Adaptive robotic tool use under variable grasps

Heiko Hoffmann, Zhichao Chen, Darren Earl, Derek Mitchell, Behnam Salemi, and Jivko Sinapov

HRL Laboratories, LLC, Malibu, CA

Abstract

Successful robotic manipulation of human tools will greatly advance robotic collaboration with humans in manufacturing and robotic assistance in human environments, e.g., in hospitals, offices, and homes. In these settings, the robot needs to grasp a tool (e.g., a drill) before using it, rather than rely on the tool being firmly attached to the robot's end effector. Thus, when using the tool, the robot has to account for the uncertainty in the hand-tool interface, since the grasp will vary between trials. To address this challenge, we propose a new framework in which control-relevant parameters are extracted about the uncertain interface between palm and tool tip. Our approach allows a robot to control position and force at the tool tip using either visual or tactile feedback. In addition, the proposed framework allows a robot to move the tip of a tool along a surface, despite uncertainty about how the tool is held in the hand and uncertainty about the structure of the surface. We demonstrated the feasibility of our new approach on two robotic platforms: the DARPA ARM robot operating a hand-held drill and an ST Robotics R17 robot drawing with a pencil.

Keywords: Grasping, manipulation, computer vision, tactile sense, kinematics, adaptive systems

1. Introduction

Dexterous robotic manipulation of hand-held tools has the potential to revolutionize manufacturing; such an ability will enable better cooperation between humans and robots, make robots more versatile, and replace human hands in dangerous tasks. Robotic manipulation in unstructured environments, however, remains a hard challenge. A key problem in robotic tool

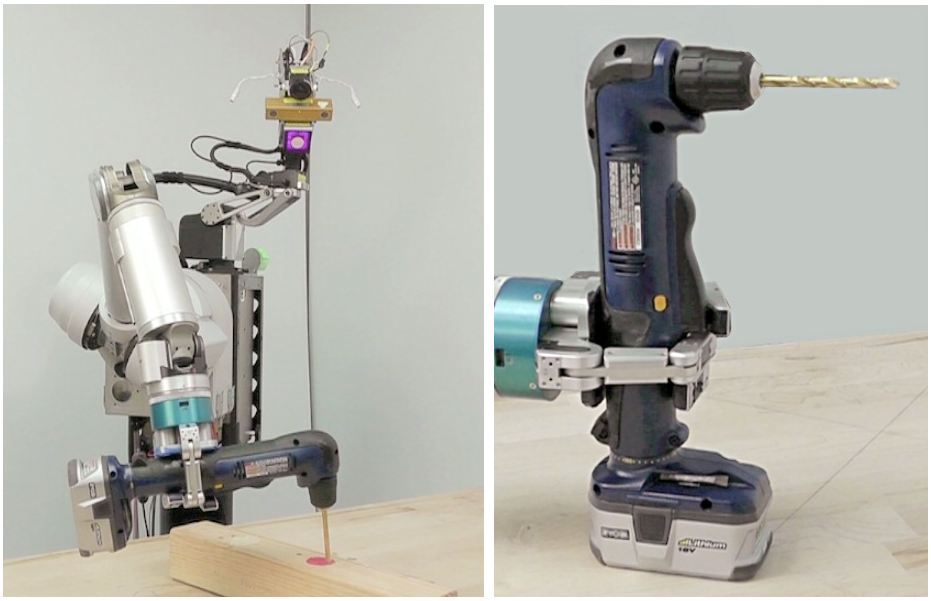

Figure 1: Accurate tool tip control requires adaptation to grasp variability. The red dot on the wooden block is the target for drilling.

use is that a robot's grasp of hand-held tools will vary from trial to trial, introducing an additional source of uncertainty. Therefore, the tool's position and pose relative to the robot will change between trials, which can negatively affect a manipulation task. This variability is particularly present when dealing with tools made for humans (Fig. 1) instead of special tools designed for robots. Moreover, the variability is exacerbated if the robot cannot be calibrated accurately, which will be the case for operations in the field. In addition, uncertainty is present in the interface between tool and environment (e.g., between the tip of a pen and paper) partly because friction is still hard to model and to predict. Thus, an analytical model of a grasp will fail under such uncertainty.

Previous efforts in the area of robotic manipulation under uncertainty focused on using low-gain compliant control [1, 2, 3]. Such control avoids hard collisions, but there is a trade-off between softness and performance (e.g., moving quickly and accurately to a desired end-effector position). To improve performance, some authors have suggested using learning methods to compute control torques without increasing the control gains for trajectory tracking [4, 5]. Through random exploration (i.e., motor babbling), the robot

can learn the kinematic and dynamic relationship between joint angles or torques and hand position. However, these efforts were limited to learning the kinematics and dynamics of the robot arm itself and thus could not cope with an uncertain interface between the robot's hand and a manipulated object.

Learning the dynamics of a robot arm requires a non-linear mapping in a high-dimensional space. Independent of the learning algorithm, the main problem is that the whole input space has to be sampled sufficiently dense to obtain the required training data. For example, for a 7 DOF robot arm, learning the mapping from joint states onto torques requires a total of $3^{14} = 4.8$ million data points even in the optimistic case of sampling only three data points per dimension. These data points would have to be collected during exploration. Assuming that we need at least one second for each data point (which is again optimistic), the total data-collection time would be 55 days of non-stop operation. Thus, obtaining a sufficient coverage of the input space is infeasible. As a result, approaches that learn the full inverse dynamics are typically restricted to controlling a robot around a given trajectory [6, 7].

Thus, we omit *learning* the arm kinematics/dynamics for controlling the hand, which usually can be controlled accurately – see, e.g., industry robots. Instead, our new framework focuses directly on the component of highest uncertainty, which is the interface between tool and hand. To achieve autonomous tool use, our framework uses visual and tactile sensory feedback. We use vision to estimate the linear transformation between tool tip and hand. Given this transformation, the tool can be controlled in task space, as other authors have suggested [8, 9], and the robot can bring the tool in contact with a surface. Next, the robot probes the surface and records tactile feedback. To enable a controlled interaction, we learn the transformation between tactile state and robot-hand motion.

We tested the key components of our framework in two robot experiments. First, on the DARPA ARM robot, we demonstrated autonomously picking up a hand drill and drilling into a wooden block. This experiment shows that the Jacobian matrix can be augmented based on visual feedback of the drill bit location to allow successful placement of the drill bit despite uncertainty about the grasp angle (± 13 degrees). Here, we used a novel vision process that combines both 2D contour and 3D spatial information to robustly locate a tool tip. Second, with the R17 robot, we demonstrated probing a surface with a tool, extracting a tactile state, and learning a mapping between this

state and the end-effector position. We used this mapping to successfully draw with a pencil on a surface despite the unknown slope of the surface and uncertainty about the orientation of the pencil within the robot gripper.

The remainder of this article is organized as follows. Section II introduces our new framework. Section III presents our new vision process for detecting tool tips. Section IV describes the experiments with the DARPA ARM robot and Section V the experiments with the R17 robot. Section VI discusses further related work in vision and tactile processing, and mentions limitations and extensions of our methods. Finally, Section VII concludes the article.

2. Framework for Robotic Tool Use

In our framework for using a tool in an uncertain grasp, we want to address a class of applications that involve moving a tool towards a surface and making the tool interact with the surface. Examples of such applications include moving a screwdriver towards a screw, inserting a key, and sanding a surface (Figure 2).

If the grasp of the tool is firm, we can operate with the tool tip in the coordinate frame of the task. We assume that the location and orientation of the coordinate frame for the palm, H (for hand), is known, i.e., the robot has known kinematics. To operate in task space, we need to find the linear transformation T that maps a coordinate in the frame of the tool tip, O (for object), onto H (as also pointed out in [9]). This transformation is obtained from vision, and since the transformation is linear, a single observation with a stereo camera or 3D sensor is sufficient (multiple observations can increase accuracy). Thus, in our proposed framework, the robot moves the tool such that the tip is visible in the robot's camera(s), processes the camera images, and then proceeds applying the tool. To use visual servoing to align the tip with a target, we augment the robot's Jacobian matrix with the transformation T, as shown below.

Once the tool is in contact with a surface, we switch from visual servoing to controlling the tool such that it "feels right" in the hand. Humans appear to do that. For example, when moving a key along a surface in search for the keyhole, we expect a certain tactile feedback to keep the key in slight contact with the surface (Figure 2). Likewise, when drawing with a pencil or sanding a surface, a certain tactile feedback of the object in the hand is expected. In our framework, the robot initially probes the surface with the tool. Thus, the robot explores the relationship between moving towards the surface and

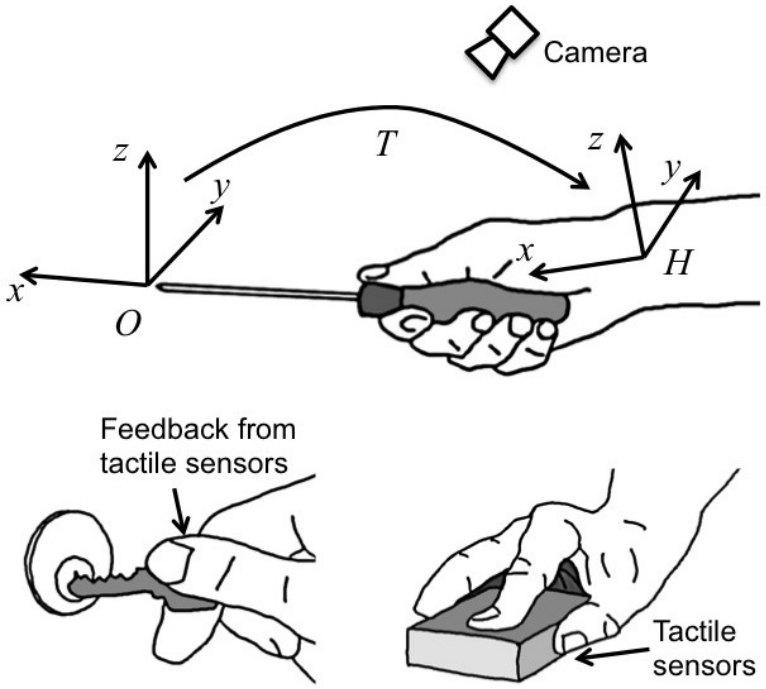

Figure 2: In our framework, we overcome the uncertainty of a grasp by a) obtaining a linear transformation between tool tip frame O and hand/end-effector frame H from vision and b) when bringing the tool in contact with a surface, using tactile feedback and learning the mapping between hand control and tactile state.

the corresponding tactile response. The resulting sensory data are reduced to a tactile state, and we learn the relationship between this state and the robot motion (see below for more details). Simultaneously, the robot figures out the tactile state that gives a desired result, e.g., keep a slight contact, or sand away a small amount of material. This observation then gives a desired tactile state, which together with our learned relationship allows close-loop control of the hand.

