A LEARNING METHOD TO DETERMINE HOW TO APPROACH AN UNKNOWN OBJECT TO BE GRASPED

BENJAMIN D. BALAGUER, STEFANO CARPIN

School of Engineering, University of California, Merced
5200 N Lake Rd Merced, CA 95343, USA
bbalaguer@ucmerced.edu, scarpin@ucmerced.edu

We present a learning algorithm to determine the appropriate approaching pose to grasp a novel object. Our method focuses on the computation of valid end-effector orientations in order to make contact with the object at a given point. The system achieves this goal by generalizing from positive examples provided by a human operator during an offline training session. The technique is feature-based since it extracts salient attributes of the object to be grasped rather than relying on the availability of models or trying to build one. To compute the desired orientation, the robot performs three steps at run time. Using a multi-class support vector machine, it first classifies the novel object into one of the object classes defined during training. Next, it determines its orientation, and, finally, based on the classification and orientation, it extracts the most similar example from the training data and uses it to grasp the object. The method has been implemented on a full scale humanoid robotic torso equipped with multi-fingered hands and extensive results corroborate both its effectiveness and real-time performance.

1. Introduction

The ability to grasp and move objects used in everyday human activities is one of the basic competences humanoid robots must exhibit in order to leave research labs and become useful companions in our everyday lives. Indeed, the general public expects robots to seamlessly integrate into daily life without having to condition the environment or carefully delimit the range of objects they will interact with. Needless to say, these legitimate expectations are far beyond what robotics research can deliver today. In this paper we address one of these problems, namely the ability to grasp objects. In particular, we focus on the capacity to apply this skill to previously unseen objects. As detailed in Section 2, grasping has already been extensively investigated since it is a core competence for humanoid robots to have. However, most solutions to this challenge rely on the availability of models for the objects being manipulated, or formulate stringent hypotheses concerning the class of objects the robot can interact with. On the contrary, our method does not start from these assumptions, but rather takes inspiration from the approach presented by Saxena et al.[1] who proposed a learning method capable of identifying good grasping points, in image-space, for a previously unseen object. The strength of Saxena's approach lies in its ability to greatly generalize from training data, thus

producing a system that does not rely on models or 3D shape reconstruction. Such method only identifies a good grasping point in the image, but to grasp an object the robot actually needs to approach it correctly and grab it. As our everyday experience suggests, when instructed to grasp an object at a certain point we may approach it in different ways. Hence, we are particularly interested in the orientation the hand (or end-effector) should assume in order to successfully complete the task. Obviously, certain orientations are more appropriate than others. Similarly to the aforementioned paper by Saxena, our method exploits supervised learning and does not rely on a priori availability of geometric models for the object to be handled, nor does it construct one online. Instead, the goal is to determine a correct approaching posture to grasp a novel object by generalizing from training data provided by a human operator. Moreover, emphasis is given to implement a system that is robust to noise and invariant to scale, translations, and rotations of the objects being considered.

Assuming that the robot is equipped with a stereo camera, given a picture and a point cloud of an object to be grasped, the goal is to determine the pose and orientation for the end-effector so that the robot can successfully grasp it. Saxena's method provides the pose, therefore in this paper we focus on the orientation, assuming that the position is given. Our method answers this question by extracting appropriate orientations from training data provided by a human operator in the form of positive examples showing how the robotic arm should approach a certain number of objects in order to successfully grasp them. At run time, a three layer approach is used where every layer reduces the search domain to identify an appropriate example in the training set. The first layer classifies the object to be grasped into one of the classes of objects used during training. This problem is solved using a multi-class support vector machine (SVM). Next, the orientation of the object to grasp is determined using plane fitting and central image moments. Finally, a nearest neighbor search identifies the most similar example in the training data matching the closest object class and orientation determined in the first two steps.

The contributions of this manuscript are as follows:

- we present a model-free hierarchical learning algorithm that exploits human imitations;
- minimal sensory input is necessary, namely a single stereo image;
- the learning process generalizes competently, allowing robots to grasp both trained and unseen objects;
- the algorithm is invariant to objects' scale, rotation, and translation as well as manipulator independent;
- the entire process runs in real-time, taking less than half a second.

We have implemented our method on a robotic torso with two arms equipped with multi-fingered hands, and we have demonstrated its effectiveness in practice. The robot successfully manages to grasp a variety of new objects, and the end-to-end processing time is in the order of half a second. We remark that the principal goal

and consequent contribution of this work come from the ability to compute, in real-time, valid grasps for untrained objects given a single stereo image. Consequently, we perform experiments without regards for planning in cluttered environments or dealing with partial occlusions since they are beyond the scope of this paper.

The remainder of the paper is organized as follows. Section 2 discusses related literature. A formal definition of the problem is then provided in Section 3. Section 4 is the core of this manuscript. It describes the training data provided to the robot and discusses the aforementioned three layer structure we propose. Section 5 introduces the experimental setup used to validate our proposal and presents detailed results showing the performance of the various layers, as well as end-to-end performance. Finally, Section 6 summarizes the lessons learned and provides directions for further investigations.

2. Related work

Grasping algorithms can be classified as model-based or feature-based. Although the line between the two can be blurry, model-based approaches generally rely on object models whereas feature-based techniques extract representative features from objects. Object models are represented in a multitude of different formats such as geometric primitives, point clouds, or three-dimensional meshes. Features that can be extracted range from point, corner, line, and edge detection to various convolution filters. Generally, both methods use a database of known models or features, labeled with appropriate grasps, which is used when trying to grasp an object. The algorithm we present in this paper is feature-based, but since its design was influenced by model-based methods we present previous work in both categories. We conclude this section with some related literature on human grasping and how it has influenced our research.

Two feature extraction algorithms frequently employed in feature-based grasping are SIFT[2] and SURF,[3] made popular for their invariance to scale, rotation, small view point changes, and illumination variations. While stemming from the computer vision literature, SIFT and SURF have been extensively employed in robotic manipulation,[4,5,6,7,8,9] generally to represent objects in a database that is then used to recognize objects seen by the robot and calculate their orientations. The principal drawback with these features, and the reason we do not use them, is that they are dependent on texture. For example, two geometrically-identical soda cans with different logos would require two different feature descriptors. SIFT and SURF are thus inappropriate for our application since we attempt to grasp completely novel objects. Piater exploits visual features from an overhead camera as a learning tool to calculate good grasps.[10] The learning approach is completely unsupervised since the manipulator mechanically tries different grasps and the visual features corresponding to a successful grasp are stored in a database. Even though the paper provides a rare online learning algorithm, it is limited in scope since it only applies to planar objects. This greatly reduces the difficulty of the problem

in terms of manipulation, feature extraction, and learning. Given the experiments were performed on very simple objects (i.e. a triangle, a circle, a square), it seems difficult to apply this algorithm to objects that cannot be represented as extrusions of simple shapes. Bowers and Lumia also attack the problem of grasping planar objects with the help of a vision system.[11] They use an overhead camera to create mappings between vision data and end-effector configurations. More specifically, blobs categorized as either a circle, ellipse, rectangle, or triangle are extracted from images, along with various parameters such as position, orientation, size, and elongation. The information extracted from the blobs can then be used to grasp objects. Even though the experiments on a real robot are very convincing, they only apply to relatively simple planar objects. Another approach to planar object grasping is presented by Morales et al.[12] An overhead image of the object is processed to find the object's contour and a grasp characterization metric is used to find grasp regions on the contour. This data is acquired through a contour curvature threshold and the force-closure criterion. While the algorithm can successfully grasp unknown planar objects of any complexity with potentially curved contours, the limiting assumption comes from the need for objects to have well-defined contours and being extrusions of those contours. This is not applicable to numerous objects in the real world. The algorithm is implemented on a real platform in a follow up publication,[13] using a decoupling between visual processing (i.e. finding a grasp given an image) and grasp execution. We view the link between visual processing and physical grasping similarly: as a hierarchical framework where the visual and grasping processes are independent. The authors re-examine their framework in which they supply more realistic examples such as scissors.[14] In addition, they make their algorithm modular with respect to the manipulator's hand configuration, a characteristic that we find crucial for manipulation algorithms. We take that characteristic further by making our algorithm manipulator-independent.

Moving away from the aforementioned limitations of planar object grasping, Remazeilles et al. offer a robust method to grasp previously unseen objects.[15] The authors exploit image features and visual-servoing in a reaction-based grasping procedure to slowly but accurately approach objects to be grasped. While providing good results and not requiring any learning, the work is intended for human-in-the-loop applications and assumes that an operator will draw a box around the object to grasp. Trying to replicate human grasping,[16] we do not embrace visual-servoing and provide a completely autonomous grasping algorithm. A different approach, which uses prior knowledge and online learning, is proposed by Kroemer et al.[17] Initially, a human demonstrates how to grasp a series of objects, each represented by Early Cognitive Vision descriptors.[18] The robot learns through a combination of Gaussian Processes Regression and Mean-Shift and the results from attempted grasps (i.e. successful or not) are incorporated into the robot's knowledge-base. The experimental results show that the robot quickly learns new grasps on its own and discards the initially learned ones.

