparametric probability distributions of possible transformation values and are denoted as *compatibility potentials*. The bottom of the tree consists of *primitive features* that correspond to local appearances of the smallest object parts. The higher-level nodes in the hierarchy represent meta-features, i.e. object parts that are created by grouping the lower-level features. The root of the tree or the top feature represents the whole object.

The model is created iteratively in a bottom-up approach. Given a set of tactile observations, they are grouped to form bigger object parts. Spatial relationships between individual parts are preserved by encoding relative transformations in compatibility potentials on the edges of the hierarchy. The pose of each feature is defined as a probability distribution in the space of positions and orientations. The density function is defined in a nonparametric way by using kernel density estimation (KDE) [Sil86] with Gaussian kernels. We use the MATLAB implementation by Ihler [1] for performing KDE calculations.

Figure 2.2 illustrates an example of a hierarchy created from tactile observations of 50 grasps of a hammer handle. Each point at the bottom represents a single observation, i.e. presence of a tactile value greater than a specified threshold. The local appearance of each point consists of the intensity values of the surrounding tactile image patch. We instantiate or learn the model from a set of observations of the object in a single scene. This means that the pose of the object stays the same in all observations. Each observation should have its specific position and orientation as well as a distinctive appearance descriptor. In order to reduce the number of possible appearances, a codebook approach is used. In particular, the *k*-means clustering algorithm [For65] is employed to create a codebook of appearances,

[1] `http://www.ics.uci.edu/~ihler/code/kde.html`

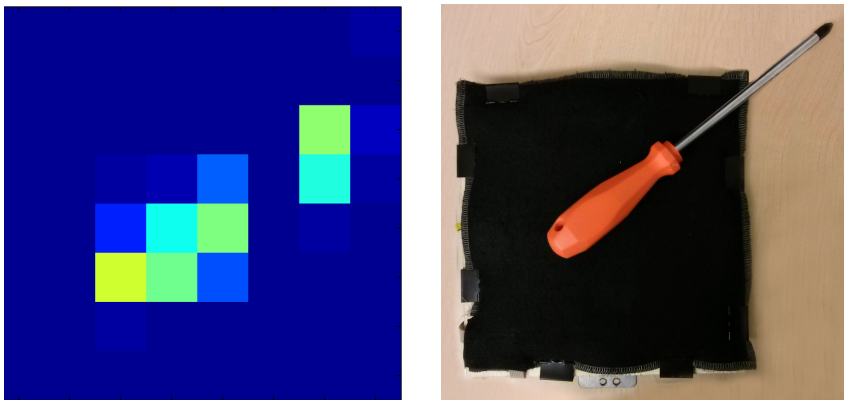

Figure 2.1: Tactile image of a screwdriver handle (left) acquired from a tactile matrix array (right).

where each appearance descriptor falls into a suitable cluster. Cluster centers are saved as a part of the model. After clustering the appearance features of observations, we compute pose distributions inside the clusters by incorporating position and orientation of each observation inside a cluster as a particle for KDE. As a result, we create primitive features or leafs of the hierarchical model tree with density functions or *observation potentials* $\phi(X_i)$. In the next step, the primitive features are randomly grouped into pairs. A new parent node or meta-feature is created for each pair. The pose of the parent is defined as the average of the poses of its children. For computing the average we sample from each children's distribution and use averages of all sample pairs as particles for the new parent density. The values of relative transformation from child to parent are used for constructing density function of compatibility potential $\psi(X_i, X_j)$ between them, where X_i is the pose of the child and X_j of the parent. After the new level of hierarchy has been constructed, the resulting meta-features are again grouped into pairs to create the next level. The algorithm terminates when there is only one meta-feature left, which represents the whole object. Individual poses of the meta-features are only needed for computing compatibility potentials and are specific to the scene. Therefore, they are discarded after the hierarchy has been constructed.

The orientation and position of the top feature depends on the choice of observation poses at the bottom of the hierarchy and can vary for different sets of observations. Therefore, after the model construction, we additionally save a correcting transformation from the pose of the top feature to a desired pose that we define manually, e.g. at the end of the tool handle. The inference on the model, as described in the next section, will return a pose to which we have to apply this transformation in order to find the desired pose.

2.1.2 Pose inference

The hierarchy as described in the previous section encodes distributions of relative transformations between different object parts. To infer the pose of the object from a new set of observation, observations are converted to primitive features and poses of higher level meta-features are iteratively computed with the help of compatibility potentials. That is, poses of the parents are computed based on the poses of their children. To propagate pose information from the bottom to the top of the hierarchy, we use NBP algorithm to send pose messages inside the hierarchy. In the following, the algorithm is described in more detail.

In the first step of the inference phase, we assign observations to clusters based on the distance of their appearance features to the cluster centers. The observation potentials $\phi(X_i)$ are then computed based on the pose distributions inside the clusters. Each child computes a message for its parent that contains its estimation of the parent's pose. The parent's pose is estimated by sampling transformation

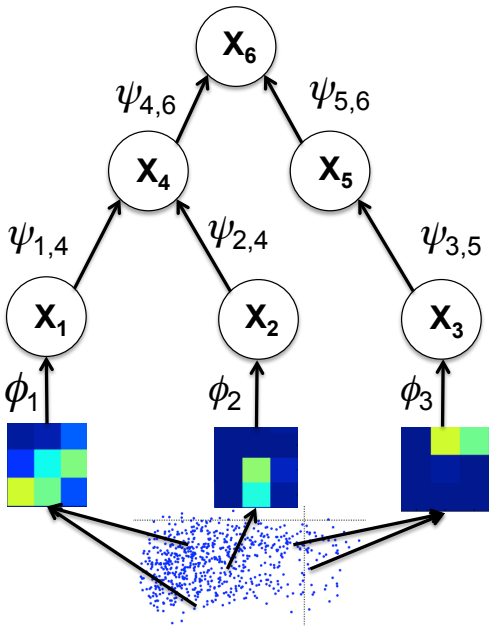

Figure 2.2: An example of a hierarchy 3x3 tactile image patches as appearance features, created from tactile observations of a hammer handle.

values from compatibility potential $\psi(X_i, X_j)$ and applying them to the current pose estimation of the child, whereas the pose estimations of the child is based on the messages from its own children. The message from node i to j is computed as follows:

$$m_{ij}(X_j) = \int \psi_{ij}(X_i, X_j)\phi(X_i) \prod_{k \in N(i) \setminus j} m_{ki}(X_i)\, dX_i,$$

where $\psi_{ij}(X_i, X_j)$ is the compatibility potential, $\phi(X_i)$ is the observation potential that is only defined for primitive features and is omitted for meta-features, m_{ki} are incoming messages. Computation of integral of the density product is in many cases intractable. Therefore, *importance sampling* is used for calculating density products [ISFW03]. The integral is computed in two steps. First, a local belief estimate is calculated with importance sampling:

$$\beta_{ts}(X_t) = \phi(X_t) \prod_{i \in N(i) \setminus s} m_{it}(X_t).$$

Here, again, $\phi(X_t)$ is only relevant for the primitive features. Then, the integral

$$m_{ij}(X_j) = \int \psi_{ij}(X_i, X_j)\beta_{ts}(X_t)\, dX_i$$

is approximated by applying transformations sampled from compatibility potential to the samples of the local belief estimate. Interestingly, the information can not only be propagated upwards but also downwards from parents to the children. Hence, this approach can also estimate poses of occluded parts of the object if we have an estimate of the pose of their parents.

The 2D pose of an object on the tactile image plane is defined on $\mathbb{R}^2 \times SO(1)$ space. In particular, the position is defined as a 2D-Vector and the orientation as an angle of rotation of the object. The parent pose is defined as the average of its children:

$$(\lambda_p, \theta_p) = (\lambda_{c1} + \lambda_{c2}, \theta_{c1} + \theta_{c2})/2,$$

where λ is the position and θ is the rotation angle.

The transformation of compatibility potential is the difference between child and parent values:

$$(\lambda_\psi, \theta_\psi) = (\lambda_c - \lambda_p, \theta_c - \theta_p).$$

Compatibility potentials are created by sampling particles from this transformation as described above. In the pose inference, the pose estimation is computed by inverting the transformation:

$$(\lambda_p, \theta_p) = (\lambda_c - \lambda_\psi, \theta_c - \theta_\psi).$$

For correcting the pose of the top feature we add the correcting transformation, as described in the previous section.