Figure 3 gives an overview of the architecture to realize our framework. The robot controller takes visual information of the transformation between tool tip and robot end-effector to move the tip along a desired path. In the closed-loop control, this transformation is fixed, and feedback about the tool tip position is used to compute the control error. When moving the tool along a surface, the controller switches mode to using tactile feedback. Also, here,

the initially learned transformation stays fixed in the closed-loop control. In summary, our framework has the following four operational steps:

1. Estimate the transformation from the palm to the tool tip,
2. Use that transformation to move the tool's tip to a surface,
3. Perform motor-babbling to learn the relationship between tactile state and actions, and
4. Use that learned relationship to control the tip along the surface.

The following two sub-sections provide more details on augmenting the robot Jacobian with the transformation matrix T and on extracting a tactile state and learning a relationship between this state and the robot's action.

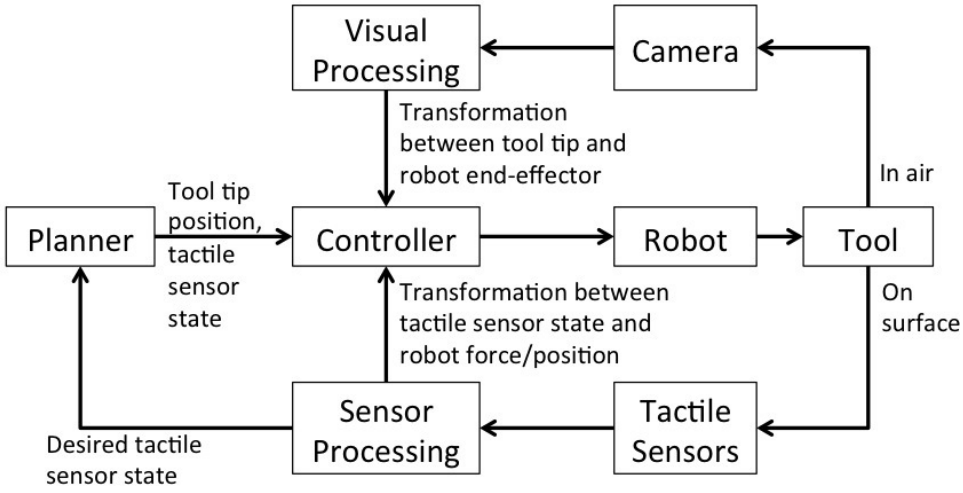

Figure 3: Visual and tactile feedback complement each other in our architecture for bringing a tool to a surface and interacting with the surface.

2.1. Tool-Tip Control using Visual feedback

After grasping a tool, the robot extracts the position and orientation of the tool tip from the images captured by the robot's cameras (see Section 3 for an algorithm that accomplishes that for an *elongated* tip). Given this information, we compute the corresponding translation vector $\mathbf{a} := (a_x, a_y, a_z)$ and rotation matrix $\mathbf{R}$ in the coordinate frame H. As it is common use for

robot kinematics [10], we use a 4×4 matrix to combine $\mathbf{a}$ and $\mathbf{R}$ into a single linear transformation,

$$\mathbf{T} = \begin{pmatrix} R_{xx} & R_{xy} & R_{xz} & a_x \\ R_{yx} & R_{yy} & R_{yz} & a_y \\ R_{zx} & R_{zy} & R_{zz} & a_z \\ 0 & 0 & 0 & 1 \end{pmatrix} . \tag{1}$$

This matrix converts a position in coordinate frame O into frame H (the fourth dimension of the position is set equal to 1). Let $\mathbf{D}^P(\boldsymbol{\theta})$ be the kinematic transformation between robot base and palm, as a function of the joint angles $\boldsymbol{\theta}$. To get the full kinematics $\mathbf{D}(\boldsymbol{\theta})$ to the tool tip, we simply multiply by $\mathbf{T}$,

$$\mathbf{D}(\boldsymbol{\theta}) = \mathbf{D}^P(\boldsymbol{\theta})\mathbf{T} . \tag{2}$$

For visual servoing, in our experiment with the DARPA ARM robot, we controlled just the position of the tool tip and not its orientation. In that case, the robot's Jacobian matrix J_{ik} for the tool tip position reduces to

$$J_{ik} = \sum_{j=1}^{4} \frac{\partial D_{ij}(\boldsymbol{\theta})}{\partial \theta_k} \delta_{j4} = \sum_j \frac{\partial D^P_{ij}(\boldsymbol{\theta})}{\partial \theta_k} b_j , \tag{3}$$

where b_j are the elements of the vector $\mathbf{b}$,

$$\mathbf{b} = \mathbf{T} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} a_x \\ a_y \\ a_z \\ 1 \end{pmatrix} . \tag{4}$$

The derivatives of $D^P_{ij}(\boldsymbol{\theta})$ can be pre-computed given the known robot kinematics, and only $\mathbf{a}$ has to be estimated.

The resulting Jacobian provides a functional relationship between changes in joint angles and changes in the tool tip position. Thus, the robot can directly control the tip position, $\mathbf{p}$, and exert a force, $\mathbf{f}$, along the bit axis,

$$\dot{\mathbf{p}} = \mathbf{J}\dot{\boldsymbol{\theta}} \tag{5}$$

$$\boldsymbol{\tau} = \mathbf{J}^T \mathbf{f} . \tag{6}$$

In our experiment, we controlled the position via impedance control, where $\mathbf{f}$ was set proportional to the position error, and $\boldsymbol{\tau}$ was added on top of the gravity-compensation torques.

2.2. Learning Tool-Tip Control using Tactile feedback

When the tool is in contact with a surface, the robot first probes the surface and then uses the resulting learned relationship between robot movement and tactile sense to control the hand. As shown in Figure 4, in our framework, there are two distinct phases: exploration and execution.

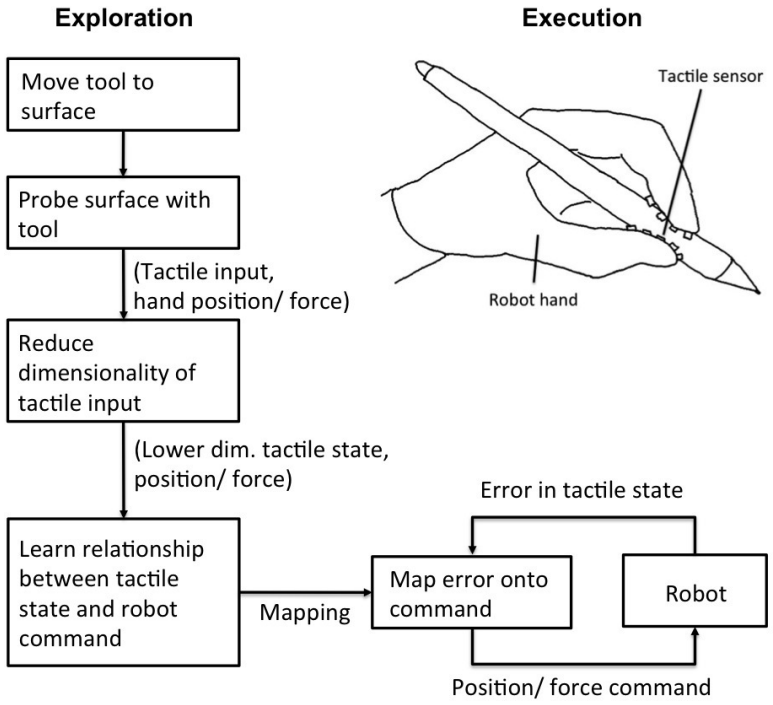

Figure 4: Learning tool-tip control using tactile feedback. This diagram shows the process flow from moving the hand for exploration to learning a sensorimotor relationship. This relationship is then used in closed loop (bottom right) to control the robot hand.

During exploration, the robot makes small movements with the tool against the surface and records the resulting data from the hand's tactile sensors. These data form a point distribution in a high-dimensional space – one dimension per tactile sensor. In this space, we want to find an intrinsically lower dimensional representation of the data point distribution (Fig. 5). For example, if the distribution extends linearly, principal component analysis [11] can be used to find the directions of maximum variance. For non-linear distributions, as illustrated in Fig. 5, other methods are available,

e.g., kernel principal component analysis [12, 13]. In machine learning, this process is referred to as unsupervised learning.

