Preparatory Object Reorientation for Task-Oriented Grasping

Anh Nguyen, Dimitrios Kanoulas, Darwin G. Caldwell, and Nikos G. Tsagarakis

***Abstract*— This paper describes a new task-oriented grasping method to reorient a rigid object to its nominal pose, which is defined as the configuration that it needs to be grasped from, in order to successfully execute a particular manipulation task. Our method combines two key insights: (1) a visual 6 Degree-of-Freedom (DoF) pose estimation technique based on 2D-3D point correspondences is used to estimate the object pose in real-time and (2) the rigid transformation from the current to the nominal pose is computed online and the object is reoriented over a sequence of steps. The outcome of this work is a novel method that can be effectively used in the preparatory phase of a manipulation task, to permit a robot to start from arbitrary object placements and configure the manipulated objects to the nominal pose, as required for the execution of a subsequent task. We experimentally demonstrate the effectiveness of our approach on a full-size humanoid robot (WALK-MAN) using different objects with various pose settings under real-time constraints.**

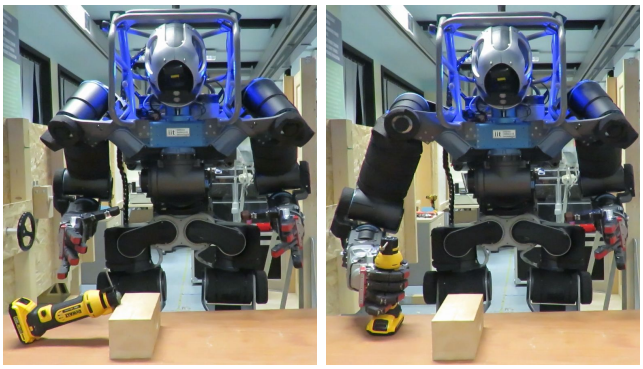
I. INTRODUCTION

Robotic grasping has been extensively studied over the last few years. Along with the manipulation trajectory planning and control, exteroceptive perception is a key aspect of autonomous grasping. Recent advances in visual perception have mainly focused on detecting grasps for *pick-and-place* tasks [1] [2] and localizing grasp affordances [3]. While this area is well studied, it is only recently that the problem of preparatory reorientation has been considered for object manipulation [4]. This human-inspired strategy plays an important role in manipulation when the object is not oriented in a pose that can be readily grasped.

Reachability limitations, a possibly infeasible starting object pose configuration, or even the energy consumption of the robot may prevent it from reaching and grasping an object at once as it is usually assumed in the literature. There is not yet any fully autonomous system that evaluates an object's pose and reorients it to a *nominal pose* such that it can be later used appropriately. For instance, a drill needs to be grasped in a particular way as shown in Fig. 1-(b) to be used. Arguably, there is no alternative way to achieve such a grasp starting from the configuration shown in Fig. 1-(a), unless the robot reorients it through a sequence of pre-grasps.

Pre-grasping in robotic manipulation was originally studied in [5] [4] from the planning point of view, with objects being rotated on a table using only a single grasp. In these approaches exteroceptive perception was not used, while the case where the final grasp is not reachable was also not considered. Moreover, the technique was limited to rotations around a single axis per object.

The authors are with the Department of Advanced Robotics, Istituto Italiano di Tecnologia (IIT), Via Morego 30, 16163, Genova, Italy. {Anh.Nguyen, Dimitrios.Kanoulas, Darwin.Caldwell, Nikos.Tsagarakis}@iit.it



(a) (b)

Fig. 1. An example of a grasping scenario. **(a)** The original object pose. **(b)** The nominal object pose after the reorientation.

In this work, we extend the idea of the preparatory object reorientation by adding visual perception in the loop, and more importantly by considering a sequence of grasps that will orient an object to the right usable pose, starting from any arbitrary one, and using only a flat support surface for the transitions. We first study the problem of pose estimation, which is challenging due to variations in the object geometric characteristics and the real-time task requirements. Then, we introduce a strategy for reorienting objects to their nominal pose using a sequence of pre-grasps. Using this technique, objects are placed in a kinematically reachable configuration, ready for the final grasp.

In particular, given a set of object images observed from various viewpoints, a database is generated offline with the object poses. For every image, a set of features is extracted along with their associated 3D correspondence points. These correspondences are generated by defining and fitting a bounding box on the object. Having defined the pose in such a way, we also denote an image per object as its nominal pose. After estimating the object pose, the rigid transformation between the detected pose and its nominal is calculated, considering only the rotation part. Finally, the robot plans and performs a sequence of rotations on the object, limited to the kinematic capabilities of the robot with the goal of bringing the object to its nominal pose.