Recently, a publication by Saxena et al. introduced a novel supervised learning

approach to feature-based grasping.[1] Instead of using features as object representations, the authors use features to represent good grasping points in image space. For training, each pixel in an image, labeled as a good or bad grasping point, is represented by a feature vector. When attempting to grasp an object, a binary classification (i.e. good or bad grasping point) is performed to classify each pixel in the image and choose the most likely good grasping point. The method is both accurate, provided a good training set, and efficient, thanks to the simple binary classification. The computationally expensive part of the algorithm, however, comes from the feature computation, which applies dozens of convolution operations on the image to extract the feature vector. These feature vector computations make for a non-real-time system with typical image resolutions. Recently we have improved this approach by using a feature selection process.[19] However, these algorithms only find a point to grasp in image space, but do not calculate an appropriate end-effector pre-shape to successfully grasp the object. In fact, this is the very issue that we address in this paper.

Early approaches to model-based grasping involved shape primitives (i.e. box, sphere, etc...), the combination of which accounted for the object model.[20] Each shape primitive is labeled with a strategy, including a grasp pre-shape, that can be exploited to grasp an object. The decomposition of an object into shape primitives is a major problem with such approach since there could be many different but valid decompositions and finding all of them would be time consuming - an issue avoided by the authors whose experiments were all performed on manually labeled objects. As is typically done in more recent model-based approaches, Goldfeder et al. utilize a database of three-dimensional synthetic object models labeled with valid grasps and attempt to match the object to grasp with the closest ones in the database.[21] They model each object in the database with a bag-of-features, composed of SIFT features from depth images, to encode different views of the objects. Given the bag-of-features for an object to grasp, a distance function allows to find the closest neighbors in the database and the best suited grasp. Our algorithm possesses similar components such as the use of an object database and the combination of range sensor data and features but our training data comes from real objects labeled with grasps shown by a human. This alleviates the assumption that there is a one-to-one correspondence between synthetic and real data. Although no temporal analysis is given, it is clear that the algorithm is computationally expensive due to the high-dimensionally encompassed in the database, a problem that we resolve in this paper. Alternatively, Sahbani et al. combine a learn-by-demonstration approach with grasp quality measures.[22] A part of each synthetic three-dimensional object is labeled as a good grasp region by a human. Finding valid grasps is then decomposed into finding the best object's part and applying a grasp quality metric[23] to the locations in that region.

We conclude this section by describing human manipulation research that has influenced the design of our algorithm. Specifically, human grasp strategies, de-

veloped in the first two years of infancy, relies primarily on learning (imitated[24] and reinforced[25]) and human senses (sight[26] and touch[27]). Furthermore, evidence suggests that, for humans, imitated learning is supplemented by sight,[24] whereas reinforcement learning is driven by touch.[28] Following human grasping, and given the fact that we are interested in imitation learning, we abstain from touch sensors, sensor-fusion, and object reconstruction and limit our algorithm's sensory input to a single stereo image. With humans arguably providing the best examples of dexterous manipulation, we use them as part of our training data acquisition in a learn-by-demonstration framework. Goodale et al. found that human processes that recognize object information (e.g. size, shape, color) are independent from the processes that compute a correct grasp.[29] The conclusions of this work were later validated by Castiello since the authors found that neural activities to encode object features and plan a grasp were indeed different.[30] Our multi-layer approach encompasses this dissociation by having a separate layer to recognize the object to grasp. In other words, we separate grasp planning from grasp execution and, consequently, we do not use reaction-based grasping or visual-servoing. Jeannerod has demonstrated the need for an hand pre-shape in human grasping[31] along with the fact that humans do not need to keep both the object and the hand in their field of view once a grasp plan has been established.[16] Our algorithm implements this approach by using the end-effector rotation as a grasp pre-shape and only requiring a single view of the object to be able to grasp it. Last but not least, Weir et al. found that there was no correlation between intrinsic object properties such as texture or weight when humans compute hand pre-shapes,[32] another finding that we exploit in our algorithm since we only rely on geometrical information to compute end-effector orientation.

3. Problem formulation

The grasping problem is, in general, very unconstrained. In order to formalize our contribution we make some operative assumptions. We assume the robot is tasked with the goal of grasping an object placed on a planar surface in front of the robot (e.g. a table). The object to be grasped can be reached by the robot by moving the arm only. In other words, we do not consider situations where the robot may need to walk to get closer to the object it has to grasp, neither does it need to squat or stand up. It is assumed the robot arm has n degrees of freedom, with $n \geq 6$. Let $\mathbf{q} = (q_1, \ldots, q_n)$ indicate a vector of n joint values specifying the end-effector pose. The object to be grasped has a size such that it can be grasped using one arm and a form-closure grasp. We do not make any hypothesis about the shape of the object, nor about its color. We do not make assumptions about its location or orientation, as long as it is within the arm's dexterous reachable space. The robot we consider is equipped with a stereo camera. This provides both an image I_g of the object as well as a point cloud P_g with depth estimates expressed with reference

to a known Cartesian global frame.[a] However, the stereo camera is not used to reconstruct a three-dimensional model of the object on the fly in order to match it against a library of models. Starting from I_g and P_g, our method shall output a configuration $\mathbf{q} = (q_1, \ldots, q_n)$ such that if the arm attains such configuration and closes its end-effector it grasps the object. This goal can be achieved by first computing the 4×4 transformation matrix specifying the pose and orientation of the end-effector, and then computing $\mathbf{q}$ via inverse kinematics. Let $\mathbf{T}_E$ be the matrix specifying the end-effector pose and orientation[33]

$$\mathbf{T}_E = \begin{bmatrix} & & & p_x \\ & \mathbf{R} & & p_y \\ & & & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where $\mathbf{R}$ specifies the rotation and $p_g = (p_x \ p_y \ p_z)^T$ provides the position. Given image I_g, the aforementioned paper by Saxena et al. computes a grasping point p_I in image space, i.e. it identifies a pixel in I_g where the object should be grasped at. Since the stereo camera provides both the image I_g and the corresponding point cloud P_g, and there is an inherent one-to-one relationship between pixels (in I_g) and points (in P_g), it is possible to extract its corresponding spatial coordinates, $p_g \in \mathbb{R}^3$, from P_g given p_I. Hence it may be assumed that the sub-problem of determining p_g is solved, and this is indeed the contribution presented by Saxena et al.[1] The focus of this paper, then, is to compute the rotational component $\mathbf{R}$, under the assumption (and constraint) that the pose component of $\mathbf{T}_E$ is already provided. The above consideration leads to formulate our question as the design of an algorithm that computes a function

$$f : \mathcal{I} \times \mathcal{P} \times \mathbb{R}^3 \to \mathrm{SO}(3) \tag{2}$$

where $\mathcal{I}$ is the space of images, $\mathcal{P}$ is the space of point clouds, and SO(3) indicates the special orthogonal group in $\mathbb{R}^3$, i.e. the space of three-dimensional rotations. We conclude this section stressing that, in our formulation, f is a function of the image I_g, of the point cloud P_g, and of the point p_g. In other words, we do not have control over how p_g is chosen, since it is the output of Saxena's algorithm. While it is true that p_g is actually a function of I_g and P_g, and then one could write $\mathbf{R} = f(I_g, P_g)$ (rather than $\mathbf{R} = f(I_g, P_g, p_g)$), the algorithm we propose in general would work even when p_g is chosen independently from I_g and P_g, as long as it is a valid grasping point. It should also be noted that we require the algorithm to compute $\mathbf{R}$ given a single view point, i.e. it is not possible to observe the object from different vantage points as is done in other methods.[21]

The system we present is scale, rotation, and translation invariant for objects. That is to say that it can cope with objects placed at different locations in the reachable area, and that their size may vary. For what concerns invariance to rotation, we

[a]Throughout the paper, subscript g is used to indicate data referring to the object to be grasped.

consider invariance to rotations about the axis orthogonal to the plane where the object is placed (such axis is indicated as z in the following). Rotations about other directions can be accounted for by providing additional training data encompassing those directions, but this extension will not be studied in this manuscript.

4. Learning how to grasp

In this section we first give a high-level overview of the algorithm and we then carefully describe its individual components in greater detail. The method we propose works by generalizing from a set of training instances showing how different objects could be grasped. Every training object, also referred to as *class*, is presented to the robot in numerous orientations about the z axis. During training, a picture of the object is taken, as well as the corresponding point cloud, and the orientation of the object is recorded. Moreover, for a subset of each object's orientations the robot is also provided with information about the right way to approach it in order to grasp it. This information is provided by a human supervisor placing the robot arm at appropriate grasping points.