2.2 Iterative Closest Point

Iterative closest point (ICP) is a general purpose algorithm for finding a transformation that minimizes the total distance between two point clouds. It has been used to register 3D shapes, such as curves and surfaces ([BM92] and [Zha94]), object modeling and image registration [CM92]. We perform pose estimation by minimizing the distance between the point cloud created from tactile observations during the model building phase to a cloud built from observations during the pose inference phase. In the following, the localization approach with ICP is described in detail.

2.2.1 Model building

In the model building phase, tactile observations are transformed to a point cloud by placing a point whenever there is a tactile value greater than a specified threshold. For that, we assume a 2D plane with origin of coordinates at the object location and vertical axes pointing in the object direction. Object position and direction are task dependent and specified manually. Subsequently, we transform each grasp image to this plane and save point coordinates of the observations.

Additionally to the point coordinates, we save appearance vector of the local patch that is centered around this point, e.g. intensity values of the tactile image. Appearance provides additional information about tactile readings, which is otherwise lost after thresholding the tactile image intensity while creating the point cloud. The final model consists of the point coordinates with the associated appearance vectors.

2.2.2 Pose inference

Given a previously created model and a set of new tactile observations converted to a point cloud, we can run ICP for estimating the object pose by aligning both point clouds. The general outline of ICP is:

1. Initialize the total transformation T_{total} with identity rotation matrix and zero translation vector.
2. For each point in the inference point cloud, find the closest point in the model point cloud, i.e. its nearest neighbor.
3. Find the rigid transformation T, i.e. the rotation matrix and the translation vector, that minimizes the distances between nearest neighbors. This can be done with the help of the single value decomposition (SVD) [BM92] as will be explained later.
4. Transform the inference point cloud.
5. Add T to the total transformation T_{total}.

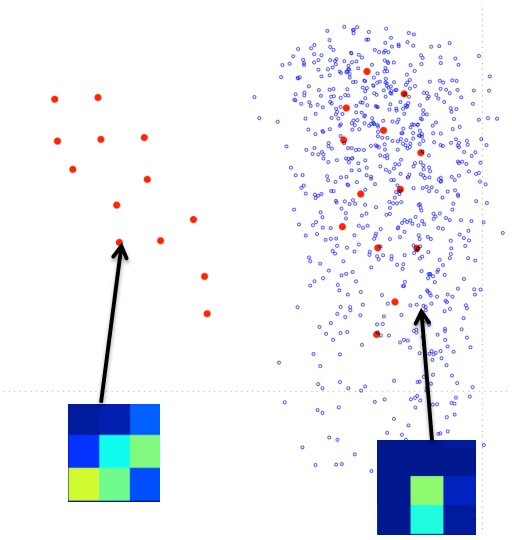

Figure 2.3: Aligning new tactile observations (red) to a model point cloud (blue) of a hammer handle. Local 3x3 patches are used as appearance features.

6. If the distance between two clouds is less than a specified threshold finish the algorithm and return T. Otherwise, continue with step 2.

Our distance measure combines the distance between point coordinates in the Cartesian space and the distance between appearance vectors associated with these points. In particular, we concatenate position vector and weighted appearance vector:

$$[\,position\_vector, w \cdot appearance\_vector\,].$$

By adjusting the weight w we can give more or less influence to the appearance compared to the spatial configuration of the object. In our experiments, we set $w = 0.001$.

Given two matrices A and B that contain the corresponding coordinates of N aligned points of the inference cloud a_i and the model cloud b_i, the rigid transformation between the points consists of a rotation matrix R and a translation vector t. It is computed by minimizing the square distance error between corresponding vectors:

$$error = \frac{1}{2}\sum_{i=1}^{N}\left\|Ra_i + t - b_i\right\|^2.$$

To compute R and t we use the SVD as described in [BM92]. First, we compute centroids or average points of both clouds:

$$c_A = \frac{1}{N}\sum_{i=1}^{N} a_i,$$

$$c_B = \frac{1}{N}\sum_{i=1}^{N} b_i.$$

Both point clouds are then translated to the origin by subtracting their centroids from all points. Then, we compute the SVD of the covariance matrix $BA^\top$ and use the resulting matrices U, S, V to compute the transformation:

$$R = UV^\top,$$

$$t = c_B - Rc_A.$$

New rotation R and translation t are then added to the total transformation:

$$R_{total} = R \cdot R_{total},$$

$$t_{total} = R \cdot t_{total} + t.$$

The angle of rotation can be estimated as

$$\theta = atan2\left(R_{21}, R_{11}\right).$$

Figure 2.3 shows an example of alignment of two point clouds of a hammer handle. Additionally, it illustrates local 3x3 patches of the points as features of the local appearance. Blue points correspond to a model point cloud with a large number of observations. Pose inference is performed by aligning red points with less observations to the model point cloud.

A disadvantage of the ICP method is that it often converges to a local optimum, especially due to a low number of points in one of the clouds. For example, in Figure 2.3 we see only one of the possible alignments of the red point cloud to the blue one. It is possible, that by beginning with a different starting point the algorithm will terminate with another alignment. In contrast, the inference in the two other algorithms is probabilistic and global convergence reduces to finding the highest mode of a distribution.

2.3 Voting Scheme

The approach of detecting objects and estimating their parameters in images by using voting schemes originates in the Hough transform [DH72]. It was first developed for identifying lines in images by transforming candidate line points in Hough space and then voting for possible slope and intercept values that could go through the single points. The distinctive feature of the algorithm is that it can detect multiple lines in the image by finding local modes of the vote distribution. Later, the algorithm was extended for arbitrary shapes, where the object form was encoded in an *R*-table during the training step [Bal81].

In this work, we take an approach similar to Glasner et al. [GGA$^+$11]. The authors employ appearance-based voting scheme for pose estimation of 3D objects in 2D images. The authors first create an object model, consisting of 3D points and appearance information of patches around the points. For pose estimation, the authors match patches of 2D images to the patches in the 3D model and collect their positions as votes for the object pose. The votes are used for constructing a nonparametric distribution, where they are converted to KDE particles with Gaussian kernels. The approach taken in this work will be described in the following sections.

2.3.1 Model building

The model of an object consists of an *appearance set* of tactile image patches and a *pose set* of relative transformations between these patches and the object frame. In the following, the pose of the object is considered to be the position of the object inside the robot hand plane and angle between the object and end-effector orientation of the robot. The relative position p_{rel} of a patch is the difference between the object position p_{obj} and the position of the patch p_{patch} (position of the patch center) in the hand plane (Figure 2.4). The absolute position of the object p_{obj} is defined manually dependent on the given task and desired localization properties. The relative angle θ_{rel} of a patch is computed by subtracting the strongest gradient orientation of the patch θ_{patch} from the object angle θ_{obj} in the hand plane:

$$p_{rel} = p_{obj} - p_{patch},$$

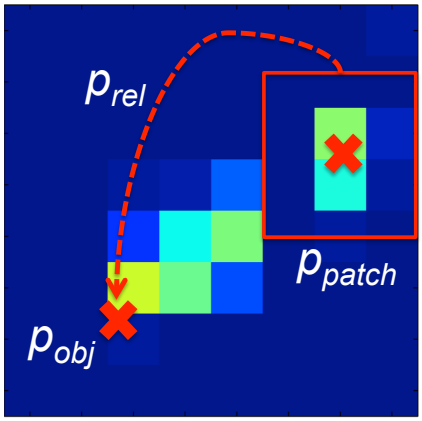

Figure 2.4: Defining a relative position p_{rel} from the absolute position of the patch p_{patch} and manually defined position of the object p_{obj}.

$$\theta_{rel} = \theta_{obj} - \theta_{patch}.$$

The model of an object contains appearance descriptors and corresponding relative transformations of all patches collected during the learning phase. A single element of the model has the form

$$[\, appearance\_descriptor,\, p_{rel},\, \theta_{rel}\,],$$

where appearance descriptor describes local appearance features of the patch, e.g. its intensity values.

2.3.2 Pose inference

For estimating the object pose from a new set of tactile observations, we calculate appearance descriptors of each patch in the tactile image and search for its K nearest neighbors in the appearance set. Then, for each found patch, we compute a vote for the object pose based on the relative transformation p_{rel} and θ_{rel} in the pose set and absolute pose of the patch:

$$p_{obj} = p_{rel} + p_{patch},$$

$$\theta_{obj} = \theta_{rel} + \theta_{patch}.$$

We construct a nonparametric distribution of the votes with KDE. The most probable poses of the object are found at the modes of the distribution. We use the mean-shift algorithm for iteratively finding all modes of the distribution [Che95]. The mean-shift update rule for computing a mode estimate in the new iteration is:

$$x_{i+1} = \frac{\sum_{j=1}^{N} k\left(\frac{v_j - x_i}{h}\right) w_j v_j}{\sum_{j=1}^{N} k\left(\frac{v_j - x_i}{h}\right) w_j},$$

where x_i is the current mode estimate, x_{i+1} is the next mode estimate, N is the total number of votes, v_j is the j-th vote, w_j is the weight of the j-th vote, $k(\cdot)$ is the kernel and h is the *bandwidth* parameter.