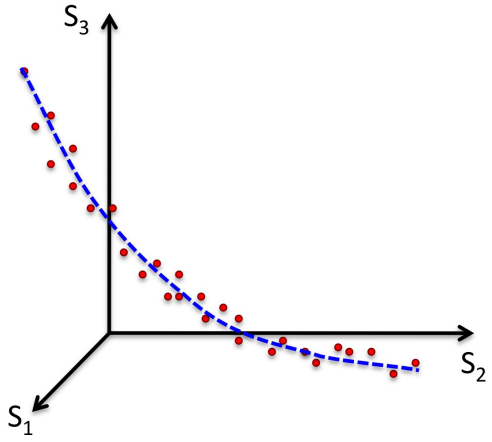

Figure 5: Distribution of sensory input during exploration (illustration - not real data). The sensory values (s_i) vary systematically depending on motor command; i.e., they are expected to lie on a lower-dimensional manifold (dashed curve). Here, only three sensory dimensions are shown for illustration.

Once the lower dimensional representation or manifold is extracted, the data are projected onto this manifold. For example, in the case of principal component analysis the data are projected onto the principal components. We call the resulting variable of this projection the tactile state.

An advantage of projecting the sensor values onto a lower dimensional state is that it eliminates a large part of the sensor noise. Background noise is very common for low cost tactile sensors. Particularly, when using a tactile array, some tactile sensors will not even be in contact with the tool and thus contribute only noise. However, the noise variance is lower than the variance of the sensors contributing useful data, and thus, the latter will dominate in the dimensionality reduction. The noise dimensions will collapse in the projection onto the lower-dimensional manifold.

After the tactile states are extracted, we need to link them with the corresponding exploratory actions of the robot (Fig. 6). In our framework, this linkage is achieved using supervised learning or regression, which maps a tactile state onto a robot's action. For example, for a linear mapping, several linear regression methods are available [14, 15], and non-linear mappings are also well understood [16]. The relationship between tactile state and action

could even be learned if the mapping is not one-to-one, as discussed in detail in [13].

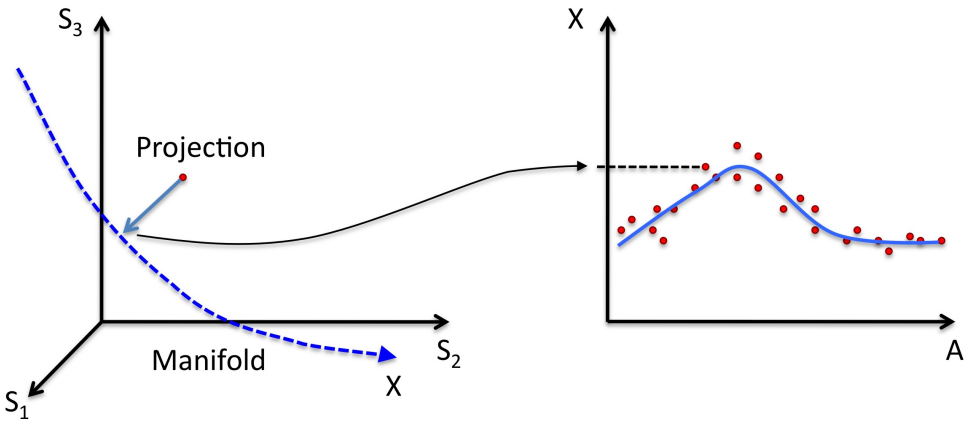

Figure 6: Learning the sensorimotor relationship (illustration - not real data). (Left) Each sensory data point is projected onto the manifold extracted from learning a low-dimensional representation of the sensory input. This manifold is parameterized, X. (Right) The obtained value from the projection is plotted against the corresponding robot action, A. Here, another learning method learns the relationship (solid blue curve) between A and X.

Once we have the learned relationship between tactile state and robot action, we can use this relationship for closed-loop control, where the objective is to maintain a certain desired tactile state (Section 5.1 presents a specific example). The desired tactile state could be, e.g., set to the observed value during initial contact. The unique aspect of the above process is the combination of exploring the sensory effect of a motor command, learning a lower-dimensional representation of this sensory effect, and learning the link between the sensory representation and the corresponding motor command.

3. Visual Processing for Localizing Tool Tips

This section presents our method for recognizing and localizing an elongated tool tip. Our method uses 3D depth information to segment the tool from background and a level-set based contour segmentation [17, 18] to preserve the accurate boundaries of the tool tip (Fig. 7).

Our procedure carries out the following steps: first, a depth map is computed using a pair of stereo images or equivalent 3D visual information.

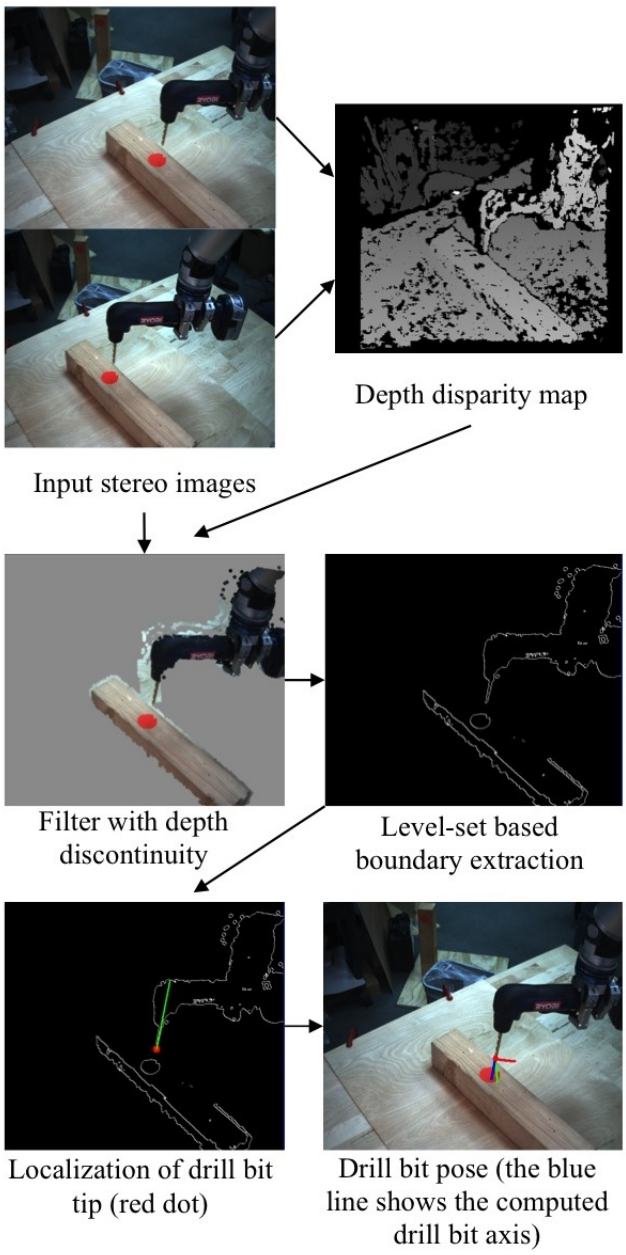

Figure 7: Image processing steps for localizing a tool tip, here, drill bit. The target for drilling is the red dot on the wooden block.

Next, the resulting scene is segmented such that image regions of continuously varying depth are grouped together; that is, the discontinuity in depth defines the segmentation boundaries. Because of noise, such segmentation may miss parts of the tool that are necessary for estimating the tool tip's position and orientation. Therefore, we expand the boundary to insure that the image of the tool is fully contained. Since this boundary is too wide for accurate tool-tip detection, we shrink it again in the 2D visual scene using a level set algorithm [18] (Appendix A).

After the boundary is extracted, the position and orientation of the tool tip are computed as follows. First, the tool tip is selected as the point on the boundary that is closest (minimum Euclidean distance) to the target in the scene. Since we use visual servoing to move the tip to a target, a target has to be in the scene anyway. Before we use our tool tip detector, in preparation, the robot moves the tool into the proximity of the target, for which the palm position is sufficient (the orientation of the tool has to be approximately known, at least within $\pm 90^{\circ}$).

To find the orientation of the tool tip, a set of candidates is generated by drawing lines through the tip point from every point on the boundary (Fig. 8). From that set of candidates, we choose the longest line that is entirely within the boundary (the level set algorithm provides the required data - see Appendix A).

The resulting line is a projection of the desired tool tip axis. To compute its 3D orientation, we have two options: 1) compute this line for both stereo images or 2) find another specific point on the line (see Section 4.1), which can be projected into 3D using the disparity map. Thus, together with the tool tip we have two points, which define the 3D orientation of the axis. The second option saves processing time since discontinuity filter and level-set algorithm have to be computed only once. The first option, however, generalizes better to other tools.

4. Experiment with the DARPA ARM robot

Using the DARPA ARM robot, we tested the visual feedback component of our framework: using vision to augment the robot Jacobian to include the tool tip if the grasp is uncertain. The successful demonstration included autonomous grasping of a drill, moving the drill to a target, and drilling at a pre-defined location. The robot was equipped (in 2011) with one Barrett

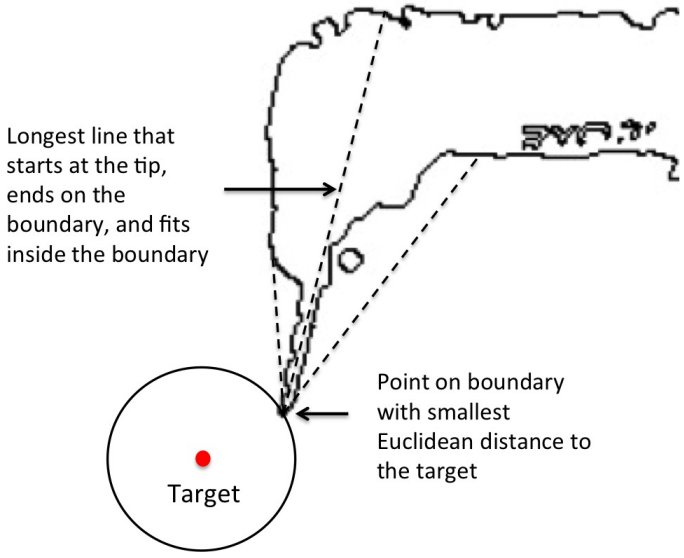

Figure 8: Given the boundary of a tool, we can find the tool tip (closest point on boundary to the target) and its axis (longest line within the boundary, originating at the tool tip).