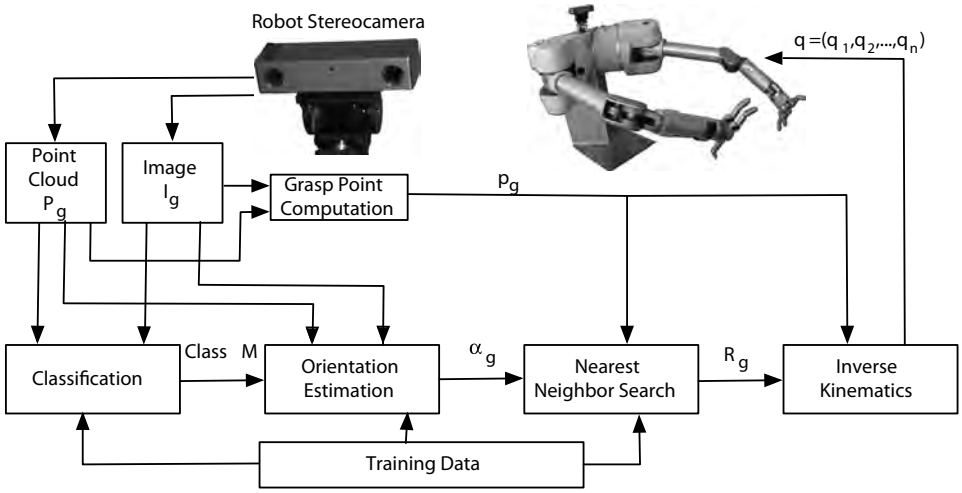

Fig. 1. A schematic representation of the system we present in this paper. The block labeled as "Grasp Point Computation" is implemented using Saxena's algorithm. The three layers and the grasp point computation blocks provide an orientation and pose for the end-effector that are then used to compute inverse kinematics.

At run time, the stereo camera mounted on the robot starts by providing an image I_g and a point cloud P_g of the object to be grasped (see Fig. 1). I_g and P_g are preliminarily used to compute the appropriate grasping point p_g using Saxena's algorithm. Then, the problem of computing the correct orientation is attacked in three stages referred to as *layers*, because they progressively prune the search space

in order to determine how to approach the object. The first layer performs a classification using a multi-class support vector machine and aims to classify the object to be grasped into one of the classes used during training. After this class has been identified, in the second layer the orientation of the object to be grasped is determined using a plane fitting technique and momentum matching. Then, based on the determined class and orientation, in the third layer a nearest neighbor search is performed in the training data to determine the correct approach pose. The nearest neighbor search takes place in a reduced search space taking into account the results of classification, estimated orientation, and grasping position p_g. Therefore, this final step is extremely efficient. At last, position p_g and orientation $\mathbf{R}_g$ are provided to an inverse kinematics algorithm that computes an appropriate joint configuration $\mathbf{q}$ for the robot.

From this description it emerges that training data is used for three purposes, i.e. object classification, orientation estimation, and nearest neighbor search. These different sub-problems are mirrored by two sets of training data. To solve the first two, classification and estimation orientation, we train the system using k instances (I_i, P_i, α_i) where I_i is an image of the object to be grasped, P_i is the corresponding point cloud, and α_i is the orientation of the object with reference to a predetermined global frame. For the third sub-problem (i.e. the nearest neighbor search), we consider a subset of orientations for each class, and for each of these we record a number of appropriate grasping positions. Training data in this case then consists of sets $(P_i, \alpha_i, \mathbf{T}_{Ei})$, where P_i, and α_i are defined as above, while $\mathbf{T}_{Ei}$ is the end-effector pose and orientation demonstrated by the human operator. It is important to stress that for a fixed object class and orientation the operator provides multiple valid grasping poses. Therefore there may be numerous training instances $(P_i, \alpha_i, \mathbf{T}_{Ei})$ with the same P_i, and α_i, but different $\mathbf{T}_{Ei}$. Moreover, for a given $\mathbf{T}_{Ei}$ let p_{Ei} be the corresponding position, and $\mathbf{R}_{Ei}$ the corresponding orientation.

A performance assessment of the individual layers is deferred until Section 5 where all experimental results are illustrated. In the remaining part of this section we describe the training data and the three layers.

4.1. *Training data*

Training is performed using 6 different objects that are shown in Fig. 2. Training objects differ in size, shape, and color. Of course, a richer set of training objects is expected to improve the performance of the system, but we have experimentally determined that already with $m = 6$ different classes (i.e. objects) the algorithm performs well.

4.1.1. *Training for classification and orientation estimation*

For classification and orientation estimation purposes we collect k instances (I_i, P_i, α_i). Since we aim for an algorithm with rotational invariance, during the training stage every object is considered in 36 different orientations. In other words,

Fig. 2. A coffee can, a drill, a mouth wash bottle, a mug, a detergent bottle and a plastic bottle were used for training. While discussing the results we will refer to them as object 1 to 6 (with 1 being the leftmost one).

for every object, 36 different values for α_i are considered. To be precise, every object is rotated about z in increments of 10 degrees. Therefore a total of $k = 6 \times 36 = 216$ instances are acquired to solve the first two sub-problems. These will be indicated as I_i^c, P_i^c, and α_i^c, with $1 \leq i \leq 36$ and $1 \leq c \leq 6$. Point clouds provided by the stereo camera are inherently noisy. Moreover, the point cloud necessarily includes not only the object, but also the supporting table (see Fig. 3(a) and 3(b)). Hence, right after acquisition, every point cloud is post-processed to remove noise and the points belonging to the table.

In order to remove points belonging to the table we use a process based on a previously-published algorithm.[34] A best-fit plane is fitted to point neighbors in the point cloud. If the best-fit plane is approximately parallel to the robot's horizontal base, we eliminate all points whose distance from the plane falls below a given threshold ε_T (see Fig. 3(c)). This process is repeated for different point neighbors until a number of different point neighbors have been tried or a sufficient number of points have been removed. This technique applies a simple heuristic, with the evidently valid assumptions that the table's top is parallel to the floor with the robot being either on the table or the floor. While being straightforward, it works well in practice[35] without assuming explicit knowledge of the distance between the camera and the table. Therefore, it can also be used in more general experimental conditions where, for example, the robot might change its posture and position with respect to the table.

Once points belonging to the table have been removed from the point cloud, an outlier detection algorithm is run to remove spurious readings. To this end, we use the method based on quartile ranges described by Laurikkala et al.[36] More specifically, two thresholds ε_L and ε_U for lower and upper outlier classification, respectively, are defined as follows:

$$\varepsilon_L = LowerQuartile - Step \tag{3}$$

and

$$\varepsilon_U = UpperQuartile + Step. \tag{4}$$

The *Step* variable is usually taken, as we do in this paper, to be 1.5 times the

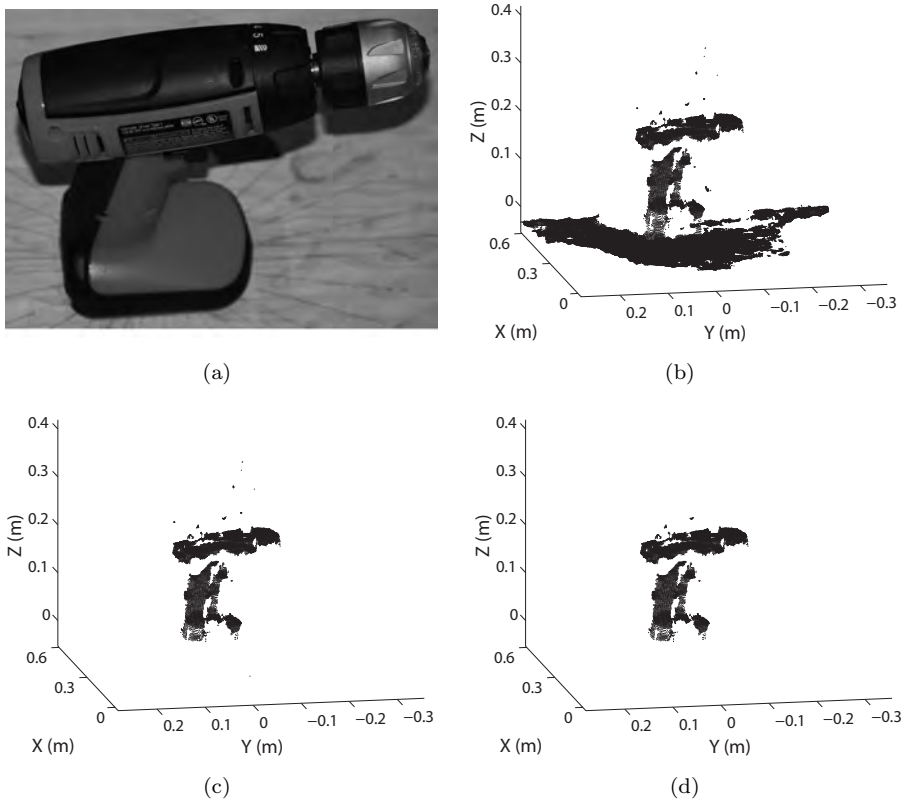

Fig. 3. Fig. 3(a) shows the training object (drill) as seen from the robot stereo camera. Fig. 3(b) displays the noisy point cloud returned. The dataset includes both points belonging to the table and noise. Fig. 3(c) shows the point cloud after points belonging to the table have been eliminated and Fig. 3(d) illustrates the final result after outliers have been removed.

interquartile range (i.e. $UpperQuartile - LowerQuartile$), although that number can be modified depending on the expected noise in the data. If p is a data point, it is labeled as an outlier and removed from the point cloud if $p < \varepsilon_L$ or $p > \varepsilon_U$. We note that we perform this process three times, for each X, Y, and Z coordinate independently, in order to equally remove outliers in each coordinate direction. An example of a resulting point cloud is shown in Fig. 3(d).