We use a multi-dimensional Gaussian kernel. The kernel estimate can be computed as follows:

$$k\left(\frac{v_j - x_i}{h}\right) = \exp\left(-\frac{1}{2}\left\|\frac{v_j - x_i}{h}\right\|^2\right)$$

For the angular part of the pose we replace $v_j - x_i$ with the rotational difference $\Theta(\alpha, \beta)$ of two angles, i.e. the smallest angle between them.

Each vote is inversely weighted by the distance between appearance descriptors. In particular, the weights are computed according to the distance of the patch appearance descriptor and its nearest neighbors in the model:

$$w_j = \frac{1}{d_j + \varepsilon},$$

where d_j is the distance of the j-th vote, ε is a small constant that is added in order to compensate for the cases where $d_j = 0$, which can occur in the case of a low variation of the descriptors.

3 Tactile-Based Manipulation

After estimating the object pose, manipulation tasks can be performed at different positions and orientations. In this section, we explain our method for incorporating tactile feedback into robot manipulation tasks. First, we describe dynamic motor primitives (DMPs) for encoding robot movements from human demonstration. Then, we show how tactile feedback is added to the system through perceptual coupling and how the dimensionality of the tactile information can be reduced. Finally, we explain how policy search can be used to learn parameters of the tactile coupling through trial and error with reinforcement learning.

3.1 Dynamic Motor Primitives

DMPs [INS02] are non-linear dynamical systems that consist of two components: a canonical system z and states x_1, x_2 of a spring-damper system with a forcing function f, which is driven by the canonical system. We use the following formulation of the DMPs:

$$\dot{x}_2 = \tau\alpha_x(\beta_x(g - x_1) - x_2) + \tau a f(z),$$

$$\dot{x}_1 = \tau x_2,$$

$$\dot{z} = -\tau\alpha_z z,$$

where $\alpha_z, \alpha_x, \beta_x$ are constant parameters, a is the amplitude modifier of the forcing function, τ is a time parameter for tuning speed of the movement execution and g is the goal, i.e. the final position of the movement. The value of z starts with one and approaches zero with exponential decay. The value of the forcing function also approaches zero with decreasing z and therefore, for $z \to 0$ the spring-damper system drives the position towards the goal g.

By introducing the function $f(z)$, we add a force to the spring-damper system. Hence, by choosing a correct $f(z)$ we can encode arbitrary movements. The forcing function depends on the canonical system and is formulated as

$$f(z) = \frac{\sum_{i=1}^{m} \psi_i(z) w_i z}{\sum_{i=1}^{m} \psi_i(z)},$$

$$\psi_i(z) = \exp\left(-h_i(z - c_i)^2\right),$$

where m is the number of normalized weighted Gaussian kernels with their centers distributed along the state axis z of the canonical system. At various state values z different Gaussians $\psi_i(z)$ become active and their values are multiplied by the corresponding weights w_i. The goal of the imitation learning process is to find weights w_i such that the resulting motion closely resembles the demonstration. This can be achieved with linear regression. First of all, we define a basis function vector $\boldsymbol{\phi}(z)$ and express $f(z)$ as the product of $\boldsymbol{\phi}(z)$ and the weight vector $\boldsymbol{w}$:

$$\boldsymbol{\phi}(z) = \begin{pmatrix} \frac{\psi_1(z)z}{\sum_{i=1}^{m}\psi_i(z)} \\ \vdots \\ \frac{\psi_m(z)z}{\sum_{i=1}^{m}\psi_i(z)} \end{pmatrix}, \qquad f(z) = \boldsymbol{\phi}(z)^\top \boldsymbol{w}.$$

In order to compute the required force of the demonstrated trajectory at each sampled time point, we rearrange the dynamical system. First, we rewrite the dynamical system, such that it has only a single state variable:

$$\ddot{x}_1 = \tau \dot{x}_2, \quad x_2 = \dot{x}_1/\tau,$$

$$\ddot{x}_1 = \tau^2 \alpha_x(\beta_x(g - x_1) - \dot{x}_1/\tau) + \tau^2 af.$$

The forcing function value of a single sample is

$$f = \frac{1}{a}\left(\ddot{x}_1/\tau^2 - \alpha_x\left(\beta_x(g - x_1) - \dot{x}_1/\tau\right)\right).$$

Then, we compute the squared error of all samples as

$$\varepsilon = \frac{1}{2}\sum_{i=1}^{n}\left(f_i - \boldsymbol{\phi_i}(\boldsymbol{z})^\top \boldsymbol{w_i}\right)^2.$$

Linear regression gives us the desired weights as

$$\boldsymbol{w} = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^\top \mathbf{F},$$

where $\boldsymbol{\Phi}$ is the basis function matrix and $\mathbf{F}$ is the vector of desired forces.

3.2 Perceptual coupling and tactile feedback

In order to react to different tactile stimuli, we have to modify the robot movement. For this task, we adapt DMPs with perceptual coupling as described by Kober et al. [KMP08]. In particular, we define the desired tactile trajectory $\bar{\boldsymbol{y}}(z)$ and the current tactile signal $\boldsymbol{y}$. The desired tactile trajectory is recorded during the demonstration of the movement, i.e. we record the current tactile image at each time point. Tactile feedback is based on the difference between $\bar{\boldsymbol{y}}(z)$ and $\boldsymbol{y}$. Prior to computing the feedback, tactile images have to be aligned with the images from the demonstration using a pose estimation method as described in Chapter 2.

For encoding the desired tactile trajectory $\bar{\boldsymbol{y}}(z)$ and reproducing it during the task execution, we take an approach similar to Pastor et al. [PKRS12]. We model the sensory trajectory as a non-linear dynamical system in the form of a DMP. This system is driven by the same canonical system as the motion system. In this way, both systems for the motion and tactile trajectory are synchronized and we receive the correct desired tactile signal at each time point.

To be able to differently react to the tactile information at different stages of the task execution, we need varying weights of the tactile feedback that depend on the state of the canonical system. We model these weights with normalized Gaussians distributed along the state axis. The tactile feedback term is added to the forcing function of the DMP as

$$\hat{f}(z) = \frac{\sum_{i=1}^{m}\psi_i(z)w_i z}{\sum_{i=1}^{m}\psi_i(z)} + \sum_{j=1}^{n}\left(\frac{\sum_{i=1}^{k}\hat{\psi}_i(z)\hat{w}_{ij} z}{\sum_{i=1}^{k}\hat{\psi}_i(z)}(\bar{y}_j - y_j)\right),$$

where n is the length of the tactile feedback vector and $(\bar{y}_j - y_j)$ is the difference of the j-th element of the current tactile vector and the desired tactile vector, k is the number of Gaussian kernel functions for the tactile feedback weights. For a better synchronization, we use the same number of kernels as in the original forcing function and the same Gaussian functions, i.e. $m = k$ and $\hat{\psi}_i(z) = \psi_i(z)$.

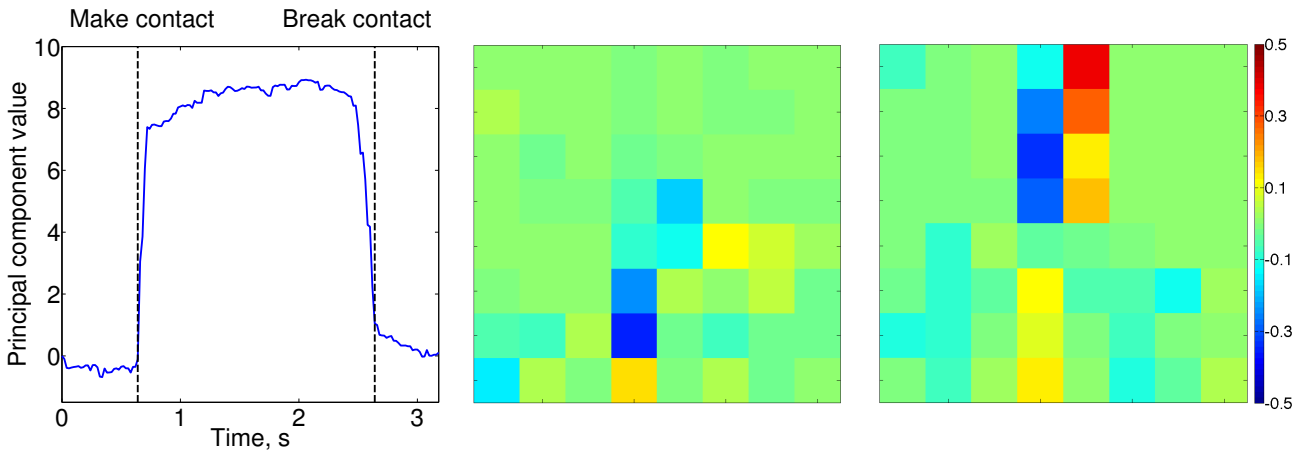

Figure 3.1: First principal component of the scraping task. *Left:* Trajectory of the tactile signal after multiplying tactile images with coefficients of the first principal component. *Middle:* PCA coefficients of the first tactile matrix. *Right:* PCA coefficients of the second tactile matrix.