Next we cover related work (Section II) followed by an overview of our approach (Section III). We then describe our pose estimation method in Section IV and introduce the reorientation strategy in Section V. Finally, in Section VI we present the experimental results on a full-size humanoid robot, reorienting different objects in real-world scenarios.

II. Related Work

Object grasping has a long history in the area of robotic manipulation [6] [7]. One of the main grasping approaches is to localize an object in the environment and then plan a grasp. For instance, the authors in [2] used a score function to measure the quality of the graspable 3D points on the surfaces of the object, achieving reliable grasps for novel objects. Instead of localizing whole objects, many works have focused on detecting grasp regions directly. A method was introduced in [8] to localize antipodal grasps in cluttered environments, by ranking all the possible grasps based on constraints formed by the robot hand and its finger configuration. The work of [3] proposed a method to detect grasp affordances using a deep learning approach. A grasp was defined as a rectangular box on the object that satisfies the robot's hand limitations, and its locations were learned by two deep neuron networks. Recently, the authors in [9] used a convolutional neural network to achieve high grasp success rate in dense clutter, while in [10] object affordances have been also used successfully for grasping.

Various other visual perception methods similar to those described above mainly focus on detecting grasping locations for *pick-and-place* tasks. They usually do not consider the case where an object needs to be grasped in a particular configuration that is not initially possible. The research that has been done on preparatory manipulation is very limited. Human pre-grasping strategies based on preparatory object rotation have been studied initially in [5]. Later in [11] the researchers from the same group developed this concept to find stable grasps. Similarly in [4] a pre-grasping strategy was used for completing transport tasks by planning object rotations. Recently, in [12] the authors introduced a robotic system that uses a gripping hand and external surfaces to perform difficult reorientation tasks like flipping. All these pre-grasping methods mainly focus on the planning aspect of the problem, to improve the success rate of grasping or transporting objects.

The method proposed in this paper, unlike [11] [4] and other in-hand object manipulation system [13], implements a fully autonomous system to estimate the pose of an object in real-time. Based on the detected pose, the robot generates online and in real-time a sequence of transformations to move the object to the desired nominal state. While previous methods focused mainly on improving the grasping success rate or the effort of moving an object, in our approach we are the first (to the best of our knowledge) to study the problem of developing a pre-grasping strategy to reorient an object to its nominal pose. This is a necessary strategy that allows the robot to finally grasp the object while still being able to reserve its kinematic reachability for other tasks.

III. An Overview of Our Approach

As shown in Fig. 2, the perception and the planning systems form a loop which repeats, until the final object configuration has been achieved. First, the object pose is estimated from an RGB-D image using a popular approach in computer vision based on the concept of feature descriptors.

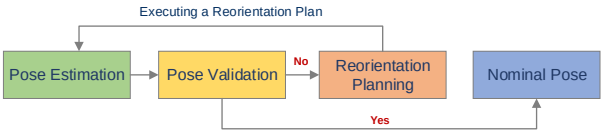


Fig. 2. The pipeline of our approach.

The main idea is to extract meaningful features from 2D images and establish the correspondence between them and their associated coordinates in the 3D space. To create these correspondences, we project the detected features onto 3D points using a predefined object bounding box. Each correspondence is computed offline and saved in a database. When a test image is acquired, its features are extracted and matched with those stored in the database in real-time. The geometric relationship between two matched features is then used to estimate the 6DoF object pose in real-time.

After estimating the object pose, our system validates the current pose of the object by comparing the found orientation with the nominal one in the database. If the object is at its nominal pose, the robot can perform the final grasp to use the object. Otherwise, a number of reorientation steps are planned to bring the object to the nominal pose. The goal of each reorientation is to move the object to a new configuration that is closer to the nominal. Each step is validated to make sure that the robot can physically execute it while keeping in the end the object steady on the table. Our method uses vision in the loop after each reorientation step to re-evaluate the current pose of the object and to plan the next step. The whole process is repeated until the robot successfully reorients the object to its nominal pose. The next two sections give the details of these steps.

IV. Pose Estimation

The pre-grasping process in this work involves an intermediate planar support surface like a tabletop. The nominal object pose (denoted by N) is the one that the object lies in a stable configuration on the support surface and it is in the right pose ready for the final grasp from its handle. To estimate the 6DoF pose, we follow the standard approach for textured objects introduced in [14]. However instead of using the full 3D object model, we only use the 3D bounding box to simplify the system. An overview of our pose estimation approach can be found in Fig. 3.