The two-step denoising process works well in practice, is efficient, and does not require human intervention. For more complex situations, more sophisticated methods for environment detection and outlier detection can be exploited.[37,35]

4.1.2. *Training for nearest neighbor search*

The most time consuming offline step consists in recording valid end-effector poses and orientations associated with the training objects. For a given object at a given orientation, a human operator moves the robot arm to a variety of appropriate grasping positions around the object to be grasped, as shown in Fig. 4. This task is facilitated in our experimental setup because the specific robot we use features a "gravity compensation mode" that allows to easily move it around. Once a valid pose is achieved, the human operator records the position. Then, by reading the joint values and computing forward kinematics, the end-effector transformation matrix $\mathbf{T}_E$ can be computed and recorded. Since this process is the most time consuming, it is performed only for a subset of orientations. Specifically, it is performed only for 12 of the 36 views, in increments of 30 degrees about the z axis. However, as anticipated, multiple grasping positions are recorded for the same view. Therefore, this stage produces training instances of the type $(P_i^c, \alpha_i^c, \mathbf{T}_{Eic}^j)$ where $1 \leq c \leq 6$, $1 \leq i \leq 12$, and j varies between 126 and 525 depending on the specific object class being considered. Typically, more training points are provided for highly asymmetric objects, and less for symmetric objects. According to this notation, $\mathbf{T}_{Eic}^j$ is the j-th grasping example provided when considering the c-th object class in the i-th orientation. Overall, a total of $n = 23984$ instances $(I_i^c, P_i^c, \mathbf{T}_{Eic}^j)$ were collected, and Table 1 gives more details for each class.

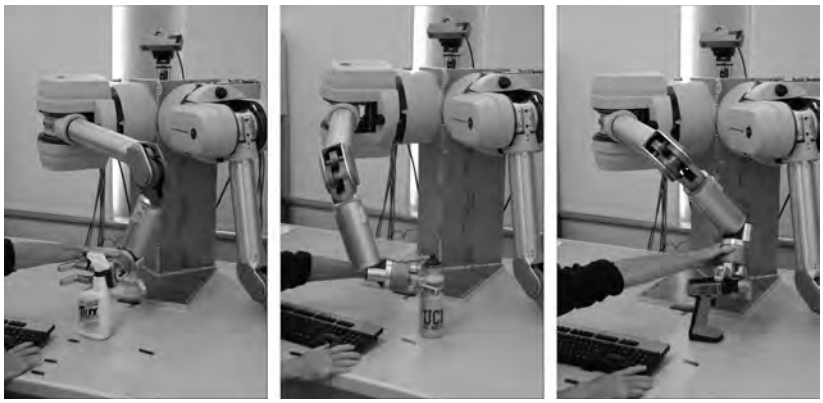

Fig. 4. A human teaching the robot to grasp a detergent bottle (left), a plastic bottle (center), and a drill (right).

4.2. *Layer 1: classification*

The classification sub-problem is solved using a multi-class support vector machine (SVM).[38,39] A classification problem with m classes can be solved in two ways. In the *one-against-all* paradigm m SVMs are used to separate every class from the

Table 1. Number of grasping examples provided for each of the six object classes considered (see also Fig. 2). Each object is assigned a number in order to simplify the results' discussion.

Object	Object Number	Number of Grasping Examples
Coffee can	1	5607
Drill	2	3623
Mouth wash bottle	3	3619
Mug	4	1795
Detergent bottle	5	4419
Plastic bottle	6	4885
Total		23984

remaining $m-1$. Alternatively, in a *one-against-one* setup one trains $\binom{k}{2}$ SVMs to separate each couple of classes. The one-against-one approach is in principle more expensive to use, but preliminary results showed that for the problem at hand it outperforms the other method in terms of accuracy, a fact corroborated by Hsu et al.[40] Moreover, even though the number of SVMs is higher, every individual classification problem is simpler, so a competitive tradeoff between speed and accuracy is obtained. In our implementation every SVM uses a polynomial kernel, since preliminary experiments outlined it performs better than a kernel based on Gaussian Radial Basis Functions.

The use of SVM to classify the object starting from a point cloud poses however a challenge. Indeed, in order to use an SVM a constant size feature vector needs to be associated to all training and input data. However, as one may easily intuit, the various point clouds include a variable number of points, and therefore an additional post-processing step is needed in order to extract a constant size feature vector from every point cloud. Hence, we need a method that, given a point cloud P_i^c, returns a fixed-size feature vector F_i^c. This conversion is performed for all point clouds, both for training data and for data acquired at run time, and works as follows.

Similarly to other publications,[41,42] for each point cloud P_i^c we first generate a new point cloud, $P_i'^c$, with zero mean. Indicating with $M_i^c \in \mathbb{R}^3$ the Euclidean mean of P_i^c, the new point cloud $P_i'^c$ is obtained from P_i^c by subtracting M_i^c from all its points. Zero mean point clouds are created to implement an algorithm that is translation invariant. Indeed, the point clouds *lose* any global spatial information, which was originally dependent on the camera and object locations. Next, we convert the point clouds $P_i'^c$ to fixed-size feature vectors F_i^c. Algorithm 1 shows pseudo-code to convert a point cloud to a feature vector, where we exploit the one-to-one relationship between stereo images and point clouds. The feature F_i^c is a $gridSize \times gridSize$ matrix, stored in row-major vector format. Every element in the matrix summarizes the information of a set of points in the point cloud: the value stored in one element is the average of the points lying in an associated region. One can think of this process as layering a grid on top of the depth image acquired through stereo vision

and taking the mean of all the points, in Cartesian coordinates, that are included in each cell. In order to remain fixed-size across all objects, the grid is fitted to the minimum $(\min_{Row}, \min_{Col})$ and maximum $(\max_{Row}, \max_{Col})$ pixel coordinates of the object. Consequently, the number of pixels encompassed by each cell in the grid $(\Delta Row, \Delta Col)$ adapts to the object. We use a $gridSize$ of 50, yielding 2500 cells, and a constant feature vector size of 7500 (each cell is comprised of three Cartesian components). It is important to note that this encoding not only creates a fixed-size feature vector necessary for SVM but also makes our algorithm scale invariant. Indeed, the grid automatically adjusts to changes in object size that could occur from being at different distances from the robot's camera. We conclude this section by acknowledging that other methods could be used to create F_i^c such as circular and radial sampling filters.[43] These methods, however, have not been investigated because the simple approach we sketched already gives satisfactory results.

Algorithm 1 Computation of F_i^c from $P_i'^c$

$gridSize \leftarrow 50$, $F \leftarrow 0$, $Count \leftarrow 0$
$\Delta Row \leftarrow \lfloor(\max_{Row} - \min_{Row})/gridSize\rfloor$ *// Number of row pixels per cell*
$\Delta Col \leftarrow \lfloor(\max_{Col} - \min_{Col})/gridSize\rfloor$ *// Number of column pixels per cell*
for all $p \in P_i'^c$ **do**
  $r \leftarrow \lfloor(Row(p) - \min_{Row})/\Delta Row\rfloor$ *//Row of cell $\in F_i^c$ encompassing p*
  $c \leftarrow \lfloor(Col(p) - \min_{Col})/(\Delta Col)\rfloor$ *// Column of cell $\in F_i^c$ encompassing p*
  $F_i^c(r \times gridSize + c) = F_i^c(r \times gridSize + c) + p$ *// Sum of all p_s in cell*
  $Count(r \times gridSize + c) \leftarrow Count(r \times gridSize + c) + 1$ *// Pixels in cell*
end for
for $k = 1$ *to* $size(F_i^c)$ **do**
  $F_i^c(k) = F_i^c(k)/Count(k)$ *// Compute the mean of the points in each cell*
end for

4.3. *Layer 2: Determining Object Rotation*

Layer 1 classifies the object to be grasped into one of the m classes, say class M. In layer 2, starting from P_g and M we determine α_g - the orientation of the object to be grasped. This is done by contrasting P_g with $P_1^M, P_2^M, \ldots, P_{36}^M$, i.e. by restricting the search domain to class M only. Before introducing the method we reiterate that we are only concerned with identifying the orientation about the z axis. This is consistent with the approach we took while collecting training data.

This problem of orientation estimation is solved using a plane fitting technique complemented by a momentum search to deal with symmetries. To be precise, for every point cloud in the training set we preliminary (offline) compute the best-fit plane using orthogonal distance regression,[44] and store its normal (see Fig. 5). Let N_i^c be the normal to the best-fit plane computed for P_i^c. At run time, given P_g, we compute its best-fit plane using the same technique, and let N_g be its normal.

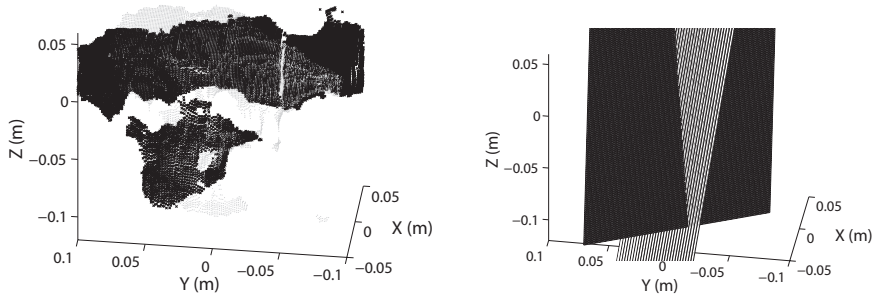

Fig. 5. Fig. 5(a) shows two overlapping point clouds for the training object drill placed at two different orientations. Fig. 5(b) shows the computed best-fit planes that clearly differentiate the two.