3.3 Dimensionality reduction of tactile information

In the Section 3.5, we will explain how to learn weights of the tactile coupling with reinforcement learning. However, by using each tactile element (tactel) individually, the number of weights becomes very large. For example, the tactile vector of two 8x8 tactile image has the length $n = 128$. With the number of Gaussians in the model $k = 50$, the number of weights that have to be learned for a single DMP is 6400. Therefore, we perform dimensionality reduction of tactile information.

3.3.1 Principal Component Analysis

We reduce dimensionality of tactile images by applying principal component analysis (PCA) [Jol86] to the tactile image vectors collected from multiple demonstrations of the same task. It is possible to use kernel similarity measure of tactile images, as described in the next section, for performing kernel PCA (KPCA) [SSM96]. However, as we have to compute principal components in real time during the action execution, KPCA is not suitable for our task due to its high time cost.

Before applying PCA, the tactile images have to be aligned by performing in-hand localization of the manipulated object, as described in Chapter 2. After that, we subtract the mean of the tactile image vectors from all data elements and save the resulting vectors in a matrix $\mathbf{X}$. Afterwards, single value decomposition is used to find eigenvectors and eigenvalues of the covariance matrix of the data set. The matrix of the resulting principal component coefficients $\mathbf{F}$ is used to find the matrix $\hat{\mathbf{X}}$ with principal components values of the tactile image vectors: $\hat{\mathbf{X}} = \mathbf{XF}$. We use only the largest principal components of the tactile images for the feedback. Thus, only the parts of tactile images with the biggest variance throughout the task execution influence the feedback term. This significantly reduces the number of the weights that have to be learned.

Figure 3.1 illustrates the first principal component of the scraping task. Tactile images in the middle and on the right visualize the PCA coefficients. The spatula is placed vertically between two tactile matrix arrays. While touching the table, the spatula presses with different strength against various parts of both tactile matrices. In the visualization, we see which parts of the tactile images have positive and negative PCA coefficients. Figure 3.1 on the left shows the trajectory of the tactile signal after multiplying it with the coefficients of the first principal component. We observe that after contacting the table, the signal increases drastically and then, falls again after breaking the contact with the table.

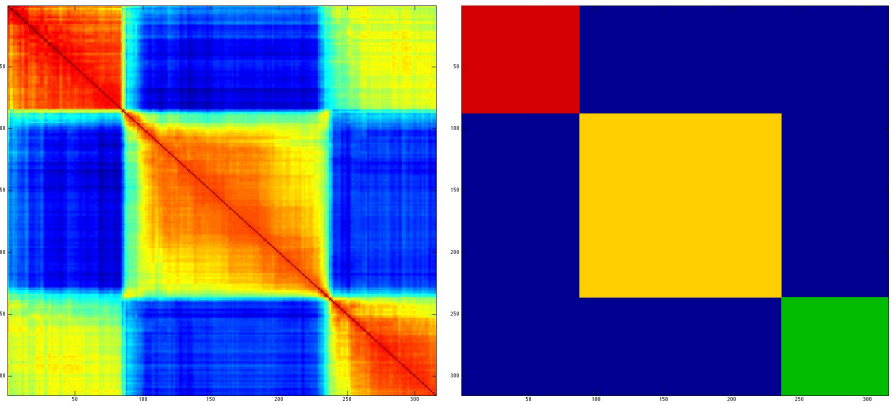

Figure 3.2: Similarity matrix heat-map (left) and spectral clustering (right) of tactile images of the scraping task. *Clusters:* red - moving towards the table, yellow - scraping along the table surface, green - moving back to the initial position

3.3.2 Spectral Clustering

Further reduction of the number of tactile feedback weights can be achieved by dividing the action into phases and learning only a single feedback weight for each phase. Recognition of the action phases can be accomplished by clustering tactile images based on their similarity. Each action phase will have similar tactile images belonging to a specific cluster. As our localization is not perfect, we often do not have exact pixel-to-pixel correspondences through the image alignment. Therefore, we employ a kernel similarity measure that takes into account and tolerates these inaccuracies.

Kernel descriptors, introduced by Bo et al. [BRF10], provide a way to compute low-level image patch features based on various pixel attributes, such as gradient, color etc. The authors use attribute and position kernels to compute similarity between two image patches. Consequently, they perform kernel principal component analysis (KPCA) [SSM96] to extract compact image features. In our work, we use the similarity of intensity values of the tactile images. The corresponding measure is computed as

$$K(P,Q) = \sum_{z \in P} \sum_{z' \in Q} k_c(c(z), c(z')) k_p(z, z'),$$

where $k_c(c(z), c(z')) = \exp(-\gamma_c \|c(z) - c(z')\|^2)$ is an intensity kernel, $k_p(z, z') = \exp(-\gamma_p \|z - z'\|^2)$ is a position kernel, P and Q are two tactile images. By changing γ_c and γ_p we can adjust the width of the Gaussians and hence, how much similarity tolerance we allow for the intensity values and pixel positions accordingly. In our experiments, we set $\gamma_c = 1$ and $\gamma_p = 2$.

Clustering of tactile images based on their kernel similarity can be performed with *spectral clustering* [Lux07]. We apply the normalized spectral clustering algorithm by Shi and Malik [SM00]. We assume a fully connected similarity graph where the weight of an edge between two images X_i and X_j is their similarity value. Subsequently, clustering is performed in a lower-dimensional space based on the eigenvalues of the similarity matrix. The general approach of performing spectral clustering with k clusters includes the following steps:

- Compute a symmetric similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, where $\mathbf{S}_{i,j} = K(X_i, X_j)$ with X_i, X_j being single tactile images.
- Compute a diagonal degree matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$, where $\mathbf{D}_{i,i} = \sum_{j=1}^{n} K(X_i, X_j)$, i.e. the total degree of the image X_i in the similarity graph as the sum of all its similarities values with other images.
- Compute a graph Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{S}$.

- Find the first k eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ of the eigenvalue problem $\mathbf{Lu} = \lambda\mathbf{Du}$.
- Construct a matrix $\mathbf{U} \in \mathbb{R}^{n \times k}$ with the eigenvectors as its columns.
- Extract rows $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^k$ of the matrix $\mathbf{U}$.
- Perform clustering of the vectors $\mathbf{y}_i$ with *k-means* algorithm in clusters $C_1, \ldots, C_k$.
- Assign image X_i to cluster C_j if $\mathbf{y}_i \in C_j$.

Figure 3.2 shows on the left a heat-map visualization of the kernel similarity matrix of the temporarily sorted tactile images acquired while performing the scraping action. We can recognize three parts of the action in the heat-map, as during these parts the tactile images are similar and therefore these portions are painted in high-temperature colors. We also notice that the first and third motion parts have high similarity values as well. They correspond to moving the hand in the air: in the first case towards the table and in the second case towards the initial position. In both cases, the tactile images are analogous. We apply spectral clustering to the similarity matrix with the number of clusters equal to three. The resulting cluster assignments of the tactile images are depicted in Figure 3.2 on the right, where clusters are painted in different colors. Hence, we are able to correctly divide the action into three segments.

After applying spectral clustering, we learn only a single effective weight θ_{js} for each action phase. The weight of each Gaussian inside a phase is then computed as

$$\hat{w}_{ij} = \frac{\theta_{js}}{c_i},$$

where j is the tactile vector index, s is the index of the action phase, i is the index of the Gaussian and c_i is the center of the Gaussian.

3.4 Tactile time series and complete action similarity

Besides being able to compute the similarity of single tactile images, we can also compute similarities between complete series of tactile images. By doing so, we are able to differentiate between various actions performed, based on the tactile signal. Although we do not employ this approach for manipulation tasks in this work, it can be used for detection of action phases and comparing tactile time series in future work.

