Abstract—In this work we propose a method to extract visual features from a tool in a hand of a robot to derive basic properties how to handle this tool correctly. We want to show how a robot can improve its accuracy in certain tasks by a visual exploration of geometric features. We also show methods to extend the proprioception of the robots arm to the new end-effector including the tool. By a combination of 3D and 2D data, it is possible to extract features like geometric edges, flat surfaces and concavities. From those features we can distinguish several classes of objects and make basic measurements of potential contact areas and other properties relevant for performing tasks. We also present a controller that uses the relative position or orientation of such features as constraints for manipulation tasks in the world. Such a controller allows to easily model complex tasks like pancake flipping or sausage fishing. The extension of the proprioception is achieved by a generalized filter setup for a set of force torque sensors, that allows the detection of indirect contacts performed over a tool and extract basic information like the approximated direction from the sensor data.

I. INTRODUCTION

Research in autonomous mobile robotics aims at extending the capabilities of mobile robots from fetch and deliver tasks to more comprehensive manipulation tasks. Many of these manipulation tasks require the competent use of tools. Robots need screw drivers to assemble book shelves, knives to cut salad [1], spatulas to flip pancakes [2], spoons to fish sausages out of the pot.

(a) Flip pancakes with a spatula. (b) Fish sausages with a skimmer.
(a) Pick up a tool. (b) Inspect tool. (c) Align tool.
(d) Push under target. (e) Lift pancake. (f) Flip pancake.

Fig. 1. The robot TUM-Rosie using tools for food preparation.
Fig. 2. TUM-Rosie flips a pancake.

The sequence of action steps consisting of the calibration of the tool in the holding hand, the extension of the body with the tool, and the use of force feedback from the tool is characteristic and necessary for the success of many tool
In this paper we investigate perception, learning, and computational methods for the automation of these action steps in tool handling. We propose a method for an autonomous analysis of a tool that starts with a visual inspection of relevant geometric features which are in our case edges, planar parts, concavities and holes. This allows us on the one hand to assign the tool basic capabilities, e.g. a mostly planar tool part with long and exposed edges can be used as a spatula, or a tool with concavities can be used as a spoon, given it has no holes. The second step is to learn a recognition model that represents a virtual force sensor for the end-effector of the tool. This sensor allows the robot to safely handle e.g. cooked material like a pancake by avoiding strong forces in certain directions, while still allowing stronger forces to enforce perfect alignment of a tool with any working surface. Our proposed description of the tool and of subtasks is a set of alignment- force- and position-constraints between tool features and workspace objects. We demonstrate a control based on such a description using the tool analysis described above.

The remainder of the paper is organized as follows: First we go shortly over related work. We will continue with the introduction of an ontology that represents tool classes based on the features we can detect. As the next step, we will show how we detect the features for the tools and continue with the explanation of how we build the virtual force sensor and integrate it into the robot’s proprioception. Then, we will explain shortly the controller that allows to specify constraints between workspace and tool features. Before we conclude the paper, we will present evaluations for the perception and the force sensing in the tool as well as some examples for the usage of the controller.

II. RELATED WORK

In prior work, affordances for tools were analyzed for artificial tools like in [3] in a low dimensional feature and workspace. In [4], Kemp and Edsinger visually estimate only a tool tip in order to act with this tool tip as a new end-effector. [5] discusses guidance of tools minimizing distances in two images from a stereo setup to guide a tool to a point. Given an unknown tool in the hand of the robot a 3D reconstruction can be performed [6], we nevertheless are not directly interested in a 3D model, but on a more accurate estimation of the previously mentioned features in the tool. Instead of detecting external tools, Michel et al. [7] presented a method to model the robots body itself including a tool, while targeting for collision avoidance mainly.