A set of 6 initial estimations for α_g is determined as follows. We start with the vector

$$S = sort_i \left(\arccos \left(\frac{N_g^T N_i^M}{||N_g|| ||N_i^M||} \right) \right) \tag{5}$$

where N_g^T is the transpose of vector N_g and the function *sort* orders the values in increasing order, returning a vector of indices. In other words, $\alpha_{S[1]}$ is the orientation of the training point cloud $P^M_{S[1]}$, whose best-fit plane normal forms the smallest angle with N_g. Evidently, $\alpha_{S[2]}$ is the orientation of the training point cloud $P^M_{S[2]}$, whose fitting plane normal forms the second-smallest angle with N_g and so on. As such, our six candidate orientations are $\alpha_{S[1]}$, $\alpha_{S[2]}$, $\alpha_{S[3]}$, $\alpha_{S[4]}$, $\alpha_{S[5]}$, and $\alpha_{S[6]}$. The choice of six candidates was made to add robustness under the assumption that the same object rotated by 180 degrees would yield similar best-fit planes, along with objects rotated by ± 10 degrees. In other words, we try to accommodate ± 10 degrees from a rotation of 0 degrees (3 candidates) and ± 10 degrees from a rotation of 180 degrees (3 candidates). In order to choose one of the six candidate orientations a comparison based on central image moments is performed. This central image moment comparison is used to differentiate views rotated by 180 degrees, which would generate similar planes. We first convert the camera image into a binary image, B, where pixels that are part of the object are set to 1 and the rest are set to 0 (see Fig. 6(a) and 6(c)). This conversion, done offline, is straightforward thanks to the one-to-one correspondence between a point in the point cloud and a pixel in the camera image. Image B is consequently composed of an object region, $\mathcal{R}$ (i.e. the set of pixels equal to 1). We divide $\mathcal{R}$ into left and right segments each having their own regions, $\mathcal{R}_L$ and $\mathcal{R}_R$ respectively as shown in Fig. 6(e) and 6(f). We calculate the central moments of order i, j using the formula

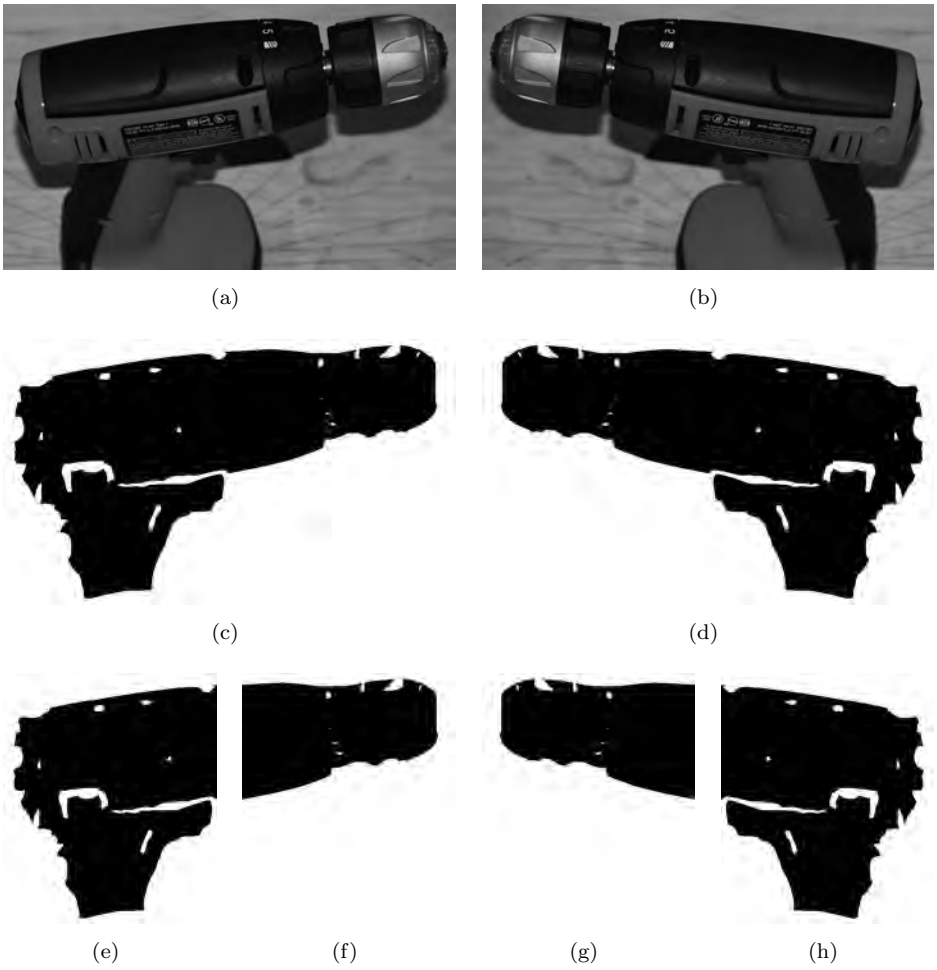

Fig. 6. Fig. 6(a) shows the cropped image of an object (drill). Fig. 6(c) displays the equivalent binary image B, with the object region $\mathcal{R}$ in black. Fig. 6(e) and 6(f) show the left and right image segments with regions $\mathcal{R}_L$ and $\mathcal{R}_R$ in black, respectively. Fig. 6(b), 6(d), 6(g), 6(h) shows the same information for the same object perfectly rotated by 180 degrees.

$$\mu_{i,j} = \sum_{X,Y} (X - \bar{x})^i (Y - \bar{y})^j \tag{6}$$

where $X, Y \in \mathcal{R}_L$ for the central moment of the left region, $X, Y \in \mathcal{R}_R$ for the central moment of the right region, and $\bar{x}$ and $\bar{y}$ represent the centroid coordinate of the region. For more robustness, we use the first two central moments, namely $\mu_{0,0}$ and $\mu_{1,1}$. Let $L_0 = \mu_{0,0}$ and $L_1 = \mu_{1,1}$ when $X, Y \in \mathcal{R}_L$ and let $R_0 = \mu_{0,0}$ and $R_1 = \mu_{1,1}$ when $X, Y \in \mathcal{R}_R$. Since we are trying to choose one of the six aforementioned candidate orientations, we introduce L_0^C, L_1^C, R_0^C, and R_1^C to indicate the various

moments for each candidate orientation, with $C = 1, 2, 3, 4, 5, 6$. We finally choose one of the candidates C by minimizing the Root Mean Square Error of the central moments:

$$\arg\min_{C} \left(\sqrt{\frac{(L_0^C - L_0)^2 + (L_1^C - L_1)^2 + (R_0^C - R_0)^2 + (R_1^C - R_1)^2}{4}} \right) \quad (7)$$

The central moment procedure allows the algorithm to differentiate between objects that have a 180 degree rotational difference, where plane fitting alone would find they have the same rotation. Fig. 6 shows a visual example of this phenomenom. The drill in Fig. 6(a) and 6(b) is rotated by 180 degrees and the plane fitting process would mistakenly deduce that they have the same rotation. It is clear from the image moments, in Fig. 6(c) and 6(d), however, that the objects' orientations are different. Algorithmically, and as aforementioned, we are essentially comparing $\mathcal{R}_L$ (Fig. 6(e) and 6(g)) and $\mathcal{R}_R$ (Fig. 6(f) and 6(h)).

4.4. *Layer 3: Calculating End-Effector Rotation*

The last layer determines the appropriate orientation in order to grasp the object, and it is activated after the object to be grasped has already been assigned to one of the classes used during training (class M, layer 1), and its orientation has been identified (orientation α_g, layer 2). In order to compute the orientation in this final step, the target pose in $p_g \in \mathbb{R}^3$ is also used. This last stage is complicated by the fact that for every object class positive grasping examples are available only for poses spaced by 30-degree intervals (i.e. for 12 of the 36 views), whereas the orientation has been determined up to 10 degrees. Therefore, before looking for the closest example in the training data concerning class M, it is necessary to *project* p_g to the right place in order to compensate for the possible difference in rotation accuracy. The projected point p_p is then obtained as follows

$$p_p = \mathbf{R}_z(\alpha_t)(p_g - p_m) + p_m \quad (8)$$

where p_m is the mean of the point cloud P_g and $\mathbf{R}_z(\alpha_t)$ encompasses a rotation of α_t about z, with α_t being the smallest rotation that brings α_g to one of the 12 grasping examples. Thanks to this projection, p_p is then aligned with the closest set of examples in the training set. At this point a nearest neighbor search is performed among all the poses p_{Ei} with p_p as target. Let $\mathbf{T}_{Ei}$ be the returned training instance with the pose p_{Ei} closest to p_p. Then, the algorithm returns $\mathbf{R}_{Ei}$, the end-effector rotation corresponding to the closest pose found. However, before returning $\mathbf{R}_{Ei}$ such rotation needs to be *back projected* in order to compensate for the change made in Eq. 8. We return

$$\mathbf{R}_g = \mathbf{R}_z(-\alpha_t)\mathbf{R}_{Ei}. \quad (9)$$

5. Experimental results

5.1. *Experimental setup*

The presented system has been implemented and tested on *George*, the humanoid robotic torso displayed in Fig. 7.