First of all, the images have to be aligned by localizing the object inside the hand, as described in Chapter 2. Localization can be performed only once, based on the first tactile image in the series, or repeated for each element of the series. The disadvantage of the second approach is that during an action execution, tactile images can be different from the images in the localization models created from the static grip images. Hence, the localization performance will suffer from the changed setup. In general, if the object stays in the same 2D position during a task, it suffice to localize it from a single tactile image at the beginning.

By analyzing time series we face a problem of aligning the series in time because the same action can be performed with different speeds. In this case, although having a similar form, the series may not have a high similarity by directly comparing values at the same time points. We solve this problem by using Dynamic Time Warping (DTW) [KL99] algorithm with a Gaussian kernel for computing similarity of two tactile images as described in Section 3.3.2.

DTW is a dynamic programming algorithm. In contrast to the Euclidean measure for calculating the time series similarity, it looks for a best alignment of two time series, that maximizes similarity (or minimizes some cost). The algorithm proceeds iteratively by starting with the last point of both time

series and allowing insertion (duplication) and deletion of points inside a specified window. The update rule of the DTW value at time point i in the first series and j in the second time series is

$$DTW(i,j) = s(i,j) + \max \begin{cases} DTW(i+1,j), \\ DTW(i,j+1), \\ DTW(i+1,j+1). \end{cases}$$

where $s(i,j)$ is the similarity of the values at time point i of the first series and time point j of the second series.

After running the algorithm, the total similarity is saved at $DTW(1,1)$. For our problem we choose $s(i,j) = K(X_i, X_j)$, i.e. the kernel similarity of tactile images X_i and X_j. Before comparing the time series it is important to normalize them [KK03]. In this work, we normalize tactile images by subtracting from each pixel its mean across the series and dividing by its standard deviation. Such normalization leads to the time series having a mean $\mu = 0$ and standard deviation $\sigma = 1$, which removes offsets and scaling factors from the time series.

3.5 Policy search for learning tactile feedback weights

In this section, we explain how the tactile feedback parameters can be optimized with reinforcement learning. The main idea of reinforcement learning is to learn a controller, i.e. a policy of a robot, that maximizes a given reward. In this work, we employ policy search to learn weights of the tactile feedback controller. Let $\boldsymbol{\theta}$ be a vector of tactile feedback weights. We define the policy $\pi(\boldsymbol{\theta})$ as a Gaussian distribution of the feedback weights with a mean $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$. At the beginning of a single task execution, i.e. an episode, the weight vector $\boldsymbol{\theta}$ is sampled from this distribution. We compute the reward $R(\boldsymbol{\theta})$ by integrating the total deviation of the tactile signal from the desired tactile trajectory during an episode.

We perform optimization of the policy with episodic relative entropy policy search (REPS) [PMA10]. The advantage of this method is that, in the process of reward maximization, the loss of information during a policy update is bounded. This means that the algorithm stays close to the data and does not allow large changes of the policy. This approach leads to a better convergence behavior. The goal of REPS is to maximize expected reward $J(\pi)$ of a policy π subject to bounded information loss. Information loss is defined as the Kullback-Leibler (KL) divergence between the old and new policies. By bounding the information loss, we limit the change of the policy and hence, avoid premature convergence.

Let $q(\boldsymbol{\theta})$ be the old policy and $\pi(\boldsymbol{\theta})$ be the new policy after the policy update. Then, we can formulate a constrained optimization problem:

$$\max_{\pi} J(\pi) = \int \pi(\boldsymbol{\theta}) R(\boldsymbol{\theta})\, d\boldsymbol{\theta} \quad \text{s. t.}$$

$$\int \pi(\boldsymbol{\theta}) \log \frac{\pi(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}\, d\boldsymbol{\theta} \leq \epsilon,$$

$$\int \pi(\boldsymbol{\theta})\, d\boldsymbol{\theta} = 1,$$

where $J(\pi)$ is the total expected reward of using policy $\pi(\boldsymbol{\theta})$. The first constraint bounds the KL-divergence between the policies with the maximum information loss set to ϵ. The second constraint ensures that $\pi(\boldsymbol{\theta})$ is a proper probability distribution.

The optimization problem can be solved with the Lagrange multipliers method. The Lagrangian is computed as

$$L = \left(\int \pi(\boldsymbol{\theta}) R(\boldsymbol{\theta}) \, d\boldsymbol{\theta} \right) + \eta \left(\epsilon - \int \pi(\boldsymbol{\theta}) \log \frac{\pi(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} \, d\boldsymbol{\theta} \right) + \lambda \left(1 - \int \pi(\boldsymbol{\theta}) \, d\boldsymbol{\theta} \right).$$

After computing the derivative of the Lagrangian with respect to $\pi(\boldsymbol{\theta})$ and combining it with the given constraints, we can formulate the dual problem

$$g(\eta) = \eta\epsilon + \eta \log \int q(\boldsymbol{\theta}) \exp\left(\frac{R(\boldsymbol{\theta})}{\eta} \right) d\boldsymbol{\theta}.$$

The integral term can be rewritten as the expected value

$$\mathbb{E}_{q(\boldsymbol{\theta})} \left[\exp\left(\frac{R(\boldsymbol{\theta})}{\eta} \right) \right] = \int q(\boldsymbol{\theta}) \exp\left(\frac{R(\boldsymbol{\theta})}{\eta} \right) d\boldsymbol{\theta},$$

which can be approximated from samples with a maximum-likelihood estimate of the expected value. The resulting dual function for N samples is:

$$g(\eta) = \eta\epsilon + \eta \log \frac{1}{N} \sum_{i=1}^{N} \exp\left(\frac{R(\boldsymbol{\theta}_i)}{\eta} \right)$$

Furthermore, from the Lagrangian it can be derived that

$$\pi(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta}) \exp\left(\frac{R(\boldsymbol{\theta})}{\eta} \right).$$

Therefore, we can compute the new policy parameters with a weighted maximum-likelihood solution. The weights are $\exp\left(R(\boldsymbol{\theta})/\eta\right)$, where rewards are scaled by η, which can be interpreted as the temperature of a soft-max distribution. By decreasing η we give larger weights to the high-reward samples. Increasing of η results in more uniform weights. The parameter η is computed according to the optimization constraints by solving the dual problem.

Given a set of feedback weight vectors $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$ and corresponding episode rewards, the policy update rules for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ can be formulated as follows [DNP13]:

$$\boldsymbol{\mu} = \frac{\sum_{i=1}^{N} d_i \boldsymbol{\theta}_i}{\sum_{i=1}^{N} d_i}, \quad \boldsymbol{\Sigma} = \frac{\sum_{i=1}^{N} d_i \left(\boldsymbol{\theta}_i - \boldsymbol{\mu}\right) \left(\boldsymbol{\theta}_i - \boldsymbol{\mu}\right)^{\top}}{\sum_{i=1}^{N} d_i},$$

with weights $d_i = \exp\left(R(\boldsymbol{\theta}_i)/\eta\right)$.

4 Experiments

In this section, we first describe the hardware used in our robot experiments. Afterwards, we present experimental results of the in-hand object localization, tactile time series analysis and tactile-based manipulation.

4.1 Hardware

In this work, we aimed at building a low-cost robot gripper equipped with tactile sensors. We used two dynamical matrix analog pressure sensors by *PlugAndWear.com*. The outer sides of the sensor are made of conductive fabric and the inner layer is made of a piezo-resistive material that changes its resistance when the fabric is pressed. The matrix has a sensitive area of 16cm x 16cm and contains 64 sensor cells in 8 rows and 8 columns. Hence, the size of a single cell and consequently the spatial resolution is 20mm. The detectable pressure range varies from 1.8 kPa to 0.1 MPa.

The values of the sensor can be recorded by supplying one matrix column at a time with voltage and measuring output voltage of each of 8 column elements by using pull-down resistors. The resistance of each tactile matrix element and the resulting pressure value can be calculated as follows:

$$R(V) = \frac{V_{cc} - V_{out}}{V_{out}/R_{pd}},$$

$$P(R) = aR^{-b} + c,$$

where V_{cc} is the input voltage, R_{pd} is the resistance of the pull-down resistor, V_{out} is the output voltage, a, b, c are coefficients that are specific for different resistance values of the pull-down resistor. The coefficient b is always negative, such that the measured pressure falls exponentially with the sensor resistance.

