Human-Inspired Grasping of Novel Objects through Imitation Learning

Benjamin Balaguer and Stefano Carpin

***Abstract*— A robotic algorithm capable of grasping novel objects is presented. With a single stereo image as input, a supervised machine learning framework is developed that is both fast and accurate. The algorithm is trained by a human in a learn-by-demonstration procedure where the robot is shown a set of valid end-effector rotations to grasp various objects. Learning is then achieved through a multi-class support vector machine, orthogonal distance regression, and nearest neighbor searches. The presented approach requires little sensor input, generalizes to novel objects, and is demonstrated to work in in real-time an a Barret WAM manipulator.**

I. Introduction

We consider the problem of grasping an unknown object with a multi-fingered robotic hand mounted on an anthropomorphic robot. Recent advances in machine learning techniques have given a new perspective to this challenge. In particular, a paper by Saxena et al. [9] presented a novel machine learning method to identify good grasping points in a picture of a previously unseen object. The method is feature-based: given an unseen object's image, a feature vector is associated with every pixel and then used to classify the pixel as being a good or bad grasping point. More recently, we have demonstrated that this technique can be further accelerated using dimensionality reduction techniques and can be executed in real time [1]. In this contribution, we extend this line of research and consider *plan execution*, after a candidate grasping point has been identified. More precisely, after a pixel in image space is selected as a good grasping point, one is left with the problem of correctly orientating the end-effector to grasp the object at the grasping point. Consequently, the question we answer is the following: given a target grasping point, determine the end-effector pose and orientation in order to grasp the object at the given point. More formally, indicating with T_E the 4×4 transformation matrix specifying pose and orientation of the end effector, the problem is cast as determining a function $f : \mathbb{R}^3 \rightarrow SE(3)$, where $SE(3)$ is the Special Euclidean group of dimension 3. Since we consider the translation component of T_E a solved problem [9], [1], the paper focuses on the 3×3 rotation component of T_E. The rest of the paper is organized as follows. Our proposed algorithm is detailed in Section II. Section III describes the robotic system used to validate our method, and a battery of experiments substantiating its performance. Finally, in section IV, we outline the algorithm's strengths and contributions to the community.

School of Engineering, University of California, Merced, CA, USA, {bbalaguer,scarpin}@ucmerced.edu

II. Learning how to grasp

A. Design choices

Our algorithm is heavily influenced by human grasp strategies, developed in the first two years of infancy, which relies primarily on learning (imitated [6] and reinforced [11]) and human senses (sight [5] and touch [7]). Furthermore, evidence suggests that, for humans, imitated learning is supplemented by sight, whereas reinforcement learning is driven by touch [8]. Following human grasping, and given the fact that we are interested in imitation learning, we abstain from touch sensors, sensor-fusion, and object/model reconstruction and limit our algorithm's sensory input to a single stereo image. Human grasping research has also shown that there exists a dissociation between recognizing objects and grasping them [2]. We also follow this finding in our approach, by separating grasp planning from grasping execution (i.e. we do not use reaction-based grasping or visual-servoing). Our solution to the problem of finding an appropriate end-effector orientation, given a stereo image and a point to grasp in image space, uses a supervised machine learning algorithm. Since we view human grasping strategies as a basis for comparison, we train our algorithm using a kinesthetic approach. We note that off-the-shelf learning algorithms (e.g. Support Vector Machines, Radial Basis Functions, Artificial Neural Networks) are not directly applicable since they all have problems with the many-to-many mappings in continuous space inherent to our problem (i.e. one image pixel maps to many valid wrist orientations, while the same wrist orientation maps to different image pixels). Therefore, we introduce a hierarchical method comprised of an object classification layer, followed by the calculation of the object's rotation, and finalized by a nearest neighbor search in the training data. The hierarchical nature of the approach allows the algorithm to aggressively prune the search space at run time. Furthermore, each layer of the algorithm offers a tradeoff between speed and accuracy. Last but not least, we put an emphasis in generating a manipulator-independent algorithm that is scale, rotation, and translation invariant for the objects.

B. Finding good grasping points

To find a good grasping point in image space, one of the two inputs of our algorithm, we use our own algorithm [1], a modification of [9]. Due to space limitations, we only give a brief overview of the method and direct interested readers to the aforementioned publications. A feature vector, acquired by a set of convolution filters, is associated with each pixel in image space and used to classify it as being a

good or bad grasping points. This binary learning approach allows to use logistic regression, resulting in a very accurate yet efficient algorithm. The authors report remarkable results for objects similar to those in the training set, but also, and more importantly, for novel objects. This is a very desirable property, which we preserve in our algorithm, since it allows for grasping tasks in uncertain and unforeseen scenarios (i.e. scenarios for which we have not trained our robot).