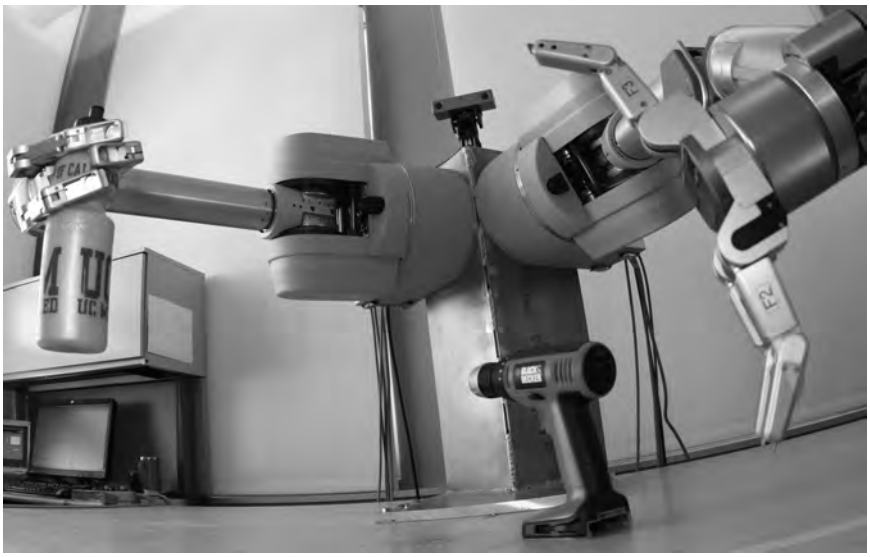

Fig. 7. The experiments presented in this paper have all been implemented and validated on the humanoid robotic torso displayed in this figure.

George is composed of two Barrett arms mounted sideways on a custom made steel frame. Each arm has seven degrees of freedom: three on the shoulder, one on the elbow, and three on the wrist. Moreover, each arm is equipped with a multi-fingered Barrett Hand. The hand has four degrees of freedom, one controlling the spread of the fingers and three controlling the closure of each finger. On top of the frame a BumbleBee 2 stereo camera is mounted on two servos allowing for motions about the yaw and pitch angles. The robot is controlled by an external PC running a real-time Linux distribution. The PC exchanges data with the robot using a CAN bus card. The servos controlling the camera are instead connected to the PC with USB cables and are controlled using a driver provided by the manufacturer (Phydget). A program written in C++ computes direct and inverse kinematics and handles image acquisition with the BumbleBee. The rest of the algorithm is currently implemented in Matlab. All code developed and collected data are freely available on the authors' website.[b] The website also features numerous videos showing the system in action.

[b]See `http://robotics.ucmerced.edu`.

5.2. *Classification Accuracy*

We start by presenting an extensive set of results showing the validity of the proposed SVM classification method using our feature space. In the first experiment, we train our algorithm with all training data except for one, and we try to classify the one that was not included in our training phase. We repeat this process removing and classifying a different view every time, until all views have been classified. Results are shown as a confusion matrix in Table 5.2 and show a classification accuracy of 97.69%.

Table 2. Confusion matrix for the experiment performed on trained objects from the training data.

	Actual						
	Object #	**1**	**2**	**3**	**4**	**5**	**6**
Predicted	**1**	**35**	2	1	0	0	0
	2	1	**34**	0	0	0	0
	3	0	0	**35**	0	1	0
	4	0	0	0	**36**	0	0
	5	0	0	0	0	**35**	0
	6	0	0	0	0	0	**36**

The next batch of experiments is performed to test scale, translation, and rotation invariance. We collected data for the same 6 objects used during training at 30 random locations and orientations. Results are shown in Table 5.2 and outline the scale, translation, and rotation invariance properties incorporated in our algorithm, with an overall accuracy rate of 90.00%.

Table 3. Confusion matrix for the experiment performed on trained objects of different scales, translations, and rotations.

	Actual						
	Object #	**1**	**2**	**3**	**4**	**5**	**6**
Predicted	**1**	**28**	1	0	0	0	0
	2	0	**28**	0	2	1	0
	3	0	0	**26**	0	1	4
	4	0	1	1	**28**	0	0
	5	0	0	3	0	**27**	1
	6	2	0	0	0	1	**25**

Finally, since the focus of this paper is about grasping novel objects, we test the classification layer with objects that are not part of the training set. Novelty in this context can be intended in two ways. The robot may be presented with objects that are similar but different from those used during training (e.g. a different bottle or

a different drill). Alternatively, the robot may be presented with objects that are completely new - objects that have no similarity to those used during training. Fig. 8 shows some of the novel objects we considered and we point the readers to the end of the section for more experiments involving highly different novel objects.

Fig. 8. A DVD case, a drill, a shampoo bottle, a mug, a detergent bottle, a water bottle, a salt container, and a rectangular box were used as novel objects. While discussing the results we will refer to them as object 1 to 8 (with 1 being the leftmost one).

The key observation in grasping novel objects is that different objects can be grasped similarly based on shared geometry with trained objects. Therefore, the SVM classifier should be capable of identifying an appropriate class to extract the grasping information. To perform this test we acquire 30 images of each novel object, varying its position and orientation. The result of this experiment is shown in Table 5.2, which shows an overall classification rate of 84.17%. Even though the objects being classified are different from those used for training, we nevertheless present the results in the form of a confusion matrix, but we need to clarify how correct associations are inferred. For different instances of the same object (e.g. different bottle, or different mug), the correct association is easy to determine. For truly novel object (e.g. DVD case, rectangular box) the association is less obvious. For example, these two objects are classified as a coffee can 63.33% and 33.33% of the time as a drill. The association with the coffee can is somehow natural, since the shape is very similar, and can be considered correct. After having analyzed instances where they are classified as a drill, we verified that because of self-occlusions they feature a long edge parallel to the table, and this is associated with the drill's shaft. In these cases, however, association with the drill still leads to a successful grasp. Indeed, even though our overall classification rate is 84.17%, the grasping rate is better because misclassifying an object will not necessarily result in a poor choice of wrist orientation.

5.3. *Estimation of object rotation*

The next layer to be evaluated concerns the orientation's estimation about the z axis. The method we embrace is somewhat similar to the one presented in the previous subsection. That is, for each object class we remove one instance from the training data and we then determine the closest neighbor in order to estimate the

Table 4. Confusion matrix for the experiment performed on novel objects of different scales, translations, and rotations.

	Actual (Trained)								
	Object #	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Predicted (Novel)	**1**	**19**	2	0	2	0	0	1	**19**
	2	10	**28**	0	2	0	0	0	10
	3	0	0	**27**	0	3	3	0	0
	4	0	0	0	**26**	0	0	**29**	1
	5	0	0	0	0	**27**	0	0	0
	6	1	0	3	0	0	**27**	0	0

orientation. The method is repeated for each object in every class. Before analyzing the results encompassed by the stacked histogram displayed in Fig. 9, we observe that different object symmetries will result in different outcomes. Three different symmetries should be accounted for. Some objects, like the water bottle, are fully symmetric about the z axis. These objects are removed from the current evaluation because the problem is ill posed for them (any rotation would do). Remaining objects can be either partially symmetric (e.g. coffee can, mouth wash bottle, mug) or fully asymmetric (e.g. drill, mug, detergent bottle). For partially symmetric objects, rotations that are 180 degrees apart will result in the same object view, so these cannot be differentiated. Fully asymmetric objects, however, never look the same under different rotations. The situation is in practice somehow more complicated because some objects (e.g. the mug) can be partially symmetric or fully symmetric, depending on whether or not the handle is self-occluded. Keeping this information in mind, Fig. 9 shows a 87.5% rate of finding the closest neighbor (the one within a 10 degree difference) for completely asymmetric objects and a 75% rate of finding the closest neighbor for partially symmetric objects (taking into account that a 180 degree rotation yields the same object view).

5.4. *Overall system performance*

We finally present the end-to-end performance of the system, where we evaluate the success rate for the final grasping action based on the outcome of the three layers. In all experiments, we exploit inverse kinematics to place the end-effector to the pose and orientation determined by the algorithm and we close the fingers. We count any form-closed grasp as being correct, given that this is how the system was trained. In the first experiment, shown in Fig. 10, we place each of the 6 trained objects at 10 random rotations, and let the algorithm determine the right pose and orientation. The overall success rate is 81.66%, where the bottle, drill, and coffee can had the highest (90%) and the mug and mouth wash bottle had the lowest (70%).