We mounted two tactile matrix arrays to a parallel gripper. In order to improve compliance and involve more sensor elements during an object contact, we put a 5mm foam layer underneath the sensor array such that the fabric of the sensor stretches when pressed. Furthermore, to increase the effective resolution of sensing we placed both sensors with a 10mm shift to each other, which is a half of the size of a single tactile element.

In addition to the tactile sensor, we attached two accelerometers to both sides of the hand. We used the *BMA020* sensor that allows measuring acceleration in the ranges $\pm 2g, \pm 4g$ and $\pm 8g$. In order to be able to determine the opening distance of the hand, we attached a potentiometer slider between the two sides of the robot hand.

We connected sensors to two *Arduino Mega 2560*[1] micro-controller boards. The sensor array columns were connected to digital outputs and rows to analog inputs with pull-down resistors. The sensor data was read by sequentially turning on each digital output and measuring voltage on the analog inputs. The sensor values were then written to the serial output of Arduino and could be read out by connecting it to a computer. To process and visualize the data we used MATLAB serial communication and visualization packages. The accelerometer was connected through I^2C bus to Arduino. Figure 4.1 shows the constructed gripper with its individual components.

With the given setup, the tactile data could be captured with a frequency of about 50Hz. We attached the gripper to a light-weight KUKA arm with 7 degrees of freedom. We used SL simulation and real-time control software package [Sch09] to run our algorithms on the robot and in simulation. Communication of the data between SL and MATLAB was implemented in the form of a TCP server.

[1] `http://arduino.cc/de/Main/ArduinoBoardMega2560`

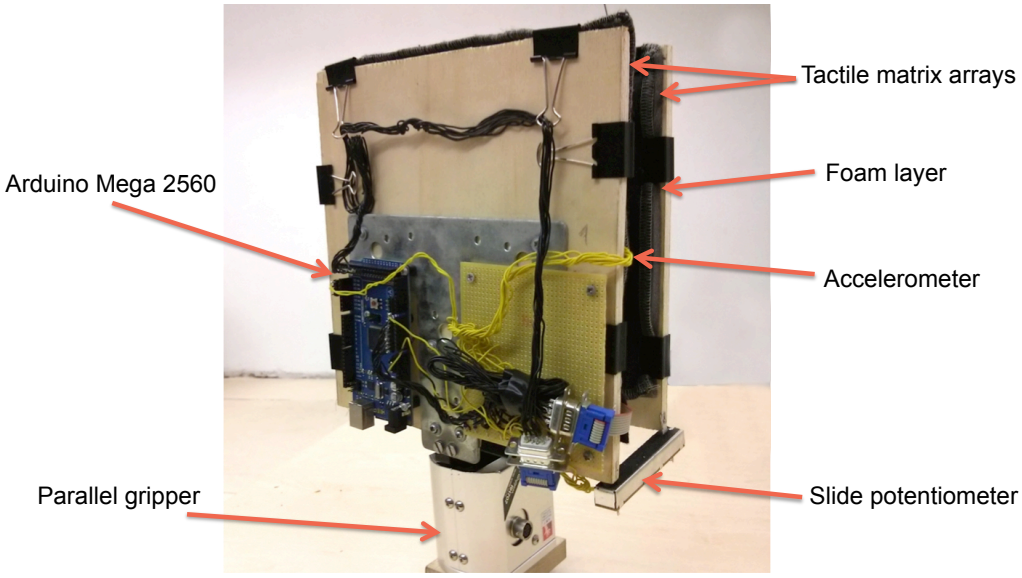

Figure 4.1: Components of the robot gripper.

4.2 In-hand localization experiments

We evaluated the performance of the pose estimation algorithms on the tactile data acquired by performing grasps of several objects: a hammer, a screwdriver, a roll of tape, a saw and a spatula. Appearance features were extracted from intensity values of 3x3 tactile image patches.

4.2.1 Experimental setup

Before collecting the tactile data, the object was placed at a known position. In this way, we were able to derive object position inside the hand based on the position of the end-effector. Subsequently, we performed 40 grasps of each object from different angles and saved the tactile images along with the poses of the object inside the hand. We applied the same gripping force in all experiments. We randomly chose grasping angles, such that they were accessible by the robot hand.

In order to be able to map single tactels to coordinates inside the robot hand, we calibrated positions of each element of the matrix array. For this, we used a small object (a ball) to activate one tactel at a time and calculate position of this tactel based on a known position of the hand. After that, we performed linear regression to find coefficients m_1, b_1 and m_2, b_2 for the coordinate equations

$$x = m_1 \cdot col + b_1,$$

$$y = m_2 \cdot row + b_2,$$

where col is the column of the activated tactel and row is its row. After calibration, we noticed that the spacing of tactels varied between both sensors. The tactels of the first sensor were distributed at 2.12cm distance in x direction and 2.00cm distance in y direction. The second sensor was smaller and tactels had 1.89cm and 1.91cm distances between each other in x and y directions accordingly.

4.2.2 Results and discussion

The localization performance was evaluated by running leave-one-out cross-validation. That is, before performing pose inference on the hold-out grasp, we instantiated the model with the 39 other grasps.

	PHOR		Voting Scheme		ICP	
Object	***Pos.***	***Angle***	***Pos.***	***Angle***	***Pos.***	***Angle***
Hammer	4.09	22.54	**1.37**	**11.83**	2.19	21.89
Screwdriver	4.55	23.13	**1.56**	**10.27**	3.21	30.76
Roll of tape	3.52	16.79	**2.12**	**12.91**	2.25	26.27
Saw	3.94	22.44	**2.19**	**6.65**	2.99	26.81
Spatula	4.74	29.96	**2.96**	**22.67**	3.11	28.51
Average	4.17	22.97	**2.04**	**12.87**	2.75	26.85

Table 4.1: Mean absolute errors of pose estimation with leave-one-out cross-validation of 40 grasps. Position: *cm*, Angle: *degree*.

	PHOR		Voting Scheme		ICP	
Object	***Pos.***	***Angle***	***Pos.***	***Angle***	***Pos.***	***Angle***
Hammer	1.55	9.24	**0.66**	**9.15**	1.80	9.89
Screwdriver	1.54	**9.63**	**0.55**	10.20	2.44	14.25
Roll of tape	1.49	**6.97**	**0.60**	8.71	1.49	10.08
Saw	1.91	16.78	1.55	**2.94**	**1.51**	12.89
Spatula	2.09	17.61	**1.80**	21.69	2.35	**16.64**
Average	1.72	12.05	**0.90**	**10.54**	1.92	12.75

Table 4.2: Mean absolute errors of pose estimation with leave-ten-out cross-validation of 40 grasps. Position: *cm*, Angle: *degree*.

Table 4.1 shows the mean absolute errors of the position and angle estimation for all methods and all objects. We could achieve the best results on leave-one-out cross-validation with the voting scheme. The main reason for that is that the other two methods required a larger number of observation in order to make reliable pose predictions. In fact, the number of tactile observations from a single grip was in the range from 11 to 32. When using PHOR, the data was distributed among the primitive features and the resulting amount of observations was too low for building reliable KDEs. The generated KDEs had much noise, which was propagated to the top of the hierarchy. The voting scheme, on contrary, used only a single KDE for all observations, which had enough data for a reliable pose inference.

The convergence of the ICP method also heavily depends on the amount of observations, i.e. the number of points to be aligned. Due to the low amount of the points, the number of possible alignments increases. Therefore, although we could improve the alignment with appearance features, in many cases ICP did not converge to a correct alignment and terminated in a local optimum.

Using more observations leads to a significant increase of the localization performance. Table 4.2 shows pose estimation results after combining tactile observations of 10 grasps for pose inference and creating models from the other 30 grasps. The increased number of grasps simulates a higher resolution of the tactile matrix. Improvement of results of PHOR and ICP methods confirms that they perform better with a bigger amount of observations. Here, all algorithms had an average positional error under 2cm, the voting scheme even under 1cm. The angular error was in the 10-13 degree range.