WAM 7-DOF arm, a Barrett Hand BH8-280 with tactile sensors, a Bumblebee2 (PointGrey) stereo camera (2x 640x480 pixels) on a 4-DOF neck unit, and two microphones (Audio-Technica U853AW).

4.1. Tool tip detection

First, we tested our tool tip detection method. We captured with the robot's stereo camera 170 image pairs (static images), in which the robot held a drill that was pointed towards a drilling target (red dot on wooden block). Figures 9 and 10 show example images that were used for testing our method. The images contained background clutter, which was included on purpose to test for robustness against distracting objects.

We implemented the method described in Section 3 with the option of computing the drill bit axis in only one image. Computing the axis in both images would have increased the computation time by 66% (the most computationally expensive components were the depth disparity map, the discontinuity filter, and the level-set algorithm, each contributing about 1/3 to the computation time). The total computation time for our method was about 1 sec on a 3.3 GHz Intel Xeon processor using a serial implementation in C++.

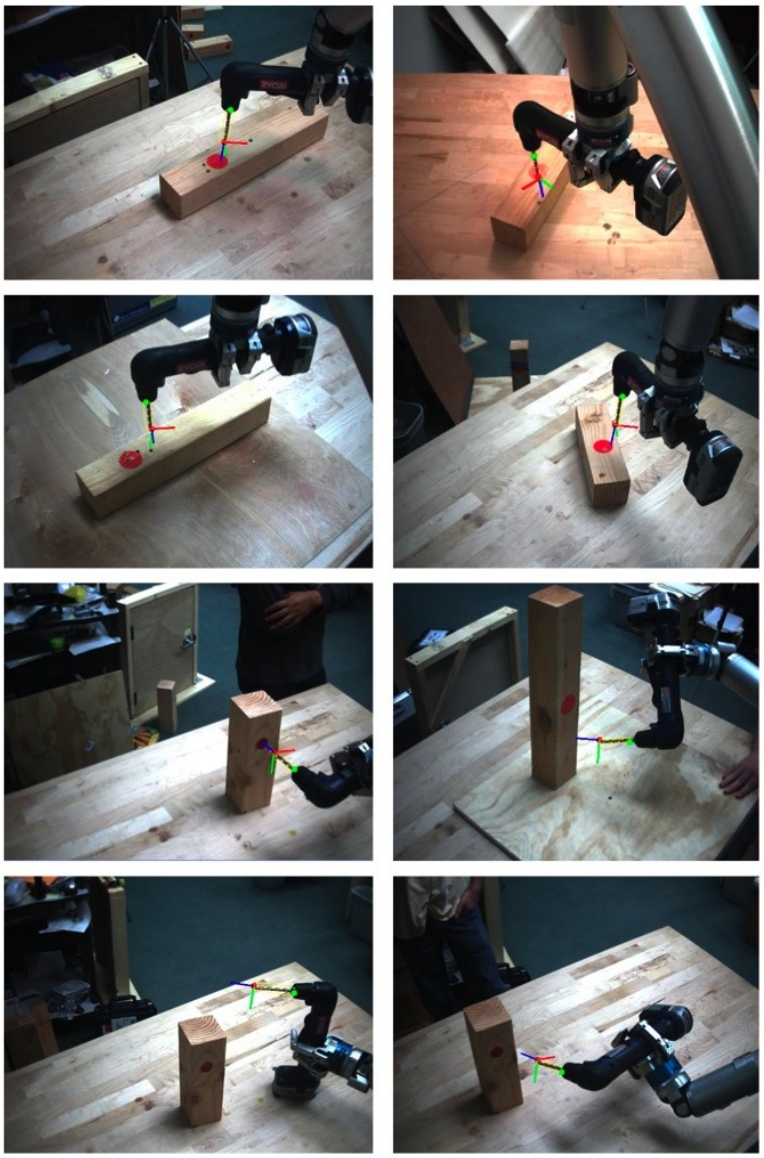

Figure 9: Examples in which our algorithm succeeded to detect and localize the tool tip. The figure shows the raw images from the Bumblebee2 camera. The lighting varied during these trials. The images show overlaid the computed bit tip (red dot) and computed coordinate axes of the drill bit (blue line along the bit axis).

As described in Section 3, we require a second point on the drill bit. For this point, we chose the location where the bit protruded out of the drill. To estimate this location, we fitted a rectangle between location estimate and bit tip. This rectangle was fitted to tightly enclose the bit boundary that we obtained from the level set algorithm. We chose the location that maximized the aspect ratio of this rectangle. Trials with an aspect ratio below a threshold (default: 2) were dismissed as failures, because typically in these trials we identified the wrong location as drill tip.

As a result, out of all trials, in 122 trials (72%), our method successfully detected the tool tip's position and orientation (Fig. 9 shows examples). The average tip position error was 0.5 ± 0.3 cm (n = 8, mean $\pm$ standard deviation, representative sample of successful trials). For these results, since we did not have ground truth data, we relied on human inspection. In 40 trials (24%), our software could not find the tool tip (based on the above threshold). Finally, the remaining 8 trials included 3 trials (2%) in which the estimated tip position was wrong, and 5 trials (3%) in which the estimated tip orientation was wrong (Fig. 10 shows examples). These errors were detected automatically based on two sanity checks: (i) the distance in 3D between the extracted two points on the drill bit had to be within 2/3 and 4/3 of the bit length and (ii) the re-projection of the 3D points onto the 2D stereo images had to match the extracted axes, as in Fig. 8 (not trivial, since we first map from 2x2D to 3D and then from 3D to 2x2D).

In these results, we did not observe any false positive, i.e., a trial in which our method detected a region that was not a drill bit. To improve the method's success rate of 72%, we could relax the aspect ratio threshold below 2, but at the cost of introducing false positives. Figure 11 shows the result of an ROC analysis varying the threshold from 2 to 1.5. The curve saturates at a success rate of 87%. However, having no false positives was more important than a higher success rate, because if the robot could not detect the tip, the robot could still move it to a slightly different location and try again.

4.2. Autonomous drilling

Our robot experiment demonstrated autonomously grasping a drill and drilling at a target (Fig. 12). The target area was a red disk with a 5 cm diameter. The robot's grasping software was designed so that the robot could lift the drill without actuating it, while at the same time, still being

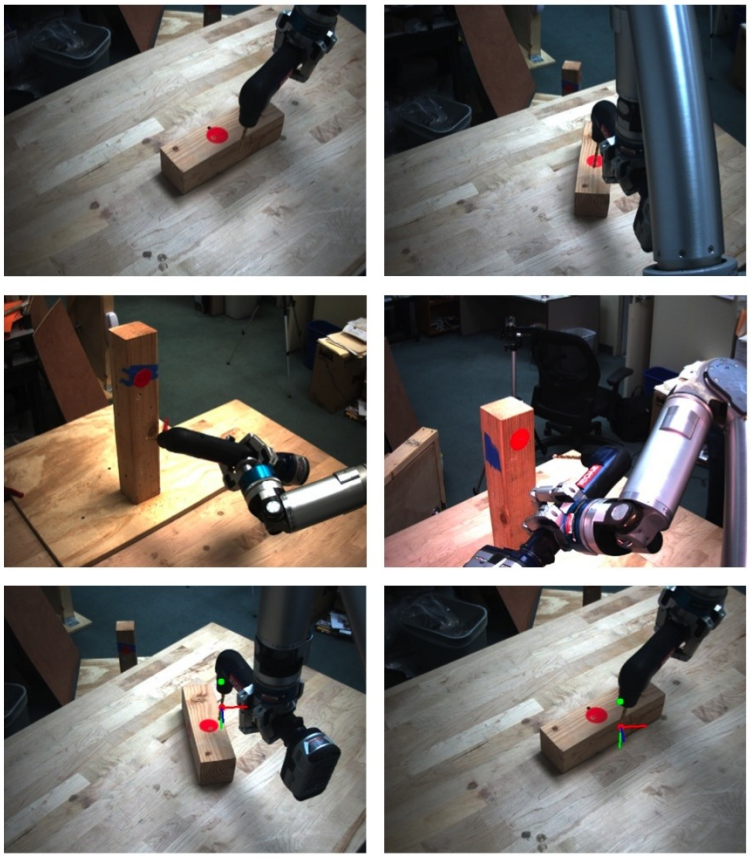

Figure 10: The top four images show examples in which our algorithm failed to detect the tool tip. The bottom left image is an example of a wrong tool tip estimate (red dot). The bottom right image is an example of a wrong estimate of the orientation of the drill bit (blue line, about 30° off). The figure shows the raw images from the Bumblebee2 camera. The lighting varied during these trials.