C. Acquiring Training Data

We have attempted to make the training data acquisition, which requires a human-in-the-loop, as automatic as possible. A diverse set of 6 objects shown in Figure 1 are used for training. For each object, we account for rotation-invariance by taking 36 images, with each image representing a different rotation around the axis perpendicular to the plane where they lay. We take the images uniformly around the full 360-degree spectrum, thus having one image every 10 degrees.

Fig. 1. The six objects used to generate training data.

For each object class c, we have a set of stereo images, so the training set is $S_I^c = \left[I_1^c, I_2^c \cdots I_n^c\right]$ where c is an integer between 1 and 6 and $n = 36$. The use of a stereo camera triggers a straightforward conversion from images to point clouds. Therefore S_I^c can be converted into a set of point clouds, $S_P^c = \left[P_1^c, P_2^c \cdots P_n^c\right]$. Each point cloud P_i^c suffers from two drawbacks: it is inherently noisy and includes the surrounding environment (i.e. the table the object is on). We first remove the table in the environment by using orthogonal distance regression [10] to find the best fit plane and removing all of the points within a certain distance from the plane. We then remove additional noise using an outlier detection algorithm based on quartile ranges [4]. The two-step denoising process is fast, requires little human supervision, and works well in practice. From now on, P_i^C refers to the denoised point cloud.

Our learning algorithm relies on a human showing the robot how to grasp some objects, in a kinesthetic framework. This knowledge can then be exploited by the robot to grasp both trained and novel objects (we refer to a trained object as an object used to train the robot and a novel object as an object not used in the training stage). Specifically, we put the manipulator in a gravity-compensated state such that a human can freely interact with it. For each object view the human moves the manipulator to a valid grasp position and records the corresponding robot configuration, q. For each point cloud P_i^c, a set of configurations $q_{i,j}^c$ represents good grasps positions and orientations. The j variable denotes that, for a given image, there exists many valid robot configurations capable of yielding valid grasps throughout the surface of the object. From a practical standpoint, the human moves the manipulator into as many good grasping positions as possible, covering as much object surface area as possible. To reduce the time-consuming aspect of acquiring training data, we only record this information for 12 of the 36 views, uniformly spaced by 30 degrees. Readers are encouraged to view videos of this process on our website[1], where the entire data set used for this paper can also be downloaded.

D. Classifying Objects

We start the hierarchical process by classifying the object to grasp. We approach the problem of classification by exploiting a multi-class SVM [3], since they work well with discrete classes. We are classifying between 6 categories, one for each of our trained objects. We chose to use the one-against-one multi-class SVM method with a polynomial kernel, after empirically determining that it produced better results while being faster (e.g. more accurate and faster than a Gaussian Radial Basis Function kernel).

The SVM requires a constant-sized feature vector across all of the training and input data. Since we want to use point clouds to differentiate between objects, and each point cloud will have a different number of points, we re-factor the original point clouds P_i^c into fixed-sized feature vectors F_i^c. We start by generating a new point cloud centered at the origin, $P_i'^c$, by subtracting the mean from every point in the point cloud, satisfying the translation-invariant algorithm property. We then generate our feature vector by encompassing the object, in image space, into a two-dimensional matrix of fixed sized. Since objects vary in size, the matrix is scaled uniformly to maintain its given size. Each cell in the matrix then corresponds to a number of pixels, each of which being represented by a 3-dimensional Cartesian coordinate in $P_i'^c$. We compose our feature vector, F_i^c, by taking the average Cartesian coordinate of all the pixels within the cell, a process we repeat for each cell. We use a 50×50 grid, producing a feature vector size of 7500 (2500 cells, each comprised of the 3 Cartesian coordinate numbers for X, Y, and Z). This encoding accounts for two important properties. First, each point cloud can be encoded with a same-sized feature vector, a desirable element for SVM. Second, the feature encoding indirectly achieves scale invariance since the grid automatically adjusts to changes in size.

Since subsequent components of the hierarchical method depend on classification accuracy, we now provide results showing the validity of the SVM classification in our feature space. In our first experiment we train the algorithm with all the training data except for one object view, and classify the omitted object. We repeat this process removing and classifying a different view every time, until all views have been classified. We achieved a classification accuracy of 97.69%. The next set of experiments test the scale, translation, and rotation invariance properties of the algorithm. We acquire new data, with each of our 6 objects, at 10 random locations and rotations in front of the robot. The

[1] https://robotics.ucmerced.edu/Robotics/ICRA2011Workshop

multi-class SVM is trained on the full training set (i.e. the 36 views for each of the 6 objects) and the new images (i.e. 10 images for each of the 6 objects) are classified. This is a harder experiment due to the translation changes, different viewpoints from the robot's perspective which create different scales for the objects, and various rotations. The overall accuracy rate lowers down to 88.33%. For the last classification experiment, we test the algorithm's ability to generalize to previously unseen (i.e. untrained) objects. The idea behind novel object grasping stems from the observation that many different objects can be grasped similarly based on shared geometry. For the sake of the experiment, we have chosen novel objects that are fairly similar to the trained objects (see Figure 2), so that a quantitative classification success rate can be exploited. We note that the novel objects are different from the trained objects by their texture, size, or geometry. For the novel object experiment, we acquire a