III. SYMBOLIC TOOL REPRESENTATION

The features SharpEdge, FlatSurface, Concavity and Hole were chosen based on the requirements we found for tools used by TUM-Rosie for food preparation. We additionally define the feature Handle, that the robot already has in the hand. We can describe the tools which we consider in this paper using mainly the transitive predicate properPhysicalParts taken from the KNOWROB system [8]. It expresses, that an object has a physical part and in this context it can be read as “has”. Additionally we use a simplification for numeric relations like min and max to express a more specific quantification than e.g. exists. By these features and predicates we can describe the tools we use here (see Figure 3) in the following way:

\[
\text{#Definitions:}
\]

\begin{align*}
\text{Class: } \text{HandTool} & \quad \text{properPhysicalParts min 1 Handle} \\
\text{Class: } \text{Blade} & \quad \text{properPhysicalParts min 1 SharpEdge} \\
\text{Class: } \text{Spatula} & \quad \text{properPhysicalParts min 1 Blade} \\
\text{SubClassOf: } & \text{HandTool} \\
\text{properPhysicalParts min 1 Concavity} & \text{properPhysicalParts max 0 Hole} \\
\text{Class: } \text{Skimmer} & \quad \text{properPhysicalParts min 1 Concavity} \\
& \quad \text{properPhysicalParts max 0 Hole} \\
\end{align*}

So, for example, a Skimmer has at least one concavity, at least one hole and (since it is a HandTool) at least one handle.

This ontology allows to classify tools based on the presence of (visual) features, as we will demonstrate in chapters IV and VII-A. A side effect of this classification is a semantic labeling of the tools sub-parts.

This labeling, in turn, is the key to grounding instructions like ‘align the blade with the pan’ or ‘push the spatula down on the pan’ in the robots perception. Given that the tool parts ‘blade of the spatula’ and ‘pan’ are localized, we demonstrate how such an explanation can be transformed into a control rule in section VI.

The features defined here might be able to express more tools than those three, but we had enough examples to evaluate the method on those three classes.

IV. VISUAL TOOL ANALYSIS

In this context we use a Microsoft Kinect sensor as a 3D sensor From this sensor we just take all points which are sticking out of the hand and are not the robots hand. These points form the first hypothesis for the form of the object. Then we try to fit a plane to it and analyze the structure of the distances of the points to the major plane. In this step we can distinguish planes from concavities and we get an estimation
of the 3D position of the border points which we can use to integrate with extracted lines.

Sensors like the Kinect have problems with shiny objects, so we avoid them in the test, assuming that this problems can be overcome by future sensors or software for 3D reconstruction. Most current 3D sensors have additionally problems with geometric edges so we use additionally a camera with higher resolution, which is calibrated to the Kinect and the robot. We extract edges in the images of this camera and fit lines to them given they are nearby the borders of the 3D body. We project the start and end points of 2D edges onto the plane which was extracted in the 3D data.

With an object in the hand, the robot can control the background as well as the observation angles. So, in this work we assume both to be already optimal and we work only on this optimal frame. Optimality in the observation angles can be based on the largest visible area of the tool, which can be easily achieved automatically given a 3D sensor that can measure the tool.

A. Flat Surface Extraction

In order to fit a plane, we perform first a simple thresholding in space to get points that belong most probably to the tool. The thresholds are known since we approximately known the relation of the potential tool to the robots hand.

We take all points in the object and perform a SVD on the matrix containing all points as rows in the matrix in order to use the resulting first eigenvector as plane normal. This simple analysis resulted for all tools in a reasonable approximation of a plane that at least intersects the tool at a relevant position. For planar tools the detected plane is additionally very close to the actual surface.

In order to distinguish from a non-planar or an object with a concavity we perform then the analysis described in the following section. This analysis gives us a height map encoding how close the plane is approximating the actual point cloud. Given we classify the object as non-concave, we can use this distance in order to optimize the plane to fit better by adjusting the orientation by fitting a plane into the height map. A result for a final plane fitting can be found in Figure 4.

B. Concavity Detection

In order to detect a concavity, we have to find a point on the tool that is lower than the border around it. Therefore, we create an image that contains the averaged distances of a certain volume over the extracted plane. This image is a height map, which look like the images depicted in Figure 5.