The lowest performance was observed for the spatula. It was the thinnest object, which led to a situation where the tactile sensors of both hands could touch each other during the grip and generate noise. The problem of detecting contacts between different parts of the robot hand, i.e. self-collisions of

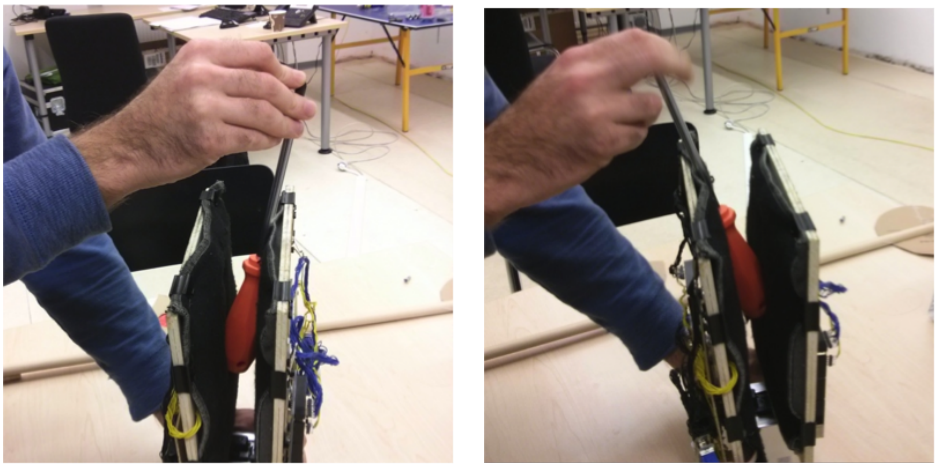

Figure 4.2: Pushing (left) and pulling (right) screwdriver actions inside the robot hand.

the robot, should be addressed in the future. Furthermore, localization of objects that are much bigger than the hand remains an open problem, as only a small object part can be observed at a time. In our experiments, a frequent kind of a localization error was a 180-degree flipping of objects with long straight handles, such as a hammer or spatula, due to the high similarity of the corresponding tactile images. This problem can be addressed by using vision for determining the approximate object pose based on its appearance outside of the hand, and refining the localization with tactile data.

For the object manipulation task, we need a good localization performance with tactile data from a single grasp. As the voting scheme was the only method that could provide reliable pose estimation from a small amount of data, we used this method in our manipulation experiments.

4.3 Tactile time series experiments

We evaluated the discriminatory power of the time series similarity measure by performing two actions inside the robot hand: pushing a screwdriver handle from one hand side to another (Figure 4.2 left) and pulling it or pushing from the other side (Figure 4.2 right).

4.3.1 Experimental setup

Both actions were performed at different angles and positions. Therefore, before computing the similarity of tactile series, the screwdriver was localized inside the hand and tactile images were aligned. We only performed localization in the first tactile image of the series. After localizing, we computed a transformation needed to align tactile image to a predefined position.

Additionally, first tactile image, i.e. the image before the begin of the action, was subtracted from all subsequent images in the series, such that the variance of the initial gripping force did not influence the calculation. This means, we compared changes of the tactile images in relation to the first image.

4.3.2 Results and discussion

Figure 4.3 shows the similarity heat map of 60 time series of two actions performed inside the robot hand at different angles, positions and velocities. First half corresponds to the action of pushing the screwdriver handle and second to pulling it. The tactile images were normalized as described in Section 3.4. We see a clear distinction between both actions. Moreover, although having inexact localization and

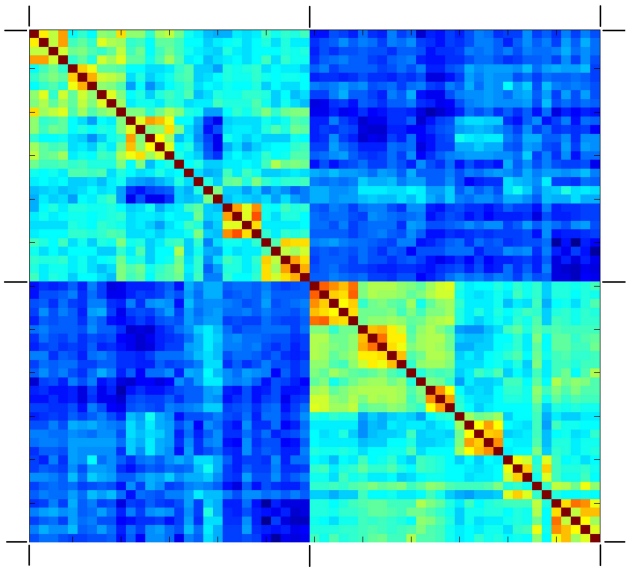

Figure 4.3: Tactile time series similarity heat-map of *Push screwdriver* (first half) and *Pull screwdriver* (second half) actions.

often different scaling of tactile values by performing action at different positions, the similarities inside same action blocks do not show a large variance.

We observe larger similarity values around the diagonal of the similarity matrix. In many cases, the position of the screwdriver was similar in adjacent experiments. This led to a similar pose estimation and after alignment, to a high similarity of the tactile images.

4.4 Tactile-based manipulation experiments

To evaluate the proposed approach for learning manipulation skills with tactile feedback, the robot was given the task of gently scraping a surface with a spatula.

4.4.1 Experimental setup

After learning the scraping movement from human demonstration, we introduced two kinds of variations from the original setup. In the first task, we used a higher table than the robot had expected by adding a surface elevation of 5cm. The goal of the robot was to adapt to the new height and stay close to the desired tactile trajectory by correcting the pressure of the spatula on the table. In the second task, we placed a ramp on the table. Similarly, the goal of the robot was to learn to adjust its tactile feedback to the dynamically changing height of the surface.

In both tasks, we used the first two principal components of the desired tactile trajectory. For each principal component, we learned three weights for the three phases of the scraping action. The robot movements were encoded in a three dimensional Cartesian space with a separate DMP for each dimension. Thus, the total number of tactile feedback weights to optimize with REPS was 18. We conducted the complete policy learning experiments three times for both tasks. Each experiment included 6 sets of 20 episodes with resulting policy updates between the sets.

4.4.2 Results and discussion

Figure 4.4 shows development of the mean reward value after each policy update of all experiments on an elevated surface and a ramp. Additionally, the figures show the standard errors of the reward

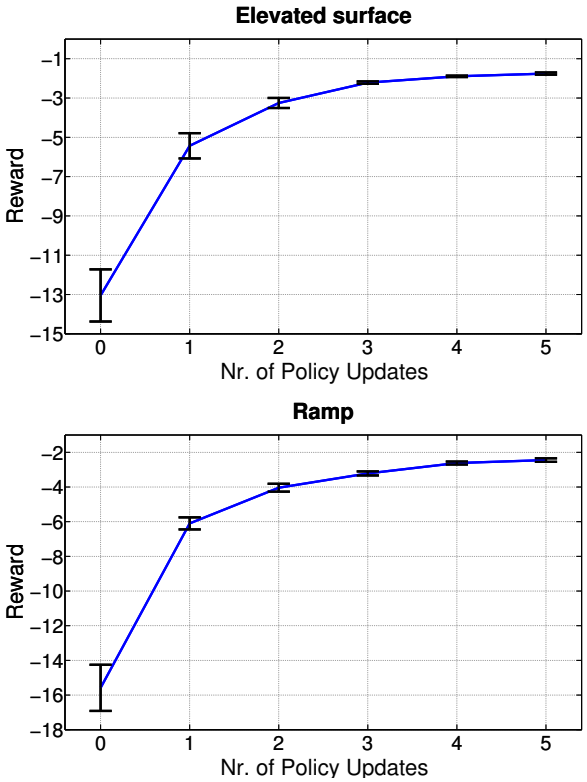

Figure 4.4: Mean reward values and standard errors after each policy update. *Top:* scraping task on an elevated surface. *Bottom:* scraping task on a ramp.

values. Although we have some high reward episodes in the first iteration, they are not consistent. As the mean reward increases, the variance of rewards and therefore the standard error becomes smaller. This indicates, that the exploration rate decreases with each policy update and the policy converges to the tactile feedback weights with high rewards.

In the elevated surface task, the biggest weights were learned for the first phase of the task. By hitting the table sooner than expected, there was a big deviation from the desired trajectory in the phase when the robot should still have been in the air, which led to a need of a large correction of the movement in order to maximize tactile trajectory similarity. In the ramp task, substantial weights were learned for the second action phase as the deviation from the desired signal gradually increased during scraping along the surface of the ramp. Therefore, the robot had to adjust its height to stay close to the trajectory.

In both tasks, the robot could achieve a significant improvement of the task performance and adapt to the changed environment by learning the tactile feedback weights.

5 Conclusion

This section concludes the work presented in this thesis by providing a summary of the main results and pointing out the directions for future work.

5.1 Summary

This work presented methods for using tactile sensing for in-hand object localization, tactile time series analysis and object manipulation with tactile feedback. In the following, we describe the main results of this thesis.