Fig. 2. The six objects used as novel objects. The row order in which they are presented corresponds to our expected classification in Figure 1.

set of 10 images, for each novel object, placed at random locations and rotations throughout the field of view of the robot. The SVM is trained on all of the training data we have acquired and each image is classified. The experiment shows an overall expected-classification rate of 85%. We emphasize that the classification accuracy presented are based on our prior belief as to how the novel objects should be classified. In some cases, this prior belief is correct (i.e. for the drill, the spray bottle, the water bottle, and the mug) while, in other cases, it is not obvious (i.e. the DVD case matches the coffee can and the shampoo bottle matches the mouth wash bottle). This latter observation is especially evident with the DVD case, which is classified 60% of the time as a coffee can (i.e. our belief, due to the rectangular nature of both objects) and 40% of the time as a drill. While our expected classification rate is 85%, our grasping success rate will be better because misclassifying an object will not necessarily result in a poor choice of wrist orientation. It simply means that a different object will be used to find the best wrist orientation - an object that was deemed better by the algorithm.

E. Calculating Object Orientation

We can now determine the object that best matches the one the robot needs to grasp. We have essentially narrowed down the scope of the problem from a set of objects (i.e. 216 images) to a single object (i.e. 36 images). In the second stage of the algorithm, we determine the correct orientation of the object to grasp using the information in our training data. We came up with a solution involving plane fitting to deduce the most likely object rotation. More specifically, we exploit the point clouds to find the best-fit plane, through orthogonal distance regression [10]. Given the two planes with their normal vectors, N_1 for an object in our training data and N_2 for the object the robot is trying to grasp, we calculate the angle between the two planes, $\theta = \arccos\left(\frac{N_1^T N_2}{||N_1||||N_2||}\right)$. This process is very efficient since the majority of the plane fitting can be done offline for the training data. We use the plane fitting process to find the nearest neighbor, in terms of rotation, in our training data. The drawback with this method is that it cannot differentiate objects that are rotated by 180 degrees, since the best fit planes would be the same. We address this issue with image moments calculated for both the left and right sides of our object image. More specifically, we calculate the first and second degree image moments, for each side, and use the Root Mean Square Error (RMSE) between an object in our training set and the object to grasp. Summarizing from a high-level perspective, we determine the object rotation as follows. First, find the angles between the best-fit planes of all our object training data and the one we are trying to grasp. Second, find the 6 training objects with the lowest angles and run the RMSE image moments calculations on all 6. Out of the 6 objects, we choose the one with the lowest RMSE to be the closest in terms of rotation to the one we are trying to grasp.

We again run an experiment to determine the validity of this algorithmic stage before moving to the next one. For each object class we remove one object from our training data and try to find its closest neighbor as per the aforementioned technique. We repeat this process removing a different object for each class until all of the views have been processed. We note that different object symmetries will result in different outcomes. There are three possible cases of symmetry to take into account. First, some objects will be fully symmetric (e.g. water bottle), where object rotations around the axis perpendicular to the table will not alter the object view. For such objects, the experiment is meaningless since any rotations would be dealt the same way by the manipulator. For this reason, we do not include the water bottle as part of the experiment. Second, some objects will be partially symmetric (e.g. coffee can, mouth wash bottle, mug), where only rotations that are 180 degrees apart will result in the same object view. In that case, we cannot differentiate between rotations that are 180 degrees apart and can equally use either. Third, some objects will be completely asymmetric (e.g. drill, mug, spray bottle), where they never look the same under different rotations and we need to find the closest match in our training set. Our results indicate a 87.5% rate of finding the closest neighbor for completely asymmetric objects and a 75% rate of finding the closest neighbor for partially symmetric objects. We note that finding the closest neighbor is identical to finding the closest object rotation, since our training data is labeled.

F. Calculating End-Effector Rotation

At this point we have found both the closest object class and its rotation in our training data. We now need to determine the correct end-effector orientation using the input pixel location in image space. First, we convert the

pixel location into a Cartesian coordinate, thanks to the stereo camera, and offset it by the point cloud's mean M to yield a point L. We then rotate the point, using the rotation matrix R_z, by the angle calculated in the previous step. This process generates a point, $L' = R_z(L - M) + M$ that is aligned with the closest object in our training set with grasp examples. After converting all of the trained configurations to Cartesian coordinates, using forward kinematics, we use a neighbor search in 3 dimensions to find the closest match to our input pixel. This closest match yields the rotation matrix, R_{nn}, for the end-effector by using forward kinematics on the closest configuration. Last but not least, we need to rotate our end-effector to match the original object, straightforwardly achieved by $R_e = R'_z \times R_{nn}$, where R'_z is the rotation matrix rotating in the inverse direction of R_z and by the same degree amount. The resulting end-effector rotation R_e can be used, with the input pixel location converted to Cartesian coordinates, as input to an inverse kinematics solver to get the manipulator to grasp the object.