We repeat the same experiment with novel objects that are fairly similar to

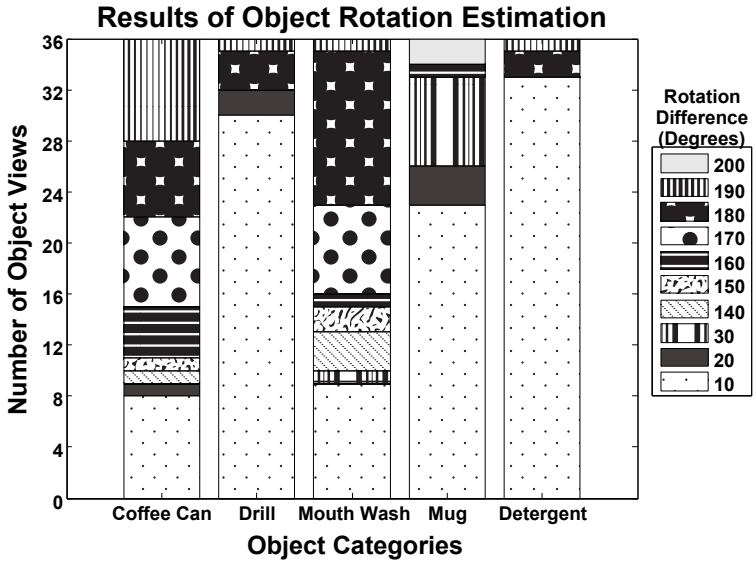

Fig. 9. Stacked histogram showing the results in finding the nearest neighbor, for each object. The patterns indicate how far, in degrees, the closest neighbor is from the actual value (depending on symmetries, it should be 10 to 20 or 160 to 200 degrees for the algorithm to work). The y-axis is a stacked count of the number of trials for which that value occurred.

those we trained on (10 trials for each of the 6 objects). Some example of which are shown in Fig. 11. In this case, the accuracy drops to 76.66%, with the highest accuracy of 100% achieved by the drill and lowest accuracy of 60% for the shampoo bottle and the DVD case.

Finally, we also performed some experiments with objects that are completely different than those we trained on, as seen in Fig. 12. We ran 4 trials for each of the 6 objects and obtained an overall success rate of 83.33%.

Last but not least, as a proof of concept, we ran the same system acting on the left arm rather than the right arm. Examples are shown in Fig. 13, and the success rate is similar.

We conclude the experimental section of the paper with Table 5.4, which shows the speed of the algorithm. We note that, while the algorithm is very fast, it can be made even faster simply by porting the MatLab implementation to C++.

6. Conclusions

This paper has presented a method to determine wrist orientations in order to determine how to grasp an object based on a single image and point cloud provided by a stereo camera. The method is feature-based and complements a recently presented method that determines the pose only. Imitation learning is exploited to extrapolate information from examples provided offline by a human. At run time, less than half

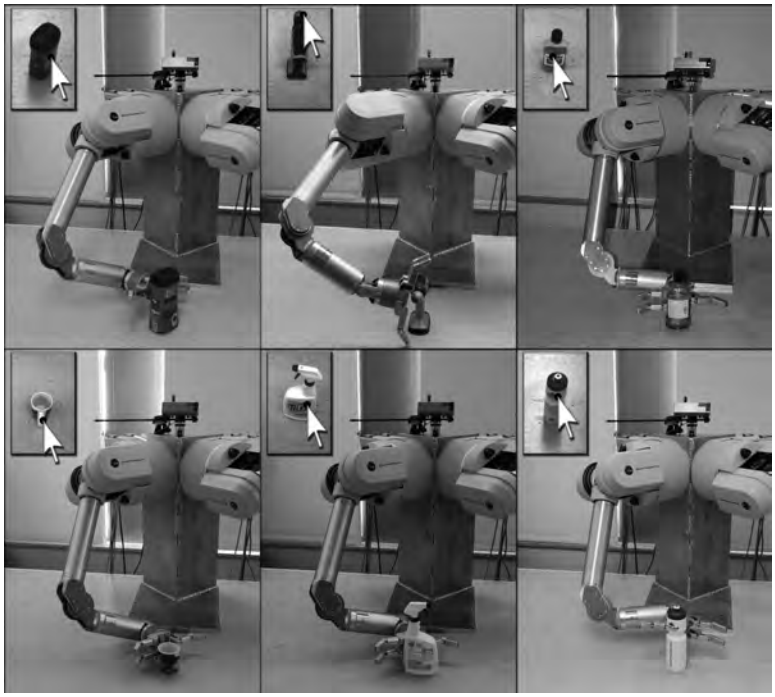

Fig. 10. Robot grasping trained objects with camera view in each left-hand corner. The cursor in each camera view shows p_g in image-space.

Table 5. Algorithm speed, divided by parts.

Algorithmic Part	**Average Time (ms)**
Outlier Removal	361.3
Object Classification	103.1
Nearest Neighbor Search	6.5
Wrist Calculation	4.4
Inverse Kinematics	20.0
Total Time	**495.3**

a second is needed in order to determine where the hand should be placed in order to grasp the object. We have described the three layers composing the algorithm, namely classification, rotation estimation, and wrist orientation computation. The proposed algorithm has been implemented on a humanoid torso and extensive experimental results corroborate the effectiveness of the method presented.

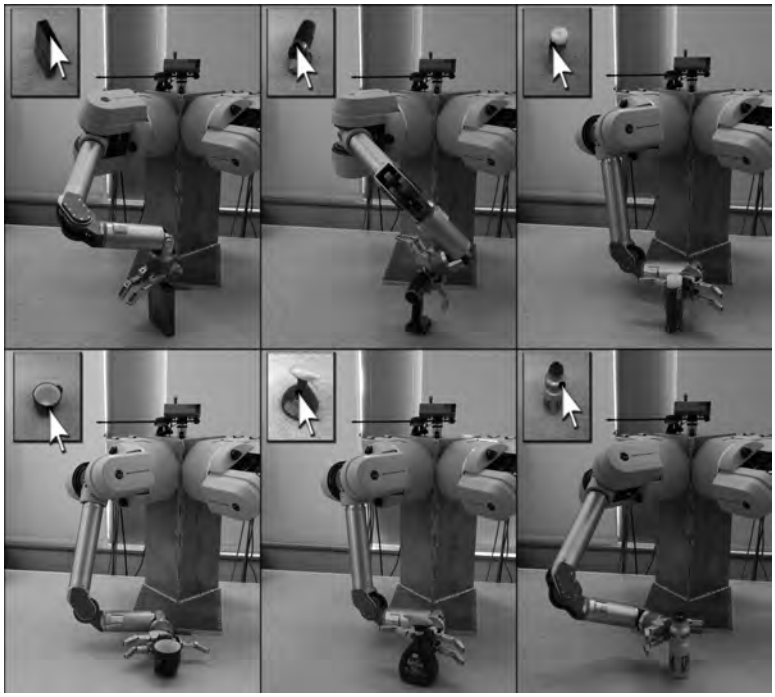

Fig. 11. Robot grasping novel objects with camera view in each left-hand corner. The cursor in each camera view shows p_g in image-space.

Acknowledgments

This paper extends preliminary results presented in a previous publication.[45] This research is partially supported by the National Science Foundation under grant BCS-0821766. Any opinions, findings and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. A. Saxena, J. Driemeyer, and A. Ng, Robotic grasping of novel objects using vision, *International Journal of Robotics Research* **27**(2) (2008) 157–173.
2. D. Lowe, Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision* **60**(2) (2004) 91–110.
3. H. Bay, T. Tuytelaars, and L. Gool, SURF: Speeded up robust features, in *European Conference on Computer Vision* (Springer, Graz, Austria, 2006), pp. 404–417.
4. C. Anderson, B. Axelrod, J. Case, J. Choi, M. Engel, G. Gupta, F. Hecht, J. Hutchison, N. Krishnamurthi, J. Lee, H. Nguyen, R. Roberts, J. Rogers, and A. Trevor, A new mobile manipulation platform for automatic coffee retrieval, in *Workshop on "Robot Manipulation: Sensing and Adapting to the Real World" at RSS* (Electronically Published, Atlanta, USA, 2007).
5. J. Choi, H. Takahashi, Y. Mae, K. Ohara, T. Takubo, and T. Arai, Interoperable

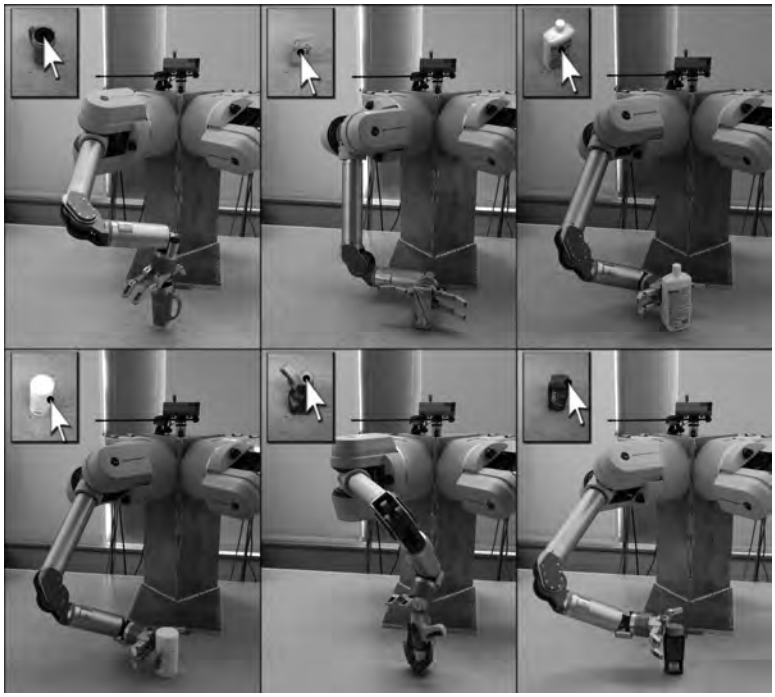

Fig. 12. Robot grasping completely novel objects with camera view in each left-hand corner. The cursor in each camera view shows p_g in image-space.