The image that results plotting the distances to the plane should form a minimum in the center and an elevation on the borders to form a concavity. We consider a convexity as a back-side view of a concavity, which just flips the sign of the comparison for concavities and will be treated as if we would have found a concavity on the other side.

To find the most probable point of a concavity, we search for the position and a radius of the two concentric circles that have the highest difference in their mean values under their border in the image we extracted. The circles are depicted in Figure 6(a), which also depict the three dimensions which are searched in order to maximize the difference in height between center and outer circle. This search can be done very fast because of the low resolution of our height map.
the concavity. An example for this representation can be seen in Figure 6(b).

C. Edge Extraction and Hole Detection

In order to find sharp edges at the tool we search for straight edges of a certain length. The steps of the edge extraction are visualized in Figure 7. We use a standard sub-pixel canny-edge filter, which avoids additional forks at edge intersections, visible in Figure 7(b). These edges we split into connected segments. We merge two of those segments if their edge direction is similar enough and the closest points on the two segments are close enough. The contours we split again based on collinearity, meaning if the direction along such a contour varies too much, it is split again. The resulting segments are shown in Figure 7(c). On all remaining segments of a reasonable size we fit a line by a robust least square regression, depicted as black lines in Figure 7(d).

All resulting lines can be intersected with the plane we extracted in the 3D data. In order to get a position in space, we can project the start and the end of a line onto this plane. The resulting 3D points define the line and its extreme points in 3D space.

By simple morphology and blob analysis we extract holes in the tool.

The existence of holes is reported as a result as well as the position and length of the edges in space. Only Edges of a significant length are reported, which are in case of the exemplary spatula the three visible in Figure 8(b).

V. TOOL INTEGRATION TO PROPRIOCEPTION

For integrating the tool with the robots proprioception, we assume a limited amount of touch- and/or force-torque sensors that are able to capture an impact experienced by a tool in hand or a force applied with a tool. Due to uncertainties in the grasp configuration and unknown force propagation in the tool we selected a statistical approach to modeling these sensor readings. In this paper we focus on the recognition and classification of an impact on the tool in hand.

The process of learning the proprioceptive model for a tool consists of three steps: First, we model the sensor input when we just move in free space with the tool. Second, the robot hits the tool to a known surface like a table in order to measure impact forces and train a classifier for them. Third and last, we use this classifier to detect online impact from certain directions.

A. Data Preprocessing and Noise Model

We monitor the joint-torque sensors of the robot’s finger with the respective tool in the hand. A first visual inspection of the sensors response due to an impact suggests that amplitude and slope may carry information about the nature of the impact.

So these readings are filtered individually by a filter bank in order to capture the changes of the sensor readings over a certain time. This filtering also serves to eliminate offsets in the sensor readings, which is caused by the grasp itself and changes with the hand orientation.

Our filter bank is based on an exponential smoothing filter, a simple one-tap Infinite-Impulse-Response (IIR) filter $F_\alpha(x[t])$:

$$F_\alpha(x[t]) = \alpha x[t] + (1-\alpha)x[t-1]$$

This is a very fast low-pass filter with an adjustable cut-off frequency. Based on this filter, we construct the following filter bank with $N$ filters:

$$y_0 = 0$$
$$y_i = F_{2i-N}(x_i - \sum_{j<i} y_j)$$

The idea is that every filter output $y_i$ is not affected by any signal that is already encoded in the lower-frequency filter outputs of $y_j$ with $j < i$. By disregarding the lowest two frequencies, the slowly changing offsets caused by orientation changes of the hand are eliminated. The outcome of this filtering using 8 filters per sensor is shown in Figure 9: For an impact of the tool on a table the filter response vector has a higher magnitude than a movement in space or no movement.

In order to detect events, sensor noise must be classified as such and ignored. To this end the robot performs some random free space movements with the tool in the hand.
The highest value is measured robustly for every filter output during the executed movements serves as a detection threshold.
Any impact should create an even higher magnitude of the IIR-filter output. For impact detection, a simple lowpass filter would have sufficient but the filter bank response is a much richer data set for the subsequent classification task.