A. 3D Bounding Box

The pose of an object is defined using a 3D rectangular box that encapsulates it. For the nominal pose, an origin root point is first selected manually in a particular corner of the bounding box, while the local object frame is defined as follows. The first axis is along the horizontal direction, the second corresponds to the table's normal vector (i.e. the vertical direction), while the last axis is their cross product. For example in the estimated pose in Fig. 3-b, the $\mathbf{x}$-axis (in red) aligns with the horizontal direction, while the $\mathbf{y}$-axis (in green) with the table's normal vector. This frame is manually picked for the nominal pose and its position always

Fig. 3. The pose estimation method. (a) **Training:** For each image the features are first extracted from the object, then the correspondences between the 2D features and their associated points in 3D space are generated using the predefined bounding box. (b) **Testing:** The detected features are matched with all the features in the database. The matched correspondences will be used to compute the object pose.

remains consistent with the object position with respect to the camera. During the training step, we manually define a set $\mathcal{S}$ that stores the conditions for the axis directions in the local object frame in which the object can stand steadily on the support surface (e.g. in Fig. 3-b, $\mathcal{S} = \{\angle(\mathbf{y}, \mathbf{n}_t) = 0\}$ since we only consider the object to be at its steadily standing configuration on the support surface if the angle between its y-axis and the table's normal vector $\mathbf{n}_t$ is 0 degree).

B. Pose Estimation

We use the pinhole camera model to estimate the 6DoF pose of the object. In particular, given the camera intrinsic parameters matrix $\mathbf{M} \in \mathbb{R}^{3\times3}$, the homogeneous representation of a 2D image points $\mathbf{p} \in \mathbb{R}^3$ in the camera frame and its corresponding 3D point $\mathbf{P} \in \mathbb{R}^4$ in the local object frame. The object pose $[\mathbf{R}, \mathbf{t}] \in \mathbb{R}^{3\times4}$ is given by the following equation:

$$\mathbf{p} = \mathbf{M}[\mathbf{R}, \mathbf{t}]\mathbf{P} \tag{1}$$

with $\mathbf{R}$ the rotation matrix and $\mathbf{t}$ the translation vector.

To estimate the 6DoF object pose, in the training phase, we first extract n ORB [15] keypoints $\mathbf{p}_i$, $i \in [1, n]$ for each input image. These keypoints are projected onto the 3D local object space using the Möller-Trumbore algorithm [16] to establish the correspondences $\mathbf{p}_i \leftrightarrow \mathbf{P}_i$. This algorithm uses the mesh model of the predefined bounding box and the detected keypoints $\mathbf{p}_i$ as input, to compute their associated 3D coordinates $\mathbf{P}_i$.

In the testing phase, the extracted keypoints are compared with all the entries in the database using the FLANN matching technique [17]. From the matched pair, the $\mathbf{p}_i \leftrightarrow \mathbf{P}_i$ correspondences, that was established during the training, can be extracted. Given those we estimated the object pose $[\mathbf{R}, \mathbf{t}]$ from Eq. 1 by using the non-iterative EPnP technique introduced in [18]. Since outliers may exist in the found correspondences due to wrong matches, we use the RANSAC [19] algorithm to identify the set of inlier correspondences and run the pose estimation only for this set.

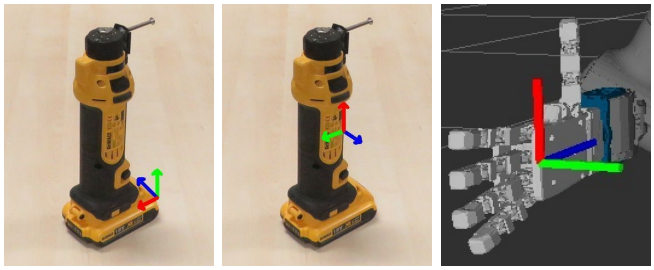
V. Object Reorientation

The goal of our method is to detect visually the current pose of an object and bring it to its nominal pose through a set of reorientation steps. The central idea of our approach is based on the difference between the current and the nominal pose that can be extracted from the database. Given this information, we develop a reorientation strategy, which incorporates perception to estimate the object's pose and an algorithm to reorient it to its nominal one. Visual perception plays an important role in this loop, since after each reorientation step it is used to evaluate the current pose and plan the next step. The whole process is repeated until the object reaches its nominal pose or the robot cannot grasp the object from its handle anymore.

A. Grasp Selection

To reorient an object, the first problem to consider is where it should be grasped from. In our application, the robot has to successfully grasp an object in such a way that it is not changing its original pose during the grasping, since this will cause an unwanted error to the reorientation result. This is particularly challenging in our system since we use an underactuated hand, and it appears that the object usually is shifted during the grasps. Even though a lot of methods have been developed for detecting grasp affordances [2] [3] [8], they do not guarantee to always find such a robust grasp, where an object is not moved using an underactuated hand.