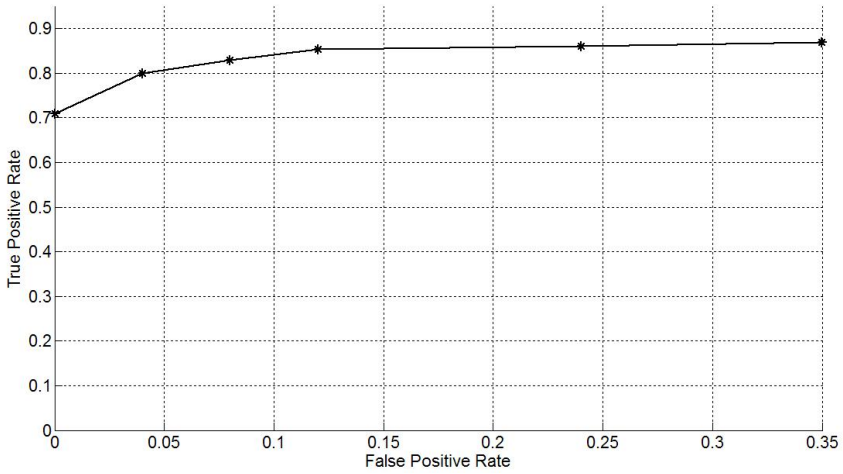

Figure 11: ROC curve for successful tool tip localization. To compute the curve, we varied the threshold for tool tip recognition.

able to actuate the drill when desired[1]. Here, we focus the description of the technical details to elements relevant to the methods introduced in this article.

The drill's initial position and orientation were obtained by fitting the drill's 3D model to point-cloud data obtained through the stereo camera. The fit was computed using the Sample Consensus-Initial Alignment method developed by Rusu *et al.* [19] as implemented in the Point Cloud Library [20]. Next, the drill's pose estimate was used to compute the target position and orientation of the robot hand, which was *a priori* defined relative to the drill. After executing the robotic motion, the position and orientation of the hand relative of the drill varied between trials. Responsible for this variability were mainly three error sources: vision calibration, visual pose estimation, and robot joint-angle measurements. To press the drill button despite this variability, we preformed a sequence of three steps: 1) exploit hand compliance to pull the hand towards the drill when grasping, 2) glide fingers along the drill handle and exploit hand compliance to stop the fingers

[1]Other teams on the DARPA ARM program chose a power grasp that was easier, but kept the drill bit spinning, while the drill was moved towards the target - a less elegant and more dangerous choice.

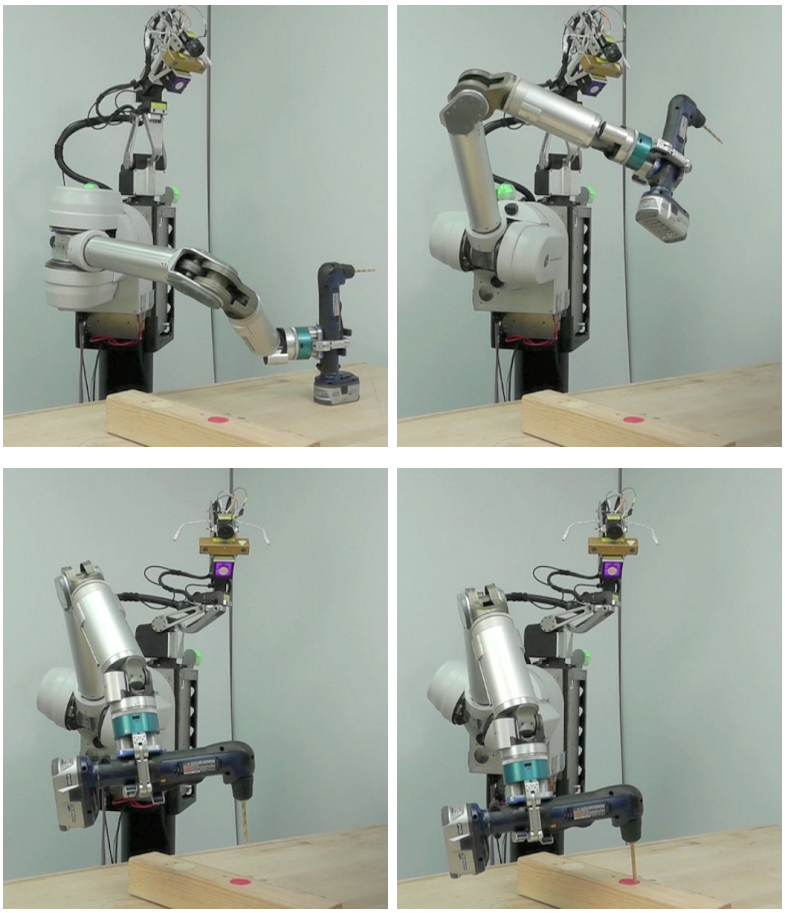

Figure 12: Autonomous drilling. The four images are taken from a video showing the whole manipulation sequence. Initially, the robot autonomously grasped the drill; then, the robot moved the drill towards a target (red dot on wooden block) using visual feedback. Finally, the robot drilled into the wooden block using the augmented Jacobian from joints to drill bit position.

at the power button (see video in Supplemental Material), and 3) use auditory feedback to verify that the drill can be actuated with one of the robot's fingers. If the test failed, the robot would release the object and re-adjust its grasp (by rotating the robot's hand around the axis of the handle by 13°) and try again.

On HRL's ARM robot, the above procedure had close to a 100% success rate for grasping and activating the drill (more than 10 trials, the exact number was not recorded). Due to the rotation around the handle, the final grasp posture was uncertain. Thus, a pre-computed posture could result in an estimate of the drill tip position that was 8 cm off (assuming a 26° rotation and radial distance between center of drill handle and bit tip of 17.5 cm). Therefore, it was indeed necessary to correct for the variable grasp.

After grasping, the robot moved the drill close to the drill target. This motion was generated by generalizing from a single demonstrated movement to the current target, as described in [21]. We used this method for generating a target trajectory in joint angles. To follow this trajectory, the robot was torque controlled using joint-level impedance control with a control loop at 200 Hz. To the impedance control torques, we added torques to compensate for the weights of the robot arm and drill.

Near the target, we used visual servoing to align drill bit with target. The location of the bit tip was detected as described in Section 4.1. To align the bit, we augmented the robot's kinematics to include the drill as described in Section 2.1. The resulting Jacobian was also used to exert a controlled force along the axis of the bit and drill into the wooden block (see video in Supplemental Material). This procedure enabled autonomous grasping of a drill and drilling at a target (Fig. 12). The resulting position error of the drill bit tip was 1.3 ± 0.8 cm ($n = 5$, mean $\pm$ standard deviation) in the target plane. These errors were mainly due to vision inaccuracies (Section 4.1) and the impedance control of the robot arm, which was not sufficiently stiff to allow lower errors. Moreover, the cable drive of the Barrett arm, particularly, the varying tension in the cables, lead occasionally to position errors.

Occasionally, our software failed to complete the drilling task, and the robot did not drill into the block. These failures had mainly three causes: (i) failed drill bit localization, as discussed in Section 4.1, (ii) joint limits that interfered with visual servoing (the DARPA ARM robot had joint limits in the middle of the work space), and (iii) slips of the drill within the grasp (the hand failed sometimes to grasp the drill tight enough). Of these three,

the last two resulted from limits of the specific hardware used in the ARM program and are thus not characteristic of our approach. When these errors did not occur, the success rate of our approach was bounded mainly by the success rate of the vision system.

5. Experiment with the R17 robot

With the R17 robot, we tested another key element of our framework: the feasibility of exploring the relationship between tactile sense and robot motion and using this relationship for control. Here, we demonstrated that the robot could use a pencil in an uncertain grasp and draw on a surface of unknown slope. This section provides more details on applying our framework for tactile feedback to a specific task and describes the actual robot experiment. We used an ST Robotics R17 5-DOF arm equipped with a linear gripper, to which we attached two 2 x 2 arrays of force sensors (Interlink FSR), one array for each gripper finger (Fig. 13). Each sensor measures only the average normal force over the sensor area.

5.1. Tool use with tactile feedback

We start with the tool in the robot's hand and near the surface of interaction. Our goal is to control the tool's position relative to a surface or the force against a surface. In the exploration phase, the robot executes movements against this surface. During this phase, data are collected for each tactile sensor (here, a force sensor). Let s_i^t be the raw measurement of sensor i recorded at time t. Simultaneously to these data, we record the robot's actions (here, end-effector position), a_k^t, where the index k indicates, e.g, the dimension of the end-effector position vector.

In our example, several tactile sensors are present in the robot's hand. Depending on the location of the tool in the hand, some sensors will never be in contact with the tool and thus contribute only noisy input. Moreover, we do not know in advance which sensors will contribute useful data. This uncertainty allows us to demonstrate the robustness gained by using a lower dimensional sensor representation, as described in Section 2.2.

Here, the sensor readings are approximately linear as a function of the force applied with the tool. If a tool is rigid and held rigidly in the hand, independent of the tool geometry, the normal force at a tactile sensor is linearly related to the force at the tool tip, because we have, essentially, a rigid lever system with linear force-torque relationships (see [6] for one

example with a linear relationship between force on an object held in hand and tactile forces). Finally, we get the overall function to be linear because the FSR sensors have a resistance which varies approximately linearly with the normal force. With these sensors, the non-linear component is negligible compared to the background noise.

Thus, we used a linear method, namely, principal component analysis (PCA), to extract our low-dimensional manifold. In PCA, we compute the eigenvectors of the covariance matrix $\mathbf{\Sigma}$ of s_i^t,

$$\Sigma_{ij} = \sum_{t=1}^{n} (s_i^t - \bar{s}_i)(s_j^t - \bar{s}_j) \ , \tag{7}$$

where $\bar{s}_i$ is the mean value of s_i^t over all t; i and j are the indices of the covariance matrix. Our desired manifold (here, a hyperplane) is spanned by the eigenvectors that have the largest eigenvalues of $\mathbf{\Sigma}$, and the hyperplane is anchored at the vector with components $\bar{s}_i$. The number of eigenvectors is chosen based on the dimensionality of the task constraints. If the goal is to exert a specific force in one direction, then we need only one eigenvector.