B. Impact Learning

For learning impacts on the tool we create movements that represent the possible contact directions based on the detected features. Towards each of those directions, we let the robot execute movements. Planes and concavities have two, edges have only one, holes have zero possible contact direction.

For each sample movement we record the output of the IIR-filter. Since we approximately know the point in time when the input should happen, we take the highest filter output in this period and record it as a sample for the respective impact direction.

The sample vectors are normalized. Those samples have only one, holes have zero possible contact direction.

For detecting an impact, we use the same filter as during the learning. We use a spatula as a test object. According to the visual feature detection this object has three lines and a plane, shown in Figure 8(b). We call the contact directions of the plane X and -X, the contact direction of the side edges are called Z and -Z and the contact direction with the front edge is Y.

The collected training data is shown in Figure 10. For each impact a 12*4=48 dimensional vector is fed to the SVM (12 and -Z and the contact direction with the front edge is Y.)

We use a spatula as a test object. According to the visual feature detection this object has three lines and a plane, shown in Figure 8(b). We call the contact directions of the plane X and -X, the contact direction of the side edges are called Z and -Z and the contact direction with the front edge is Y.

The highest value is measured robustly for every filter output during the executed movements serves as a detection threshold.
Any impact should create an even higher magnitude of the IIR-filter output. For impact detection, a simple lowpass filter would have sufficient but the filter bank response is a much richer data set for the subsequent classification task.

B. Impact Learning

For learning impacts on the tool we create movements that represent the possible contact directions based on the detected features. Towards each of those directions, we let the robot execute movements. Planes and concavities have two, edges have only one, holes have zero possible contact direction.

For each sample movement we record the output of the IIR-filter. Since we approximately know the point in time when the input should happen, we take the highest filter output in this period and record it as a sample for the respective impact direction. The sample vectors are normalized. Those samples are used to train a Support Vector Machine (SVM) [9] with a class for each direction.

We use a spatula as a test object. According to the visual feature detection this object has three lines and a plane, shown in Figure 8(b). We call the contact directions of the plane X and -X, the contact direction of the side edges are called Z and -Z and the contact direction with the front edge is Y.

The collected training data is shown in Figure 10. For each impact a 12*4=48 dimensional vector is fed to the SVM (12 torque sensors, each filtered by 4 filters). The training samples are annotated with the contact directions and trained using a multi-class \(\nu\)-SVM with linear kernels and \(\nu = 0.1\) (The choice of \(\nu\) practically did not affect the result by much). In theory such a classifier is specific to the exact grasp configuration that it was trained with. But a manual validation after regrasping showed, that the SVM is capable of generalizing to at least slightly differing grasps.

C. Online Application

For detecting an impact, we use the same filter as during learning. If we detected an event we can infer the direction and assign the event to one of the features we extracted for the tool. With this knowledge we can easily react to inaccurate alignments of a certain line and e.g. the pan containing a pancake.

VI. TOOL-FEATURE BASED CONTROL

For the purpose of controlling we select some of the detected features, based on the size of the feature and distance from the hand.

Every tool feature is represented as a position and a direction: A line segment is encoded as direction and center point, a plane is encoded as the normal direction and its center. A concavity is represented as its deepest point and the normal vector of the plane that was used to find the cavity. This feature has has one implicit constraint: alignment with gravity.

Similarly, we define “features” for workspace objects like a pan or a table (both planes).

In order to control the robots position and orientation we establish position and alignment constraints between tool features and objects in the workspace. These constraints are expressed as functions mapping two features onto a scalar value.