In this work, we assume that the objects have a handle on which we manually define the grasp frame in each object's nominal pose. In particular, given the root frame of the estimated pose O_f, a rigid transformation is manually defined to transform this frame to the grasp frame G_f. The origin of G_f is chosen on the surface of the handle, while its orientation is defined as follows. The $\mathbf{x}$-axis co-aligns with the object's handle axis and the $\mathbf{y}$-axis points towards the handle axis, so that the hand can encapsulate more robustly the object. For each reorientation step, the grasp frame G_f co-aligns with the end-effector frame to generate the trajectory for the arm. In this way, we allow the grasps to be only around the object's handle, which reduces the chance of unwanted object rotations during the grasping. These frames are shown in Fig. 4.



(a) (b) (c)

Fig. 4. Grasping frames. (a) The estimated object pose. (b) The grasp frame transformed from the estimated pose. (c) The end-effector frame that needs to be coaligned with the grasp frame on the object.

B. Trajectory Planning

The motion planning has been used as a black box in our reorientation system. To plan the trajectory for the end effector, we first find a grasp frame G_f as described in the previous section. The orientation of the end effector is matched with the grasp frame G_f in order to perform the actual grasping. In practice, we use the Stack-of-Task (SoT) [20] method combined with OpenSoT [21] library to control the whole-body motion of the robot. The SoT method solves the inverse kinematics by employing cascaded Quadratic Programming solvers to efficiently find an optimum, in the least-square sense with a description of hierarchical tasks and constraints.

C. Reorientation Strategy

Given an input image including the object, the 2D features are extracted and matched with the right entry E in the database. Then, using the found 3D correspondences, the rotation matrix $\mathbf{R}_{CE}$ that represents the current pose of the object in the camera frame C, is computed. Furthermore, the nominal pose N orientation is pre-computed as $\mathbf{R}_{CN}$, giving the object's goal configuration in the camera frame. The rigid transformation $\mathbf{R}_{EN}$ between these two rotation matrices defines the reorientation action to be applied on the object to bring it in the nominal pose, and can be estimated as follows:

$$\mathbf{R}_{EN} = \mathbf{R}_{EC}.\mathbf{R}_{CN} = \mathbf{R}_{CE}^{T}.\mathbf{R}_{CN} \tag{2}$$

The estimated matrix $\mathbf{R}_{EN}$ provides the visual rotational information to reorient the object. However, due to the kinematic limitations of the robot arm, it might not be possible to reorient the object to its nominal pose in a single step. To deal with this problem, we propose a new algorithm (Algorithm 1) to generate a plan that is feasible for execution in each reorientation step.

In Algorithm 1, we first decompose the rotation matrix $\mathbf{R}_{EN}$ to its roll, pitch, yaw values and represent them as a set $\mathcal{A}_{r,p,y}$. To reorient the object to its nominal pose, the robot has to use its hand to manipulate the object along its three axes until all values in the set $\mathcal{A}_{r,p,y}$ reach zero (in practice, we use a small threshold ε close to zero). Now the problem of rotating an object from the current pose to the nominal pose has become that of choosing the order along the three axes to bring it closer to its nominal pose. To increase the successful rate of the real motion planning, we assume that in each reorientation step the robot will rotate the object based only on one axis. In particular, given a value $\mathcal{A}_i \in \mathcal{A}_{r,p,y}$, the robot will call the real motion planning function *Generate* that takes the grasp frame G_f and the expected angle to be rotated $\mathcal{A}_i$ as input, and generates the physical trajectory $\mathbb{P}_i$ for its arm. To make sure that the object after each reorientation step doesn't fall into an unexpected or a more complicated configuration due to unstable standing on the support surface, we evaluate every plan $\mathbb{P}_i$ using the *Evaluate* function. This function uses the predefined stable standing set $\mathcal{S}$ to check if the object pose after executing the plan $\mathbb{P}_i$ has any axis of its local object frame satisfying the conditions in $\mathcal{S}$. Intuitively, the accepted plan must satisfy two conditions: first to be physically executable and second to have the object stand steadily on its support surface after each reorientation step.