In the next step, we use this manifold to get a lower-dimensional representation of the sensory input, by projecting s_i onto the manifold. In the linear case, this projection is a linear operator $\mathbf{P}$ that contains the above eigenvectors of $\mathbf{\Sigma}$ in its rows. Thus, the sensory representation x_j is computed as

$$x_j = \sum_i P_{ji}(s_i - \bar{s}_i) \ . \tag{8}$$

Finally, we link this sensory representation with its corresponding action. We achieve this by using all data $\{x_i^t\}$ and $\{a_k^t\}$ to learn a mapping between these data points, as illustrated before in Fig. 6.

Here, we control the robot in Cartesian space; thus, this mapping is linear [6], and we use linear regression for learning [15] (the linear regression and the dimensionality reduction performed by PCA could be also combined, which may benefit the removal of irrelevant noise dimensions [22]). From the regression results, we obtain the Jacobian, $\mathbf{J}$, of the hand-tool system, which is the regression coefficient in

$$x_i - \bar{x}_i = \sum_j J_{ij}(a_j - \bar{a}_j) \ . \tag{9}$$

For control, we use the Jacobian to update the robot commands in a closed loop,

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \mathbf{J}^+(\mathbf{x}^* - \mathbf{x}) \ , \tag{10}$$

where $\mathbf{J}^+$ is the Moore-Penrose pseudoinverse of $\mathbf{J}$, and $\mathbf{x}^*$ is the desired value of the sensory representation $\mathbf{x}$.

5.2. Robot task: drawing with a pencil

In our experiment, the robot's task was to draw with a pencil on a surface of unknown slope (i.e., not known to the robotic system). Each trial started by placing a pencil arbitrarily between the tactile sensors in the robot's gripper. Thus, it was not known a priori which sensors would be in contact with the tool (Fig. 13).

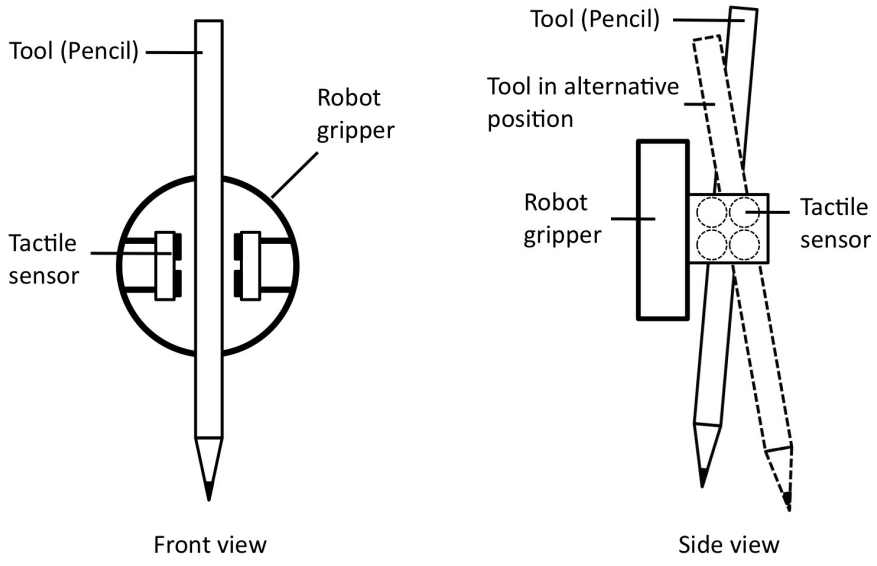

Figure 13: Robot gripper and tool placement. Two 2x2 arrays of tactile sensors were attached to the robot gripper. Here, as tool, we used a pencil, whose position and orientation in the gripper varied between tests (illustrated by dashed pencil contour).

Prior to drawing, the robot explored the relationship between tactile response and hand movements. On a sinusoidal curve, the gripper moved up and down touching the surface with the pen. The robot was controlled by setting the position of the end-effector using the inverse kinematics provided by ST Robotics. In the exploration phase, the gripper moved for three periods,

during which all eight sensor values were recorded at 60 time steps (uniformly distributed in time, at only 2 Hz sampling rate since the robot moved very slowly, few mm/s). On the resulting sensory data, we computed a principal components analysis and extracted the direction of maximal variance (first principal component). Then, we projected all sensory values onto this component (Fig. 14). The resulting relationship between projected sensor values and corresponding height of the gripper is shown in Fig. 15.

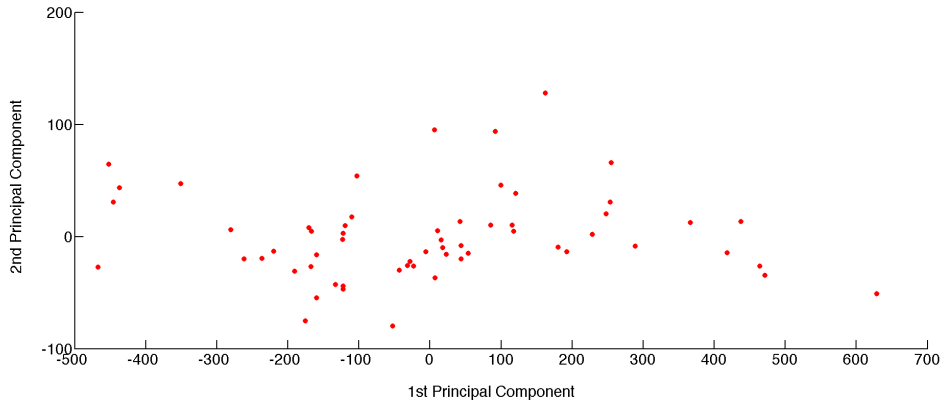

Figure 14: Distribution of sensory input during exploration. In this figure, the data points from the eight tactile sensors are projected onto their first two principal components.

To learn this relationship between gripper height and sensor representation (i.e., sensory input projected onto first principal component), we used ordinary least squares linear regression [15]. Next, we set the desired sensor representation to the mean of observed values during exploration. During drawing, the robot gripper moved uniformly in the horizontal direction, and we controlled the height of the gripper. The robot could draw on a surface with unknown slope based on tactile feedback despite uncertainty of the pencil-sensor interface (Fig. 16 shows a typical sample trajectory, measured through the robot kinematics). We tested the pencil drawing for several trials varying the orientation of the pencil in the gripper and the slope of the drawing surface. The results were consistent across trials. Figure 17 shows video frames from the trial corresponding to the trajectory in Fig. 16. The slope changed direction; nevertheless, the robot could follow the paper surface.

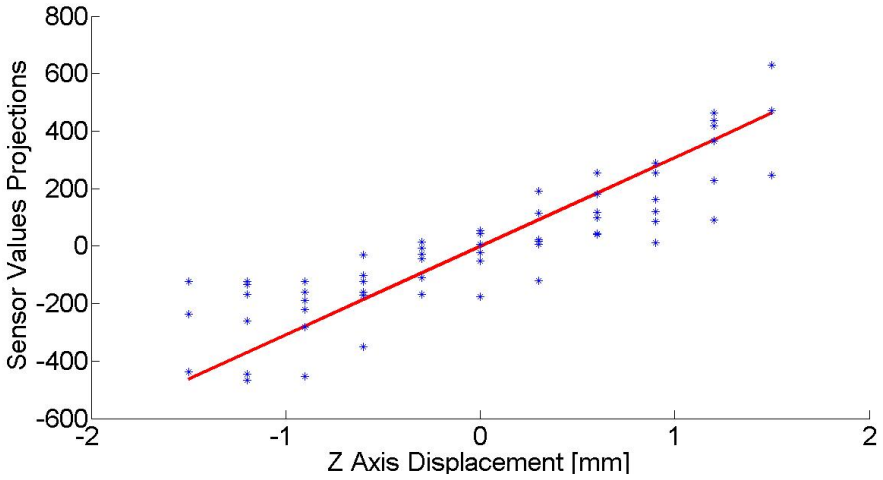

Figure 15: Relationship between hand location (Z-axis displacement) and low-dimensional representation of tactile input (sensor values projected onto first principal component). Recorded values from one exploration trial are shown together with a linear fit to the data (red line).

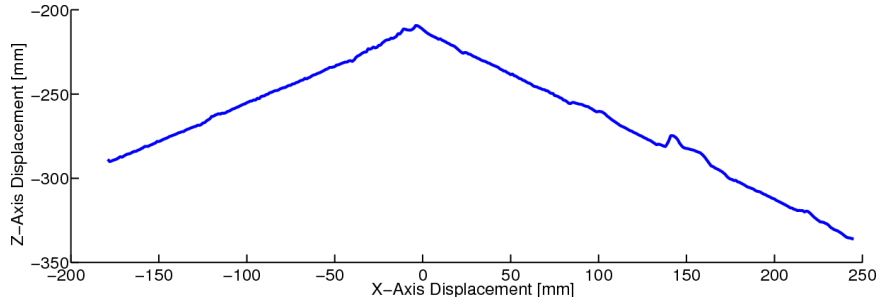

Figure 16: The robot adapted its gripper's vertical position to the slope of the paper surface for drawing. The trajectory was recorded through the robot's internal kinematics (accuracy $<$ 1mm). The trajectory progressed from right to left. The bump in the trajectory reflects a bump in the paper surface (see pictures in Fig. 17).