We define four exemplary functions to model natural language constraints:

- \(\text{aligned}(f_1, f_2) = \text{dot}(\text{dir}(f_1), \text{dir}(f_2))\)
- \(\text{perpendicular}(f_1, f_2) = \sqrt{1 - \text{dot}(\text{dir}(f_1), \text{dir}(f_2))^2}\)
- \(\text{above}(f_1, f_2) = \text{dot}((\text{pos}(f_1) - \text{pos}(f_2)), \text{world}_z)\)
- \(\text{same\_position}(f_1, f_2) = \text{dist}(\text{pos}(f_1) - \text{pos}(f_2))\)

which can be computed for every pose of the robots hand. Our method requires the constraints to be continuous and differentiable over the space of possible hand poses.

We define a control objective as a set of triples \((c, f_1, f_2)\), consisting of of a constraint function \(c\), one feature \(f_1\) on the tool (e.g. an edge or a plane) and one feature \(f_2\) in the workspace e.g. a table.

To do control over these constraints, we numerically derive the constraint functions in their workspace coordinates. This yields an \(\nu\times 6\) task Jacobian \(J\) which we invert using the Moore-Penrose pseudo inverse:

\[
J^+ = (J^T J)^{-1} J
\]  

This inverse is used to compute a desired twist \(t \in \mathbb{R}^6\) for the robot hand that moves it closer to fulfilling the selected constraint.
constraints. Let \( v_d \in \mathbb{R}^n \) be the desired values for the constraints and \( v \in \mathbb{R}^n \) be the current values of the constraints. We then calculate \( t \) as

\[
t = J^+(K_p(v_d - v))
\]

Where \( K_p \in \mathbb{R}^n \) is a vector of coefficients. This essentially is a \( p \)-controller for each of the formulated constraints, each of which has a counterpart in natural language and it uses features that are grounded in the robots perception.

In our experiments the pseudo inverse could deal well with an underconstrained situation, i.e. with less than six non-conflicting constraints. However, when there were conflicting constraints and unactuated axes at the same time then the pseudoinverse became unstable: The control output for the supposedly unactuated axes became arbitrarily high. This could be fixed by adding “stay-here” constraints, that kept the unconstrained axes in the current position. When there are conflicting constraints, it is also possible to place them in a hierarchy and let some constraints execute in the null space of the others [10, 11].

### VII. Results

In this section, we will summarize the visual inspection results for the set of tools shown in Figure 3. We will show the results for impact direction identification and we want show in an image sequence how we can make use of the controller allowing to work with directions as constraints.

#### A. Visual Inspection

In the test setup, the tools are put into the hand of the robot, such that their position is approximately known (search space 15cmx15cmx15cm). We classify the concavity and extract the lines and holes. Table I shows the results we got for several trials for each of the tools. If there was variation in the results the minimal and maximal numbers of results are printed. The results show that in the setup on the robot we can robustly detect the features we are interested in. Visualizations for some selected results can be found in Figure 11. Only one of the spoons, namely the wooden spoon, could not be classified correctly. Its concavity was too small to be detected with the Kinect sensor. The measured maximal difference between the two circles was beyond the sensor noise of the Kinect. Big holes are detected completely accurate, while the number of small holes is not estimated accurately, but the existence of holes is detected correctly. Depending on the positioning of the tool in the search space, additional edges may be detected from the end of the tool-head towards the handle. Those edges created the variation in the number of detected edges, this is also not considered as an error, if the relevant edges were detected correctly, too.

With our simple ontology we could come from the features to a robust classification, which only fails for features that are smaller than the sensor resolution. The combination of a semantic model and the features we propose here turns out to be valid for the class of tools we investigate here.

![Image](319x343 to 374x405)

**Fig. 11.** Visualization of several results: First the small holes which could only be partially detected, then two examples for extracted edges.