Algorithm 1 Generate Reorientation Plan

Input: Matrix $\mathbf{R}_{EN}$, grasp frame G_f, standing axis set $\mathcal{S}$
Output: Reorientation plan $\mathbb{P}$

$\mathbb{P} = \emptyset$
Decompose $\mathcal{A}_{r,p,y} = \mathbf{R}_{EN}$
while $\mathbb{P} = \emptyset$ and $\mathcal{A}_{r,p,y} > \varepsilon$ **do**
  for each $\mathcal{A}_i \in \mathcal{A}_{r,p,y}$ **do**
    $\mathbb{P}_i$ = $Generate(G_f, \mathcal{A}_i)$
    b = $Evaluate(\mathbb{P}_i, \mathcal{S})$
    if b **then**
      $\mathbb{P} = \mathbb{P}_i$
      break
    end if
  end for
  if $\mathbb{P} = \emptyset$ **then**
    $\mathcal{A}_{r,p,y} = \mathcal{A}_{r,p,y}/2$
  end if
end while

In the case that there are no kinematically feasible solutions for all three axes, we heuristically divide all the values in $\mathcal{A}_{r,p,y}$ in half, to look for a new less ambitious plan. The Algorithm 1 will stop when a feasible reorientation plan $\mathbb{P}_i$ is found. If the robot cannot find any feasible plan while all the values in $\mathcal{A}_{r,p,y}$ are divided repeatedly to less than ε, it means that due to kinematic constraints there are no solutions to grasp and rotate the object in any direction given the grasp frame G_f, even with a small angle. This problem can only be solved by searching for the a new grasp frame G_f at a different location on the object. The execution of each reorientation plan $\mathbb{P}_i$ will bring the object to the new configuration that is closer to its nominal pose, assuring that the object can stand steadily on the support surface. After each reorientation step, we again use the visual pose estimation method to re-estimate the object's pose and plan the next step. The use of vision in the loop is necessary since the object may be in a different unexpected pose after each grasp by the underactuated hand.

TABLE I

POSE ESTIMATION RESULTS AND ERRORS (IN DEG)

	Positions					*Average Errors*	
	P_1	P_2	P_3	P_4	P_5	$P_1 \rightarrow P_3$	$P_4 \rightarrow P_5$
Exp. Values							
$\mathcal{A}_{r,p,y}$	$(0.0, 0.0, 22.5)$	$(0.0, 0.0, 50.0)$	$(0.0, 0.0, 75.5)$	$(30.0, 0.0, 30.0)$	$(60.0, 0.0, 30.0)$		
Est. Values							
Juice Box	$(3.1, 5.3, 28.7)$	$(1.5, 2.5, 53.3)$	$(2.2, 3.5, 72.2)$	$(24.3, 4.6, 36.7)$	$(69.1, 6.3, 35.1)$	$(2.3, 3.8, 3.3)$	$(7.4, 5.5, 5.9)$
Chair	$(6.2, 3.2, 23.1)$	$(3.2, 4.6, 55.7)$	$(3.6, 5.1, 71.8)$	$(22.6, 5.3, 38.5)$	$(67.1, 4.7, 37.2)$	$(3.3, 3.2, 3.0)$	$(7.3, 5.0, 7.9)$
Drill	$(6.1, 3.3, 31.0)$	$(3.2, 0.3, 44.5)$	$(5.6, 0.8, 78.0)$	$(36.5, 3.5, 33.4)$	$(53.4, 1.8, 21.6)$	$(5.0, 1.5, 4.5)$	$(6.6, 2.7, 5.9)$
Hammer	$(4.3, 6.1, 31.1)$	$(2.6, 0.8, 47.6)$	$(4.2, 3.9, 69.2)$	$(38.5, 6.7, 25.7)$	$(53.2, 5.1, 22.6)$	$(3.7, 3.6, 4.8)$	$(7.7, 5.9, 5.9)$
Watering Can	$(7.2, 4.2, 33.6)$	$(2.4, 5.4, 56.7)$	$(3.5, 4.3, 81.1)$	$(34.5, 2.3, 37.3)$	$(66.4, 2.1, 36.4)$	$(4.4, 4.6, 6.8)$	$(5.5, 2.2, 6.9)$

VI. EXPERIMENTS

To validate the introduced method, we run experiments on the WALK-MAN full-size humanoid robot. We design different sets of experiments in order to verify all the components of our framework. We demonstrate that using our approach, WALK-MAN is able to reorient the objects in real life scenarios through one or multiple reorientation steps using its underactuated hand.

A. Robotic Platform

WALK-MAN [22] [23] is a 31DoF full-size humanoid robot which is 1.85 meters tall and weighs 118kg. It is designed to operate in man-made environments to assist or replace humans in damaged sites after natural or civil disasters. WALK-MAN is equipped with human-like underactuated hands (IIT-Pisa Soft Hand [24]), including five fingers that are driven by a single motor and are designed to adapt to the shape of the objects to be grasped. Each arm has 7DoF and the wrist is equipped with a 6DoF force/torque sensor. For the visual sensing a MultiSense-SL system is used, which can capture lidar and stereo data. The stereo vision, which we use in the work, acquires 1024×1024 RGB-D data in 15 frames per second. The communication with the robot is done with the use of the YARP middleware framework [25] that provides and delivers the high-level commands for motion primitives to the low-level torque controller. The control system runs on a computer with a Core i5 3.2GHz x 4 processor and 12GB RAM.