III. Experimental results

The robotic platform used to evaluate the proposed algorithm is shown in Figure 3. *George* is a humanoid robotic torso composed of two anthropomorphic Barrett arms. Each arm is equipped with a Barrett Hand, and the torso is completed by a BumbleBee stereo camera. The software computing kinematics and image processing has been developed in-house and is written in C++. The rest of the algorithm is implemented in MatLab. We have performed a large amount of experiments, which we briefly highlight in this section through accuracy measures and representative pictures. We encourage readers to visit our aforementioned website for videos showing the experiments running on our robot in real time. For all the experiments presented herein, we simply close the fingers of the hand to grasp the object and count any form-closed grasp as being correct. In our first experiment, we place each of our 6 trained objects at 10 random rotations, manually input a pixel location, and let the algorithm try to grasp the object. 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 our first experiment (i.e. 6 objects, each with 10 trials) with novel objects that are fairly similar to those we trained on, some example of which are shown in Figure 3. Under those conditions, 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. We conclude the experimental section of the paper by mentioning that the algorithm runs in less than 0.5 second.

IV. Conclusions

We have presented a novel algorithm, influenced by human grasp research and imitation learning, aimed at computing wrist orientations for feature-based grasping algorithms. In addition to speed, one of the most desirable properties of the algorithm is that it is manipulator-independent. Training data acquired with one manipulator can be transferred to another. Last but not least, the algorithm is capable of grasping novel objects and unseen object parts, usually due to self-occlusions, and works with a single image taken from a stereo camera.

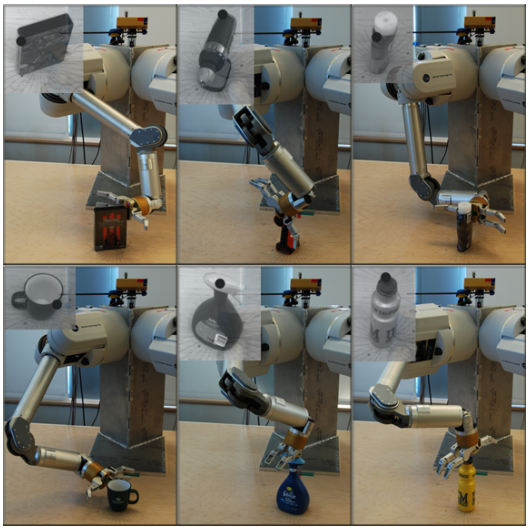

Fig. 3. Robot grasping novel objects with camera view in each corner.

Acknowledgments

This work is supported by the NSF grant BCS-0821766. An earlier version of this work appeared at the IEEE/RSJ 2010 Humanoid conference.

References

[1] B. Balaguer and S. Carpin. Efficient grasping of novel objects through dimensionality reduction. In *IEEE International Conference on Robotics and Automation*, pages 1279–1285, 2010.

[2] M. Goodale, A. Milner, L. Jakobson, and D. Carey. A neurological dissociation between perceiving objects and grasping them. *Nature*, 349:154–156, 1991.

[3] A. Karatzoglou, D. Meyer, and K. Hornik. Support vector machines in r. *Journal of Statistical Software*, 15(9):1–28, 2006.

[4] J. Laurikkala, M. Juhola, and E. Kentala. Informal identification of outliers in medical data. In *Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology IDAMAP*, 2000.

[5] M. McCarty, R. Clifton, D. Ashmead, P. Lee, and N. Goubet. How infants use vision for grasping objects. *Child Development*, 72(4):973–987, 2001.

[6] A. Meltzoff. Infant imitation after a 1-week delay: Long-term memory for novel acts and multiple stimuli. *Developmental Psychology*, 24(4):470–476, 1988.

[7] M. Olmos, J. Carranza, and M. Ato. Force-related information and exploratory behavior in infancy. *Infant Behavior and Development*, 23(3):407–419, 2000.

[8] E. Oztop, N. Bradley, and M. Arbib. Infant grasp learning: a computational model. *Experimental Brain Research*, 158:480–503, 2004.

[9] A. Saxena, J. Driemeyer, and A. Ng. Robotic grasping of novel objects using vision. *International Journal of Robotics Research*, 27(2):157–173, 2008.

[10] 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):633–641, 1998.

[11] C. von Hofsten. The structuring of neonatal arm movements. *Child Development*, 64(4):1046–1057, 1993.