#### B. Direction Classification with a Spatula

<table>
<thead>
<tr>
<th>Tool</th>
<th>Concavity/Plane</th>
<th>Num Edges</th>
<th>Num Holes</th>
<th>Resulting Tooltype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concavity</td>
<td>6</td>
<td>81</td>
<td>2</td>
<td>Skimmer</td>
</tr>
<tr>
<td>Plane</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>Spatula</td>
</tr>
<tr>
<td>Concavity</td>
<td>2-3</td>
<td>0</td>
<td>0</td>
<td>Spoon</td>
</tr>
<tr>
<td>Concavity</td>
<td>0-1</td>
<td>0</td>
<td>0</td>
<td>Spoon</td>
</tr>
<tr>
<td>Concavity</td>
<td>4-6</td>
<td>22-35</td>
<td>0</td>
<td>Skimmer</td>
</tr>
<tr>
<td>Plane</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>Spatula</td>
</tr>
<tr>
<td>Plane</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>HandTool</td>
</tr>
<tr>
<td>Plane</td>
<td>3-4</td>
<td>15</td>
<td>0</td>
<td>Spatula</td>
</tr>
</tbody>
</table>

**TABLE I**

The visual exploration results for the set of tools, errors are marked in red.

![Image](469x343 to 556x405)

![Image](516x702 to 595x832)

(a) Holes. (b) Edges of a skimmer. (c) Edges of a spatula.

**TABLE II**

The confusion matrix of the impact direction classification.

<table>
<thead>
<tr>
<th>Y</th>
<th>-X</th>
<th>X</th>
<th>Z</th>
<th>-Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>0.959</td>
<td>0.040</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>-X</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>X</td>
<td>0.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td>-Z</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.000</td>
</tr>
<tr>
<td>Z</td>
<td>0.001</td>
<td>0.046</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

The impact direction classification was performed using the 12 joint torque sensors of a DLR-HIT hand holding a spatula tool. The five model impacts were determined by the recognized visual features: A plane (-X, X), a front edge (Y), and two side edges (-Z, Z). After calibrating for the sensor noise the robot performs 20 hitting movements for each impact direction and records the strongest sensor reading of each impact. These \( 5 \times 20 = 100 \) sensor readings were fed to an
SVM classifier. For a confusion matrix, we randomly selected 25% of the data for training and 75% for validation. Repeating this process 100 times yields table II.

This table shows overall good results - only 4.5% of the impacts with the front edge (Y) and 4.6% of the inner side edge impacts (Z) were classified as impacts on the blade (-X). We validated this result by hitting the spatula with the wooden spoon from the learned impact directions. The classification was correct as long as the impact was strong enough to not be considered as sensor noise.

C. Alignment of Features

![Image](61x508 to 136x583) (a) Edge to align. (b) Upright object. (c) Laying object.

Fig. 12. The robot TUM-Rosie aligns a tool with an detected object, the two green lines show an edge of the spatula and a virtual point over the object along the blue z-Axis of the object.

To demonstrate alignment using the constraint based controller, we selected one edge of the spatula in the hand of the robot (left green line segment in Figure 12(a)) and the main direction of a bottle, moved 40cm up, in order to make the robot perform its movements in the air (right green line segment in Figure 12(a)).

The task of the robot was to hold the spatula line perpendicular to the bottle line and move it to the same position as the other feature. Using our constraint based controller the robot chooses a direct movement to fulfill these constraints. The robot first aligned its spatula while the bottle was standing on the table (Figure 12(b)). Then the bottle was turned and redetected by the robot when it was laying on the table. The robot re-aligned it’s spatula according to the new position of the bottle (Figure 12(c)).

VIII. CONCLUSION

In this paper we showed methods to go deeper into autonomous tool analysis. Based on a logical representation involving a set of geometrical features and their usage, we show how to visually localize these features on common household tools and classify the tools depending on which features were recognized. We showed that we can robustly detect the four features for the tool-classes spoon, skimmer and spatula, using a 3D camera and a robot that can position the tool for optimal observation angle and background.

We then showed how the robots proprioception can be linked to these tool features by classifying impact directions on them. This classification did not require any knowledge about the coupling of the involved sensors with the tool features and is applicable to unknown sensor configurations.

Finally, using the logical tool representation that is now grounded in visual and proprioceptive perception, we demonstrate the creation of control rules based on constraints which have a meaning that humans can easily access. We show how such a controller can fulfill alignment and positioning constraints involving tools.

REFERENCES