Three pose estimation algorithms from computer vision were adapted for tactile images acquired from tactile matrix arrays: probabilistic hierarchical object representation, iterative closest point and voting scheme. We showed that the number of tactile observations and in general the resolution of the tactile data have a big influence on their performance. The probabilistic voting scheme showed the best results for performing localization from low-resolution tactile images acquired from single grasps of objects. We observed that the localization performance could be improved by increasing the number of tactile observations, e.g. with tactile images from multiple grasps.

Imitation learning and dynamic motor primitives were employed for learning a manipulation action from human demonstration by kinesthetic teach-in and encoding desired tactile trajectories. Tactile feedback was incorporated into the robot control by computing deviations from the desired tactile trajectory and multiplying them by feedback weights. The dimensionality of tactile data was reduced by performing principal component analysis of tactile images collected during multiple demonstrations of a task. The number of tactile feedback weights was reduced by learning single weights for each action phase. Action phases were recognized by applying spectral clustering to a similarity matrix based on the kernel similarity of tactile images. Furthermore, we showed how tactile time series can be compared with dynamic time warping and kernel similarity. The parameters of tactile feedback were optimized with the episodic version of the relative entropy policy search reinforcement learning algorithm.

Robot hand equipped with low-cost tactile matrix arrays was built and used for collecting tactile data and providing tactile feedback. In-hand pose estimation was evaluated by performing leave-one-out and leave-ten-out cross-validation of tactile images of five different objects. Similarity measure for tactile time series was used to compute similarities of two actions performed at different positions, orientations and velocities. Learning of manipulation skills with tactile sensing was evaluated on a real robot by performing a gentle scraping task with a spatula in an altered task setup. Substantial improvement of the task performance could be achieved by learning the tactile feedback weights that made the robot stay close to the desired tactile trajectory.

5.2 Future work

In future work, presented methods should be tested on more complex tasks with a larger number of action phases. In order to be able to perform manipulation in complex cluttered environments and receive additional cues for object localization, the problem of combining tactile data and visual information should be addressed in the future. Employing of higher-resolution sensors can provide new kinds of local tactile features with greater details of the object form and surface. However, it will further increase the amount of tactile information and need for the dimensionality reduction. In this work, we attached tactile sensors to a parallel gripper. Although our methods can be used for other kinds of grippers, e.g. grippers with fingers, the application of them will require adaptation to the new arrangement of tactile elements.

Classification of tactile time series should be further studied in order to detect action phases. This can be useful for combining and merging of movements based on detected phases, e.g. by switching and blending between various DMPs. Dynamic time warping similarity measure can be used with different classification algorithms that are capable to incorporate similarity kernels.

Self-collision detection, as mentioned in Section 4.2.2, remains an important problem. Ability to differentiate between contact with an object and contact with another parts of the hand is especially crucial for compliant grippers. This skill can help to remove noise and provide information, which parts of the tactile image can be discarded or treated differently during localization and manipulation phases. Possible ways to detect self-collisions include building models of typical tactile signals during self-collision and employing classification algorithms for detecting them. Furthermore, additional sensors that can recognize temperature or material properties can provide cues about the self-contacts similar to the human skin.

Bibliography

[Bal81] Dana H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[BM92] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.

[BRF10] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Kernel descriptors for visual recognition. In *NIPS*, pages 244–252. Curran Associates, Inc., 2010.

[Che95] Yizong Cheng. Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(8):790–799, 1995.

[CM92] Yang Chen and Gérard G. Medioni. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, 1992.

[CSPB11] Sachin Chitta, Jürgen Sturm, Matthew Piccoli, and Wolfram Burgard. Tactile sensing for mobile manipulation. *IEEE Transactions on Robotics*, 27(3):558–568, 2011.

[DH72] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, 1972.

[DNP13] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

[DPP09] Renaud Detry, Nicolas Pugeault, and Justus Piater. A probabilistic framework for 3D visual object representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(10):1790–1803, 2009.

[EPDP13] Peter Englert, Alexandros Paraschos, Marc Peter Deisenroth, and Jan Peters. Probabilistic model-based imitation learning. *Adaptive Behaviour*, 21(5):388–403, 2013.

[FGMR10] Pedro F. Felzenszwalb, Ross B. Girshick, David A. McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, 2010.

[For65] E. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–780, 1965.

[GGA+11] Daniel Glasner, Meirav Galun, Sharon Alpert, Ronen Basri, and Gregory Shakhnarovich. Viewpoint-aware object detection and pose estimation. In *ICCV*, pages 1275–1282. IEEE, 2011.

[INS02] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *ICRA*, pages 1398–1403. IEEE, 2002.

[ISFW03] Alexander T. Ihler, Erik B. Sudderth, William T. Freeman, and Alan S. Willsky. Efficient multiscale sampling from products of gaussian mixtures. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *NIPS*. MIT Press, 2003.

[JF09] Roland S. Johansson and Randall J. Flanagan. Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature Reviews Neuroscience*, 10(5):345–359, April 2009.

[Jol86] I. T. Jolliffe. *Principal component analysis*. Springer, New York, 1986.

[KDPP10] Oliver Kroemer, Renaud Detry, Justus H. Piater, and Jan Peters. Combining active learning and reactive control for robot grasping. *Robotics and Autonomous Systems*, 58(9):1105–1116, 2010.

[KK03] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371, 2003.

[KL99] Joseph B. Kruskal and Mark Liberman. The symmetric time-warping problem: from continuous to discrete. In David Sankoff and Joseph B. Kruskal, editors, *Time Warps, String Edits, and Macromolecules - The Theory and Practice of Sequence Comparison*, chapter 4. CSLI Publications, Stanford, CA 94305, 1999.

[KLP11] Oliver Kroemer, Christoph H. Lampert, and Jan Peters. Learning dynamic tactile sensing with robust vision-based training. *IEEE Transactions on Robotics*, 27(3):545–557, 2011.

[KMP08] J. Kober, B. Mohler, and J. Peters. Learning perceptual coupling for motor primitives. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 834–839, 2008.

[Lux07] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[Mel00] Claudio Melchiorri. Slip detection and control using tactile and force sensors. *Mechatronics, IEEE/ASME Transactions on*, 5(3):235–243, 2000.

[MKP10] Katharina Mülling, Jens Kober, and Jan Peters. Learning table tennis with a mixture of motor primitives. In *Humanoids*, pages 411–416, 2010.

[MRY11] Roozbeh Mottaghi, Ananth Ranganathan, and Alan L. Yuille. A compositional approach to learning part-based models of objects. In *ICCV Workshops*, pages 561–568. IEEE, 2011.

[PKRS12] Peter Pastor, Mrinal Kalakrishnan, Ludovic Righetti, and Stefan Schaal. Towards associative skill memories. In *Humanoids*, pages 309–315. IEEE, 2012.

[PKTN07] Anna Petrovskaya, Oussama Khatib, Sebastian Thrun, and Andrew Y Ng. Touch based perception for object manipulation. *Robotics Science and Systems, Robot Manipulation Workshop*, 2007.

[PMA10] Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI*. AAAI Press, 2010.

[PS06] Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *IROS*, pages 2219–2225. IEEE, 2006.

[Sch09] S. Schaal. The sl simulation and real-time control software package. Technical report, 2009.

[SII+10] Erik B. Sudderth, Alexander T. Ihler, Michael Isard, William T. Freeman, and Alan S. Willsky. Nonparametric belief propagation. *Communications of the ACM*, 53(10):95–103, 2010.

[Sil86] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall/CRC, April 1986.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 2000.

[SSM96] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, 1996.

[SSS+09] Alexander Schneider, Jürgen Sturm, Cyrill Stachniss, Marco Reisert, Hans Burkhardt, and Wolfram Burgard. Object identification with tactile sensors using bag-of-features. In *IROS*, pages 243–248. IEEE, 2009.

[SSSL09] Min Sun, Hao Su, Silvio Savarese, and Fei-Fei Li. A multi-view probabilistic model for 3d object classes. In *CVPR*, pages 1247–1254. IEEE, 2009.

[STFW06] Erik B. Sudderth, Antonio Torralba, William T. Freeman, and Alan S. Willsky. Depth from familiar objects: A hierarchical model for 3d scenes. In *CVPR (2)*, pages 2410–2417. IEEE Computer Society, 2006.

[TKI+04] N. Tsujiuchi, T. Koizumi, A. Ito, H. Oshima, Y. Nojiri, Y. Tsuchiya, and S. Kurogi. Slip detection with distributed-type tactile sensor. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 331–336 vol.1, 2004.

[Zha94] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.

[ZL12] Xiaoyou Zhang and Rongqiang Liu. Slip detection by array-type pressure sensor for a grasp task. In *Mechatronics and Automation (ICMA), 2012 International Conference on*, pages 2198–2202, 2012.