B. Experimental Setup

For the experiments, we used six objects as shown in Fig. 5. Throughout the experiments, the robot is positioned in front of a table, on which we place the object in a right-hand reachable position. The robot only uses its right hand to manipulate the object, while using the OpenSoT library [21] its Center-of-Mass is always kept in the convex hull guaranteeing stable balancing during the manipulation task. A preparatory reorientation is successful if the robot can finally bring an object to its nominal pose of up to 5 degrees error.

For each object, the local object frame is defined at the right bottom part of its bounding box in the nominal position. To create the database pose, we rotated the object by 5 degrees along its handle axis from each starting position. In the implementation the RANSAC outliers removal process runs in a loop of 1000 times, with the maximum inline distance and the successful confidence to be 2.5 and 0.95% respectively, while the minimum number of correspondences that are needed to estimate the object pose is set to 12. The threshold ε that indicates the acceptance error of the rotation angle in Algorithm 1 is set to 5 degrees.

C. Evaluating the Pose Estimation Method

To evaluate the efficiency of the visual system, we run our pose estimation method on all the test objects (except the clothes stand) in various positions. We first place the object in its nominal pose and we then manually rotate it to a known position. For each of them, the robot uses its vision system to estimate the pose of the object and calculates the difference $\mathcal{A}_{r,p,y}$ between the estimated and the nominal one. Table I summarizes the results. The first row of the table shows the expected values for each position, while the last two columns show the absolute average errors when the test objects are rotated around one (position $P_1 \rightarrow P_3$) or two axes (position $P_4 \rightarrow P_5$). The performance of our pose estimation method when the objects are rotated around one axis has approximately 5 degrees of error, while it becomes less accurate when the objects start from a more complicated configuration. We noticed that the main reason that decreases the quality of our pose estimation method is the existence of the outliers that cannot be removed by the RANSAC algorithm.

D. Reorienting with Underactuated Hand

To validate the full system, we run two different types of experiments. In the first case (*Case 1*), we position each object on the table standing on its support surface and the goal is to bring it to its nominal pose. The robot will have to rotate it only around its rotational axis which is perpendicular to the table in one or more steps. In the second case (*Case 2*), the object is placed in an arbitrary position on the table and the goal is to bring it in such a configuration that it stands on its support surface (not the nominal one which is studied in the first case). Table II shows the reorientation success rate (in $\%$) for both cases for 10 trials. Overall, we achieve a 72% success rate on the first case and a 46.7%

TABLE II

REORIENTATION SUCCESS RATE (IN %)

	Case 1	*Case 2*
Juice Box	60	40
Chair	60	40
Clothes Stand	n/a	80
Drill	80	40
Hammer	80	40
Watering Can	80	40
Average	**72%**	**46.7%**

success rate on the second one. We notice two main reasons that make the reorientations unsuccessful: (1) The estimated pose usually has around 5 degrees of error (as tested in the above section), and (2) since we use the underactuated hand with multiple contact points, the geometry of the object also affects the success rate. In the case that the object has a large support standing surface (e.g. the clothes stand) it is easier to bring it to its stable standing configuration, while it becomes more difficult when we reorient the object with a small standing support surface. For each reorientation plan, the total execution time is around 1 minute, and the time that is needed to estimate the object pose and search for a reorientation plan is approximately 45 milliseconds.

Using our approach, WALK-MAN can also reorient objects around more than one axis starting from a more complicated position. It is notable that with our method the robot can rotate objects to their nominal position with a relatively high success rate given the unstable grasping by its underactuated hand. This advantage is mainly achieved by using vision in the loop to re-estimate the pose after each step, where rotational errors were detected online and corrected during the next step's rotation planning. In the case that an object does not originally stand on its support surface, it is very important to successfully bring the object to its stable standing position in one step to guarantee that it doesn't fall into positions that are not kinematically reachable. Fig. **??** shows an example sequence of successful reorientation steps from an arbitrary pose to its nominal. The experimental video with all the objects from various stating poses can be found in the following link:

https://sites.google.com/site/preparatoryreorientation/

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a novel strategy to reorient objects to their nominal pose using a visual perception approach. The main goal of our framework is to bring the object to a better configuration so it can be used in later tasks. The contribution of the paper is the use of the current visually detected object pose to find the rigid transformation that is needed to reach the predefined nominal pose. A sequence of pre-grasps is planned and applied on the object, having the visual estimation always in the loop. To our knowledge, we are the first to use visual perception along with planning and control in a fully autonomous manner, where the goal position is visually determined and reached with a sequence of reorientation steps.