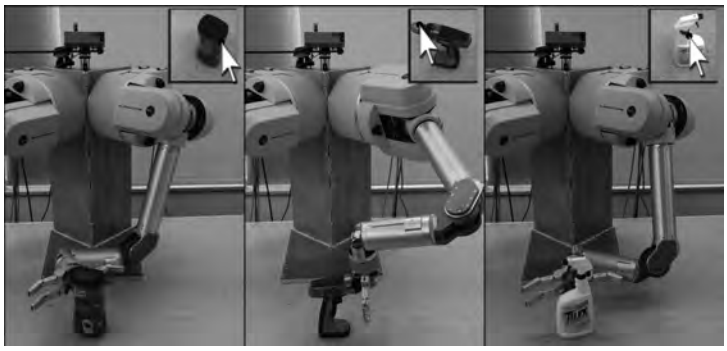

Fig. 13. Left arm grasping objects with camera view in each right-hand corner. The cursor in each camera view shows p_g in image-space.

RT component for object detection and 3D pose estimation for service robots, in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE Press, St. Louis, USA, 2009), pp. 2710–2715.

6. M. Levin, Textured object detection in stereo images using visual features, in *CS 229 Machine Learning Final Projects* (Electronically Published, California, USA, 2008).

7. W. Li and L. Kleeman, Interactive learning of visually symmetric objects, in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE Press, St. Louis, USA, 2009), pp. 4751–4756.
8. M. Munich, P. Pirjanian, E. Bernado, L. Goncalves, N. Karlsson, and D. Lowe, SIFTing through features with ViPR, *IEEE Robotics and Automation Magazine* **13**(3) (2006) 72–77.
9. A. Romea, D. Berenson, S. Srinivasa, and D. Ferguson, Object recognition and full pose registration from a single image for robotic manipulation, in *IEEE International Conference on Robotics and Automation* (IEEE Press, Kobe, Japan, 2009), pp. 48–55.
10. J. Piater, Learning visual features to predict hand orientations, in *Workshop on "Machine Learning of Spatial Knowledge" at ICML* (Electronically Published, Sydney, Australia, 2002).
11. D. Bowers and R. Lumia, Manipulation of unmodeled objects using intelligent grasping schemes, *Transactions on Fuzzy Systems* **11**(3) (2003) 320–330.
12. A. Morales, P. Sanz, and A. del Pobil, Vision-based computation of three-finger grasps on unknown planar objects, in *IEEE/RSJ International Conference on Intelligent Robots and Systems Conference* (IEEE Press, Lausanne, Switzerland, 2002), pp. 1693–1698.
13. A. Morales, P. Sanz, and A. del Pobil, An experiment in constraining vision-based finger contact selection with gripper geometry, in *IEEE/RSJ International Conference on Intelligent Robots and Systems Conference* (IEEE Press, Lausanne, Switzerland, 2002), pp. 1711–1716.
14. A. Morales, P. Sanz, A. del Pobil, and A. Fagg, Vision-based three-finger grasp synthesis constrained by hand geometry, *Robotics and Autonomous Systems* **54**(6) (2006) 496–512.
15. A. Remazeilles, C. Dune, E. Marchand, and C. Leroux, Vision-based grasping of unknown objects to improve disabled people autonomy, in *Workshop on "Manipulation: Intelligence in Human Environments" at RSS* (Electronically Published, Zurich, Switzerland, 2008).
16. M. Jeannerod, The timing of natural prehension movements, *Motor Behavior* **16**(3) (1984) 235–254.
17. O. Kroemer, R. Detry, J. Piater, and J. Peters, Active learning using mean shift optimization for robot grasping, in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE Press, St. Louis, USA, 2009), pp. 2610–2615.
18. F. Worgotter, N. Kruger, N. Pugeault, D. Calow, M. Lappe, K. Pauwels, M. Hulle, S. Tan, and A. Johnston, Early cognitive vision: Using gestalt-laws for task-dependent, active image-processing, *Natural Computing* **3**(3) (2004) 293–321.
19. B. Balaguer and S. Carpin, Efficient grasping of novel objects through dimensionality reduction, in *IEEE International Conference on Robotics and Automation* (IEEE Press, Anchorage, USA, 2010), pp. 1279–1285.
20. A. Miller, S. Knoop, H. Christensen, and P. Allen, Automatic grasp planning using shape primitives, in *IEEE International Conference on Robotics and Automation* (IEEE Press, Taipei, Taiwan, 2003), pp. 1824–1829.
21. C. Goldfeder, M. Ciocarlie, J. Peretzman, H. Dang, and P. Allen, Data-driven grasping with partial sensor data, in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE Press, St. Louis, USA, 2009), pp. 1278–1283.
22. A. Sahbani and S. Khoury, A hybrid approach for grasping 3D objects, in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE Press, St. Louis, USA, 2009), pp. 1272–1277.
23. S. Khoury and A. Sahbani, On computing robust N-finger force-closure grasps of 3D

objects, in *IEEE International Conference on Robotics and Automation* (IEEE Press, Kobe, Japan, 2009), pp. 2480–2486.
24. A. Meltzoff, Infant imitation after a 1-week delay: Long-term memory for novel acts and multiple stimuli, *Developmental Psychology* **24**(4) (1988) 470–476.
25. C. von Hofsten, The structuring of neonatal arm movements, *Child Development* **64**(4) (1993) 1046–1057.
26. M. McCarty, R. Clifton, D. Ashmead, P. Lee, and N. Goubet, How infants use vision for grasping objects, *Child Development* **72**(4) (2001) 973–987.
27. M. Olmos, J. Carranza, and M. Ato, Force-related information and exploratory behavior in infancy, *Infant Behavior and Development* **23**(3) (2000) 407–419.
28. E. Oztop, N. Bradley, and M. Arbib, Infant grasp learning: a computational model, *Experimental Brain Research* **158**(4) (2004) 480–503.
29. M. Goodale, A. Milner, L. Jakobson, and D. Carey, A neurological dissociation between perceiving objects and grasping them, *Nature* **349**(6305) (1991) 154–156.
30. U. Castiello, The neuroscience of grasping, *Nature Reviews Neuroscience* **6**(9) (2005) 726–736.
31. M. Jeannerod, Intersegmental coordination during reaching at natural visual objects, in *Attention and Performance IX*, ed. J. Long and A. Baddeey (Erlbaum, Hillsdale, USA, 1981), pp. 153–168.
32. P. Weir, C. MacKenzie, R. Marteniuk, and S. Cargoe, Is object texture a constraint on human prehension?: Kinematic evidence, *Motor Behavior* **23**(3) (1991) 205–210.
33. J. J. Craig, *Introduction to Robotics – Mechanics and Control* (Prentice Hall, Upper Saddle River, NJ, 2005).
34. R. Rusu, I. Sucan, B. Gerkey, S. Chitta, M. Beetz, and L. Kavraki, Real-time perception-guided motion planning for a personal robot, in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE Press, St. Louis, USA, 2009), pp. 4245–4252.
35. R. Rusu, A. Holzbach, R. Diankov, G. Bradski, and M. Beetz, Perception for mobile manipulation and grasping using active stereo, in *IEEE/RAS International Conference on Humanoid Robots* (IEEE Press, Paris, France, 2009), pp. 632–638.
36. J. Laurikkala, M. Juhola, and E. Kentala, Informal identification of outliers in medical data, in *Workshop on "Intelligent Data Analysis in Medicine and Pharmacology" at ECAI* (Electronically Published, Berlin, Germany, 2000).
37. V. Hodge and J. Austin, A survey of outlier detection methodologies, *Artificial Intelligence Review* **22**(2) (2004) 85–126.
38. C. Bishop, *Pattern analysis and machine learning*, (Springer-Verlag, New York, USA, 2006).
39. A. Karatzoglou, D. Meyer, and K. Hornik, Support vector machines in R, *Journal of Statistical Software* **15**(9) (2006) 1–28.
40. C. Hsu and C. Lin, A comparison of methods for multiclass support vector machines, *Transactions on Neural Networks* **13**(2) (2002) 1045-1052.
41. J. Glover, D. Rus, and N. Roy, Probabilistic models of object geometry for grasp planning, in *Robotics: Science and Systems* (Zurich, Switzerland, 2008).
42. B. Steder, G. Grisetti, M. Van Loock, and W. Burgard, Robust on-line model-based object detection from range images, in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE Press, St. Louis, USA, 2009), pp. 4739–4744.
43. H. Kim and S. Arajo, Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast, in *Pacific-Rim Symposium on Image and Video Technology* (Springer, Santiago, Chile, 2007), pp. 100–113.
44. C. Shakarji, Least-squares fitting algorithms of the NIST algorithm testing system,

Journal of Research of the National Institute of Standards and Technology **103**(6) (1998) pp. 633–641.
45. B. Balaguer and S. Carpin, Learning end-effector orientations for novel object grasping tasks, in *IEEE/RAS International Conference on Humanoid Robots* (IEEE Press, Nashville, USA, 2010), pp. 302–307.