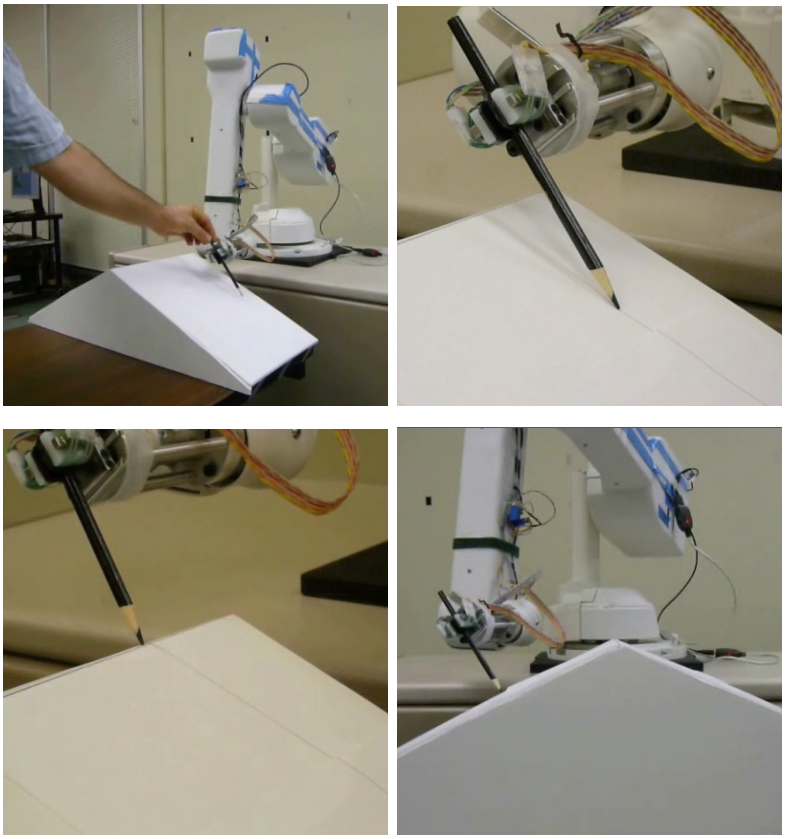

Figure 17: Autonomous drawing with a pencil on a surface of unknown slope. The four images are taken from a video showing the whole drawing sequence. Initially, the pen is placed into the gripper in an arbitrary orientation (the pencil tip has to be able to touch the paper); then, the robot moves the gripper up and down to explore the tactile feedback. Finally, the robot draws on the surface while autonomously adjusting the pencil height.

6. Discussion

This section discusses other prior work for visual tool tip detection and tactile sensing related to our experiments and mentions limitations and extensions of our methods.

6.1. Visual Tool Tip Detection

Template matching is one of the most common methods used to locate the end-effector of a tool grasped by a robot. The image template describes color, texture, and gradient-based edges. The region providing the maximal similarity measure is selected as the location of the object in the image. The disadvantage of this method is that it is sensitive to variations in lighting and background clutter [23].

Other researchers have applied feature-based methods, such as Harris corner features [24], KLT features [25] and SIFT features [26], and significant work has been reported in object detection and recognition [27]. A drawback of those methods is that they work best on objects with a rich surface texture, which is typically not the case for end-effectors such as a drill bit. Another problem of feature-based approaches is that they require the features belonging to the object to be easily separable from the features belonging to the background. While methods that use binocular disparity may be used to separate the object from the background, such methods will not be suitable when the difference in depth between the object and its background is small. Optical-flow-based object segmentation methods, on the other hand, may result in noisy and inconsistent flow patterns, especially if the motion of the object is large. Moreover, they require an oscillation-like movement of the tool [28], which is undesirable for a heavy tool like a drill.

Active contours or so-called snakes are able to segment rigid and semi-rigid objects and are better able to preserve the shape of object [29]. They allow tracking of arbitrary shapes and are relatively robust to occlusions. However, snakes are sensitive to parameters and the initialization of the algorithm. Moreover, snakes have a limited capture range and fail to detect concavities.

6.2. Tactile Sensing for Manipulation

The importance of the sense of touch for object manipulation has been demonstrated by numerous studies with human participants (see [30] for a review). While much work in robotics has focused on developing a variety

of tactile sensors, there is little research on how the tactile sensory modality can be integrated into a robot controller in a principled manner. Instead, most studies on the robotic sense of touch have focused on methods that enable robots to recognize objects and classify them according to different properties [31, 32].

Several studies have also examined how the tactile sense can be used to improve a robot's manipulation skills. For example, a study by Hsiao et al. [33] introduced an information-theoretic motion planning framework that uses tactile feedback to solve grasping and insertion tasks. Other researchers have demonstrated that grasp planning can be integrated with tactile sensing to better detect whether a grasp is stable or not [34].

In contrast to much of the existing work in tactile sensing, the approach proposed in this paper uses motor-babbling coupled with tactile sensing to directly learn an adaptive controller for manipulating a tool. A study by Hoffmann *et al.* [6] has indeed shown that this is possible. However, that study used only simulated sensor readings and therefore may not generalize well to real world data. The method proposed here builds on that previous work by learning a controller which handles tactile data that is both high-dimensional and noisy.

6.3. Limitations and Extensions of our Methods

We presented a novel method to detect tool tips. While the method was demonstrated only for a drill bit, we omitted any drill-specific geometric or color constraints; our only assumption was the presence of an elongated tip (Fig. 8). Thus, this method should also generalize to other tools like, e.g., screwdrivers, pencils, ice picks, and grill lighters. Moreover, our approach for extracting a tool boundary could be used for any tool. As prerequisite for our method, we assumed that a rough estimate of the tool's position and orientation is available such that the tool tip can be identified (e.g., through proximity to a known location). To deal with occlusions, we suggest to control the robot hand to move to a location, where there are no obstacles between robot camera and tool tip. Partial occlusions of the tool are okay (the tool was partially occluded by the hand in our experiments), as long as the tip area is fully exposed.

Our image processing implementation was not optimized for speed. While we used a serial implementation, a parallel implementation (e.g., using GPUs) would greatly improve the computation time. Particularly, the time to compute the disparity map and the depth filter would drop orders of magnitude,

since different rows within an image could be computed separately.

For localizing the tool tip, we observed a success rate of 72%. To improve the reliability of our method, we could make the robot look at the tool from different directions or move the tool with the robot hand to get multiple exposures, while we exploit that we had zero false positives in our tool tip detection method. Thus, using several images would improve both the success rate for localizing the tip and the accuracy of the position and orientation estimates.

The computation of a linear transformation from palm and to tool is related to work in hand-eye coordination, which calibrates a linear transformation between the robot palm and a camera mounted on the palm [35, 36, 37]. Such calibration might provide an alternative to our vision routine for extracting the linear transformation. However, this alternative would require robot arm movements with the tool.

We demonstrated feasibility of our framework for interacting with a surface. We could control a robot to move a tool tip along a surface despite uncertainty about the slope of the surface and variability of the grasp. The same task could be also achieved with the following three methods: 1) impedance control of the hand that uses a force sensor at the wrist, 2) passive compliance in the Z direction, and 3) online estimation of the surface normal [38]. However, all these alternatives have limitations: 1) the force sensor has to be calibrated appropriately, i.e., the force direction relative to the palm motion has to be known a priori, 2) achieving passive compliance in a desired direction requires specifically designed robot mechanics (not present, e.g., in R17), and 3) the method in [38] assumes a known Jacobian between robot base and tool tip - only the radius of a hemispherical tool tip can be unknown. The advantage of our method is that it explores the relationship between sensor values and the corresponding forces between tool and surface; this causal link overcomes the inaccurate or unknown geometry of the sensor's location and allows us to omit sensor calibration. Thus, our method is more general and could complement standard impedance control.

The method described in Section 5.1 is more general than the application to our R17 robot task suggests. The geometric distribution of tactile sensors as well as the shape of hand and tool can all be arbitrary. For any linear force sensor, we can use the method as is, and for sensors with a non-linear force-voltage relationship, the adaptation to non-linear learning methods is possible. Likewise, for non-rigid tools, we will have non-linearities that can be addressed the same way. To use a grasped tool, our framework requires

that the desired tactile state is known to the robot. This tactile state will vary between applications; for example, for sanding, it could be the amount of pressure required to achieve a certain discoloration of the surface that the robot works with. Here, a vision system needs to detect the discoloration and the robot takes note of the coincident value of the tactile state. Once we have that match, visual feedback is not needed anymore.

Hardware and software limitations of the R17 robot resulted in very slow movements; thus, we used a low sampling rate (2 Hz). Generally, however, the control equation (10) can be computed very quickly (low-dimensional matrix-vector multiplication). Thus, the sampling rate is limited only by the basic robot control loop and the physical properties of the tactile sensors.

In both of our experiments, we assumed that the robot is able to firmly grasp the tool. If the tool would wobble in the hand, we cannot apply the same approach for control. However, we can accommodate occasional slips if the robot is able to detect the slip, e.g., through the use of tactile sensors. In our R17 setup, a slip could be detected as a deviation of the sensor values from the first principal component (compare with Fig. 14), because a different tool orientation in the hand would result in a different principal component. When we detect a slip, we need to re-localize the tool tip or relearn the mapping between palm motion and tactile sense.

7. Conclusion

Robotic manipulation of human tools remains a hard challenge because robotic grasps of a tool (e.g., a drill) vary between trials. To handle this uncertainty, this paper presented a framework in which a robot, after firmly holding on to a tool, a) extracts from vision the transformation between palm and tool tip, which enables control of the tip's motion in mid-air and b) learns the relationship between hand motion/force and tactile sense to control the tool in contact with a surface. This adaptation makes the robot insensitive to the uncertainty about the tool's position and orientation within the grip. To test our framework, we looked at tasks that involve controlling position and force of a tool tip: placing a tool tip on a target, exerting a force at the tip, and sliding the tip along a surface.