Fig. 5. Objects used in our experiments. **Top row:** A drill, a juice box, and a watering can. **Bottom row:** A chair, a hammer, and the lower part of a clothes stand.

The main limitation of our approach is the need to predefine a grasp frame for each object. In future work, we aim to overcome this limitation by developing an automated grasp-choosing system for the underactuated hands that can minimize the unwanted rotation during the grasps. Another interesting problem is to extend our approach to more difficult scenarios, where the objects may be in clutter environments or cases where the objects cannot be rotated into their stable standing positions in one reorientation step.

ACKNOWLEDGMENT

This work is supported by the European Union Seventh Framework Programme [FP7-ICT-2013-10] under grant agreement no 611832 (WALK-MAN).

The authors would like to thank Luca Muratore for helping with the experiments, and the anonymous reviewers for their useful comments.

REFERENCES

[1] R. Detry, C. Ek, M. Madry, J. Piater, and D. Kragic, "Generalizing Grasps Across Partly Similar Objects," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 3791–3797.

[2] I. Gori, U. Pattacini, V. Tikhanoff, and G. Metta, "Ranking the Good Points: A Comprehensive Method for Humanoid Robots to Grasp Unknown Objects," in *16th International Conference on Advanced Robotics (ICAR)*, Nov 2013, pp. 1–7.

[3] I. Lenz, H. Lee, and A. Saxena, "Deep Learning for Detecting Robotic Grasps," *International Journal of Robotics Research (IJRR)*, vol. 34, pp. 705–724, 2015.

[4] L. Y. Chang, S. Srinivasa, and N. Pollard, "Planning Pre-Grasp Manipulation for Transport Tasks," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2010.

[5] L. Chang, G. Zeglin, and N. Pollard, "Preparatory Object Rotation as a Human-Inspired Grasping Strategy," in *8th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2008, pp. 527–534.

[6] A. Sahbani, S. El-Khoury, and P. Bidaud, "An Overview of 3D Object Grasp Synthesis Algorithms," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, 2012.

[7] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-Driven Grasp Synthesis A Survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, April 2014.

[8] A. ten Pas and R. Platt, "Using Geometry to Detect Grasp Poses in 3D Point Clouds," in *International Symposium on Robotics Research (ISRR)*, 2015.

[9] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt, "High Precision Grasp Pose Detection in Dense Clutter," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[10] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Detecting Object Affordances with Convolutional Neural Networks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[11] D. Kappler, L. Chang, M. Przybylski, N. S. Pollard, T. Asfour, and R. Dillmann, "Representation of Pre-Grasp Strategies for Object Manipulation," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2010.

[12] W. Wan and K. Harada, "Reorientating Objects with a Gripping Hand and a Table Surface," in *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, pp. 101–106.

[13] K. Hertkorn, M. A. Roa, and C. Borst, "Planning in-hand object manipulation with multifingered hands considering task constraints," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 617–624.

[14] K. Pauwels and D. Kragic, "SimTrack: A Simulation-Based Framework for Scalable Real-Time Object Pose Detection and Tracking," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sept 2015, pp. 1300–1307.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, Nov 2011, pp. 2564–2571.

[16] T. Möller and B. Trumbore, "Fast, Minimum Storage Ray-triangle Intersection," *J. Graph. Tools*, vol. 2, no. 1, pp. 21–28, Oct 1997.

[17] M. Muja and D. G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration," in *International Conference on Computer Vision Theory and Application (VISSAPP)*. INSTICC Press, 2009, pp. 331–340.

[18] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An Accurate O(n) Solution to the PnP Problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2009.

[19] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun 1981.

[20] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical Quadratic Programming: Fast Online Humanoid-Robot Motion Generation," *International Journal of Robotics Research (IJRR)*, vol. 33, no. 7, pp. 1006–1028, June 2014.

[21] A. Rocchi, E. Hoffman, D. Caldwell, and N. Tsagarakis, "OpenSoT: A Whole-Body Control Library for the Compliant Humanoid Robot COMAN," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6248–6253.

[22] N. G. Tsagarakis *et al.*, "WALK-MAN: A High Performance Humanoid Platform for Realistic Environments," *Journal of Field Robotics (JFR)*, 2016.

[23] F. Negrello, M. Garabini, M. G. Catalano, P. Kryczka, W. Choi, D. G. Caldwell, A. Bicchi, and N. G. Tsagarakis, "WALK-MAN Humanoid Lower Body Design Optimization for Enhanced Physical Performance," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 414–420.

[24] M. Catalano, G. Grioli, A. Serio, E. Farnioli, C. Piazza, and A. Bicchi, "Adaptive synergies for a humanoid robot hand," in *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Nov 2012, pp. 7–14.

[25] G. Metta, P. Fitzpatrick, and L. Natale, "YARP: Yet Another Robot Platform," *International Journal on Advanced Robotics Systems*, vol. 3, no. 1, pp. 043–048, 2006.

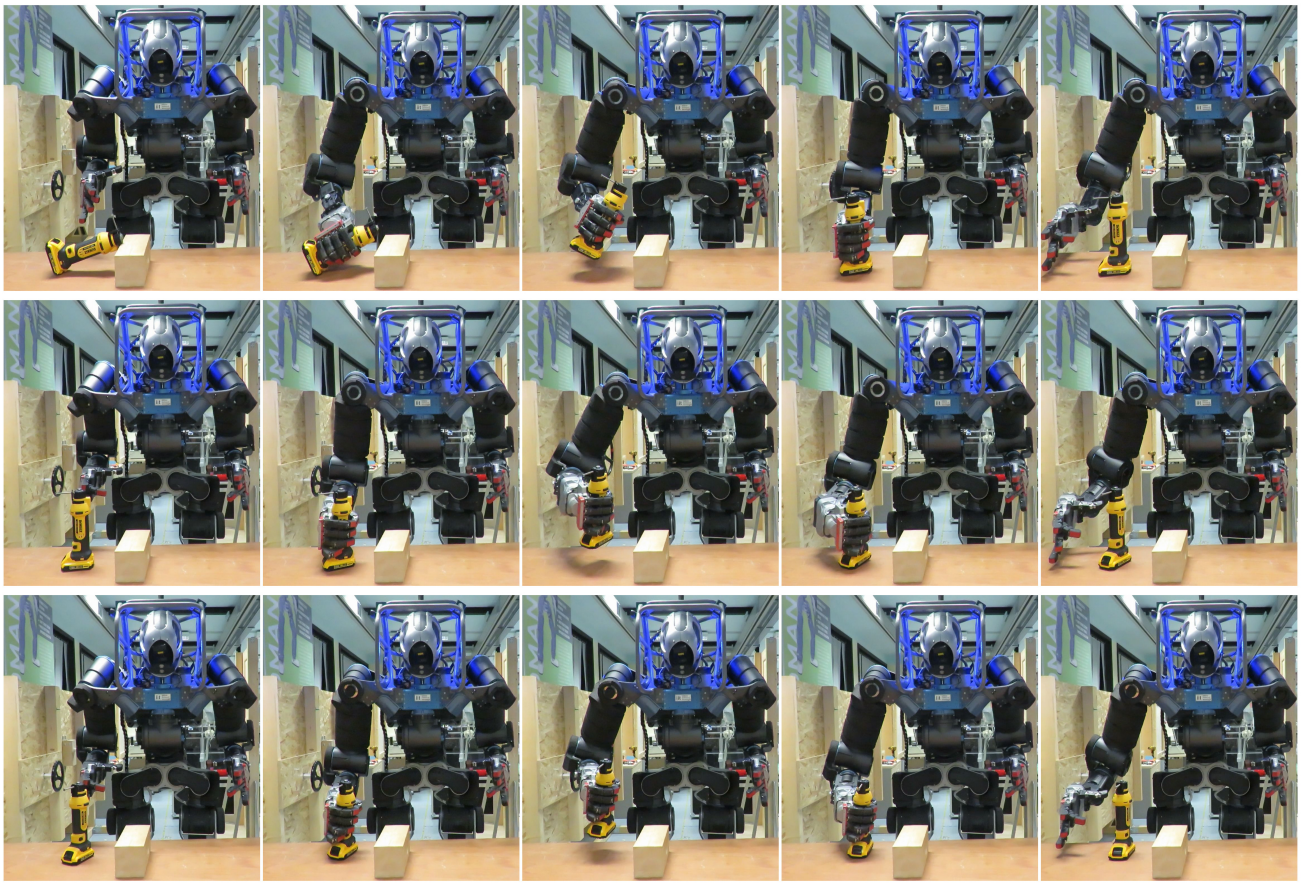


Fig. 6. Sequence of reorientation steps demonstrated on WALK-MAN. **From top to bottom:** Each row represents one reorientation step. The first row is the pre-grasp step when WALK-MAN rotates the drill around its yaw axis. The last two rows show the pre-grasp steps when WALK-MAN rotates the drill around its roll axis in two steps. **From left to right:** Five main states in each step: detect, grasp, lift up and rotate in the air, lift down, and leave the object.