Our experiments demonstrated the two key elements of our framework: In the first experiment, the DARPA ARM robot picked up an electric drill and drilled autonomously at a predefined target. The robot used visual feedback to localize the tool tip of a drill and estimate the transformation between tip

and palm. To control the force at the tool tip, the robot's kinematics were augmented using the estimated transformation which allowed the robot to exert a desired force at the tool tip. In the second experiment, an R17 robot drew with a pencil on a surface of unknown slope. Here, the robot used tactile feedback coupled with motor babbling to learn the relationship between palm motion and sensory input. In both experiments, the tool's relative position and orientation in the hand varied between trials. Our approach uses adaptation efficiently since the robot "learns" only the most uncertain kinematics, i.e., the transformation between palm and tool tip, which is (usually) linear. Moreover, our approach does not rely on accurate calibration and thus is suitable for unstructured environments, e.g., applications in the field or manufacturing of flexible materials.

Acknowledgments

The authors gratefully acknowledge the support of HRL Laboratories, LLC and helpful discussions with David Payton. The experiments on the DARPA ARM robot were supported by the DARPA ARM program under contract W91CRB-10-C-0126. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Distribution Statement 'A': Approved for Public Release, Distribution Unlimited.

Appendix A. Level set algorithm

This section presents our implementation of the level set algorithm [18], which generates a contour of an object within an image. We used this formulation of level sets due to its numerical stability.

Let $\Gamma(s) = [x(s), y(s)]^T, s \in [0, 1]$, be a closed curve in $\mathbb{R}^2$ and define an implicit function $\phi(x, y)$ such that the zeroth level set of ϕ is Γ, i.e., $\phi(x, y) = 0$ if and only if $\Gamma(s) = [x, y]^T$ for all $s \in [0, 1]$. For a region inside the curve, ϕ is initialized to -1, and for a region outside the curve ϕ is initialized to $+1$. To extract the boundary of the object, the goal is to minimize the energy functional [18], as below, which is an Euler-Lagrange equation,

$$\begin{aligned} E(c_i, c_o, \Gamma) &= \mu \cdot \texttt{Length}(\Gamma) \\ &+ \lambda_i E_i(c_i, \Gamma) + \lambda_o E_o(c_o, \Gamma) \quad , \end{aligned} \tag{A.1}$$

where μ, λ_i, and λ_o are constants (for the experiments in this article, we chose $\mu = 10$, $\lambda_i = 1$, and $\lambda_o = 1$), and c_i and c_o are the average intensities inside and outside of Γ. We used the regularized Heaviside function, $H(z) = \frac{1}{1+e^{-z}}$, as a differentiable threshold operator. With this function, the terms in Equation (A.1) are given as

$$\texttt{Length}(\Gamma) = \int_\Omega |\nabla H(\phi(x,y))|\, dxdy \quad , \tag{A.2}$$

$$E_i(c_i, \Gamma) = \int_\Omega |I(x,y) - c_i|^2 H(\phi(x,y))dxdy \quad , \tag{A.3}$$

and

$$E_o(c_o, \Gamma) = \int_\Omega |I(x,y) - c_o|^2 [1 - H(\phi(x,y))]\, dxdy \quad , \tag{A.4}$$

where I is the intensity of the gray image, and Ω is the image area.

We minimized the Euler-Lagrange equation using gradient descent [39],

$$\begin{aligned} \Delta\phi = &- |\nabla\phi| \cdot \\ &\left[\lambda_i |I(\mathbf{x}) - c_i|^2 - \lambda_o |I(\mathbf{x}) - c_o|^2 - \mu\nabla \cdot \left(\frac{\nabla\phi(\mathbf{x})}{|\nabla\phi(\mathbf{x})|}\right)\right] \quad . \end{aligned} \tag{A.5}$$

Finally, the complete algorithm is summarized as below:

1. Initialize Γ and ϕ,
2. Use ϕ to compute c_i and c_o,
3. Iterate the following steps until convergence:
   - Update ϕ using one gradient descent step according to (A.5),
   - Compute Γ from ϕ, and
   - Reinitialize ϕ.

References

[1] M Zinn, O Khatib, B Roth, and J K Salisbury. A new actuation approach for human-friendly robot design. Int. J. of Robotics Research, vol. 23, no. 4/5, pp. 379-398, 2005.

[2] A De Luca, B Siciliano, and L Zollo. PD control with on-line gravity compensation for robots with elastic joints: Theory and experiments. Automatica, vol. 41, no. 10, pp. 1809–1819, 2005.

[3] A Bicchi and G Tonietti. Fast and soft arm tactics: Dealing with the safety-performance tradeoff in robot arms design and control. IEEE Robotics and Automation Mag., vol. 11, no. 2, pp. 22–33, 2004.

[4] S Schaal, C Atkeson, and S Vijayakumar. Real time robot learning with locally weighted statistical learning. IEEE International Conference on Robotics and Automation, San Francisco, California, vol.1, pp.288–293, 2000.

[5] M I Jordan and D E Rumelhart. Forward models: Supervised learning with a distal teacher. Cognitive Science, 16, 307-354, 1992.

[6] H Hoffmann, G Petkos, S Bitzer, and S Vijayakumar. Sensor-assisted adaptive motor control under continuously varying context. Proc. International Conference on Informatics in Control, Automation, and Robotics, 2007.

[7] D Nguyen-Tuong, M Seeger, and J Peters. Model Learning with Local Gaussian Process Regression. Advanced Robotics 23(15) 2015-2034, 2009.

[8] O Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE Journal of Robotics and Automation, vol. 3, no. 1, pp. 43-53, 1987.

[9] M Prats and P J Sanz and Angel P del Pobil. A framework for compliant physical interaction based on multisensor information. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, 2008.

[10] B Siciliano and O Khatib (Eds.). Handbook of Robotics, Springer, 2008.

[11] K I Diamantaras and S Y Kung. Principal Component Neural Networks. Hoboken, NJ: Wiley, 1996.

[12] B Schölkopf, A Smola, and K-R Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Computation, 10:5, 1299-1319, 1998.

[13] H Hoffmann. Unsupervised Learning of Visuomotor Associations. MPI Series in Biological Cybernetics, Vol. 11, Logos Verlag Berlin, 2005.

[14] C R Rao. Linear statistical inference and its applications (2nd ed.). New York: John Wiley & Sons, 1973.

[15] T Hastie, R Tibshirani, and J Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics). Springer, 2009.

[16] C M Bishop. Pattern Recognition and Machine Learning. Springer, 2007.

[17] N K Paragios and R Deriche. A PDE-based level-set approach for detection and tracking of moving objects. In Proceedings of the 6th International Conference on Computer Vision, pages 1139–1145, 1998.

[18] T F Chan and L A Vese. Active contours without edges. IEEE Transactions on Image Processing, 10(2): 266277, Feb. 2001.

[19] R.B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3D Registration. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3212–3217, 2009.

[20] R.B. Rusu and S. Cousins. 3D is Here: Point Cloud Library (PCL). Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 1-4, 2011.

[21] H Hoffmann, P Pastor, D-H Park, and S Schaal. Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance. IEEE International Conference on Robotics and Automation, 2009.

[22] H Hoffmann, S Schaal, and S Vijayakumar. Local dimensionality reduction for non-parametric regression. Neural Processing Letters, Vol. 29, pp 109–131, 2009.

[23] K Kragic and H I Christensen. Integration of visual cues for active tracking of an end-effector. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 1999.

[24] C Harris and M Stephens. A combined corner and edge detector. Proceedings of the 4th Alvey Vision Conference, 1988.

[25] C Tomasi and T Kanade. Detection and Tracking of Point Features. Carnegie Mellon University Technical Report CMU-CS-91-132, 1991.

[26] D Lowe. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 60 (2): 91, 2004

[27] F Hoffmann, T Nierobisch, T Seyffarth, and G Rudolph. Visual servoing with moments of SIFT features. IEEE International Conference on Systems, Man, and Cybernetics, 2006.

[28] C C Kemp and A Edsinger. Robot manipulation of human tools: Autonomous detection and control of task relevant features. 5th IEEE International Conference on Development and Learning, 2006.

[29] T Drummond and R Cipolla. Real-time tracking of complex structures with online camera calibration. Proceedings of the British Machine Vision Conference, BMVC, Vol. 2, Nottingham, pp. 574–583, 1999.

[30] R S Dahiya, G Metta, M Valle, and G Sandini. Tactile sensing: from humans to humanoids. IEEE Transactions on Robotics, 26(1), pp 1–20, 2010.

[31] S Chitta, J Sturm, M Piccoli, and W Burgard. Tactile Sensing for Mobile Manipulation. IEEE Transactions on Robotics, 27(3), pp 558–568, 2011.

[32] J Sinapov, V Sukhoy, R Sahai, and A Stoytchev, Vibrotactile Recognition and Categorization of Surfaces by a Humanoid Robot. IEEE Transactions on Robotics, 27(3), pp 488–497, 2011.

[33] K Hsiao, L Kaelbling, and T Lozano-Pérez. Task-driven tactile exploration. Proceedings of Robotics: Science and Systems, Zaragoza, Spain, 2010.

[34] Y Bekiroglu, K Huebner, and D Kragic. Integrating grasp planning with online stability assessment using tactile sensing. Proceedings of IEEE International Conference on Robotics and Automation, 2011.

[35] K. Daniilidis. Hand-eye calibration using dual quaternions. The International Journal of Robotics Research, 18 (3): 286-298, 1999.

[36] Y. Shiu and S. Ahmed. Calibration of wrist-mounted robotic sensors by solving AX=XB. IEEE Trans. On Robotics and Automation, 5: 16-29, 1989.

[37] R. Tsai and R. Lenz. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. IEEE Trans. On Robotics and Automation, 3: 345-358, 1989.

[38] Y. Karayiannidis and Z. Doulgeri. Blind force/position control on unknown planar surfaces. IET Control Theory and Applications, 3(5): 595-603, 2009.

[39] P Chockalingam, N Pradeep and S Birchfield. Adaptive fragments-based tracking of non-rigid objects using level Sets, Proceedings of the IEEE Conference on International Conference on Computer Vision, Kyoto, Japan, 2009.