Leveraging Autonomous Segmentation and Grasp Calculation for Programming by Demonstration

David Kent*, Ung Hee Lee‡, Sarah Elliott† and Russell Toris†

***Abstract*— We present an Autonomous Grasping Pipeline that uses object segmentation and grasp calculation to make grasping more robust in robots programmed by demonstration. The object segmentation was performed as a pre-processing step before grasping. We leveraged a model-free approach which enables segmenting unknown objects in a scene without prior knowledge. We applied the existing work of Richtsfeld [1], [2] by dividing the point cloud into small surface patches and using Support Vector Machines to evaluate the relationships between patches and form objects. The segmented 3D data can then be passed to an autonomous grasping algorithm. We use a novel hybrid of model-based and model-free approaches to generate grasps. We learn a model for grasp preferences over context provided by object features, formulated using pairwise ranking. We show that this pairwise grasp preference model yields statistically significant grasp rate improvements over contemporary methods for autonomous grasp selection. We then incorporate the object segmentation and autonomous grasping into a Programming by Demonstration system.**

I. INTRODUCTION & RELATED WORK

One challenge of robot Programming by Demonstration (PbD) is ensuring that the programs are flexible enough to succeed in new situations. Programming robot manipulation by demonstration is difficult since grasping objects requires very precise movements. Small changes in gripper position during a grasp can mean the difference between a successful and a failed grasping attempt. If a user demonstrates a grasp on an object in a certain pose, and then in the future the object's pose changes, even slightly, the grasp pose must be transformed to match. This requires precise object pose detection, as the grasp transformation must be calculated very accurately to avoid grasping failures. Clearly, the method of detecting the object pose and transforming the grasp pose is not the most effective way to program grasps. Rather than attempting to very accurately estimate the pose of objects over time, we propose an Autonomous Grasping Pipeline that leverages autonomous object segmentation and grasp calculation systems to make grasping in PbD programs more robust.

A. Object Segmentation

Object segmentation methodologies can be classified into two categories: model-based approaches and model-free approaches [3]. Model-based approaches have a priori knowledge of objects, which match extracted features from existing object databases [3]–[11]. These approaches are limited due to the lack of reliable feature correspondences when introduced to highly cluttered scenes. Moreover, this method is restricted to a number of objects learned from a database. On the other hand, model-free approaches apply rules or heuristics to segment a scene into distinct regions [4], [11]. Compared to model-based approach, these methods are not restricted to an object dataset, thus segmentation can be generalize to various unknown objects.

B. Autonomous Grasping

As with object segmentation, autonomous grasping approaches can be divided into model-based and model-free approaches. Typical model-based approaches leverage object model databases, constructed from 3D sensor data [12]–[15], shape databases [16], [17], or from CAD models [18], as prior information for grasping. Given an object model, grasps can be precomputed with simulators [19], [20], learned autonomously from trial-and-error [21], or learned from human demonstrators [15], [17]. With an appropriate database of models and associated grasps, grasp planning is reduced to either object recognition with 6-DoF pose estimation or shape matching. These approaches work well at performing specific grasping approaches for individual objects, but they cannot generalize to new objects, for which adding new models can be cumbersome.

Autonomous grasp calculators represent the model-free approaches. Methods vary from sampling over depth data to locate hand-engineered grasp features [22]–[25], ranking sampled grasp poses with human-inspired heuristics [26], [27], discovering grasp features through physical trial-and-error [28], or learning reliable grasp features on adversarial synthetic data [29]. These approaches can generalize to novel objects, and as such can be easily deployed in new environments, but they fail to grasp outlier objects for which the predefined grasp features, heuristics, or learned features fail. In this work we use a novel hybrid of model-based and model-free approaches, by learning a model for grasp preferences over context provided by object features. We formulate this preference model using pairwise ranking. Incorporating object context allows our approach to adapt its ranking strategy for outlier objects that cause failures in model-free approaches, while the underlying pairwise ranking model provides generalization to objects on which the approach was not trained.

C. Ranking

Ranking problems found in information retrieval provide a useful analogy for grasp ranking. We focus on the object

*Georgia Institute of Technology dekent@gatech.edu
‡University of Michigan, Ann Arbor unghee@umich.edu
†Fetch Robotics, Inc. selliott@fetchrobotics.com, rtoris@fetchrobotics.com

ranking problem, a class of preference learning [30], in which given a search query, we seek to determine an ordering of returned items, learned from a few exemplary pairwise preferences provided as training data. Common object ranking problems include web page ranking [31], [32], document retrieval [33]–[35], image retrieval [36], and recommender systems [34], [37]. Beyond information retrieval, object ranking has been applied to scheduling [38], where the query is a set of observations over current tasks, and the items are the set of scheduling decisions at a current time step. We take a similar approach to grasp ranking, with the object itself as the query, and the items are a set of candidate grasps produced by antipodal grasp sampling. This model has advantages in reducing the amount of demonstrations required to obtain adequate training data, while also reducing the demonstration process to simply choosing a good grasp from a set of grasp candidates.

D. Programming by Demonstration

Programming by Demonstration (PbD) is a method of robot programming where the user demonstrates a behavior and the robot uses the demonstration to create a program. Elliott *et al.* developed a system called Programming by Demonstration and Adaptation (PbD+A) that allows the user to adapt their demonstration with input given through a GUI [39]. This system used a very simple perception system, and interaction with objects (including grasping) had to be generated by hand through demonstrations and complex GUI manipulation. We improve the PbD+A system by incorporating the perception and grasp calculation systems described in this work.

II. APPROACH

Our pipeline begins with model-free object segmentation that is specifically robust to handling cluttered environments with novel objects. The segmented point clouds then serve as input to our novel autonomous grasp calculator, which can generate and select grasps based on grasp preferences learned from the user. We incorporate the segmentation and grasp calculation pipeline into the demonstration, adaptation, and reproduction modalities of the PbD+A GUI. Further, any data collection required to train or finetune the autonomous grasp calculator can be performed using the interface, resulting in an end-to-end PbD system that incorporates autonomous grasping to increase manipulation robustness.

A. Segmentation

Access to accurate sensing devices (e.g. Microsoft Kinect) has advanced the state of the art in robot perception. Despite the recent progress, grasping unknown objects has been a challenging problem in robot grasping tasks. In particular, the problem becomes arduous in real-world scenarios where a large number of objects are cluttered and occluded. Thus, object segmentation is a key step in enabling autonomous grasping and manipulation.

For the perception section of our Autonomous Grasping Pipeline, we used a method developed by A. Richtsfeld [1], which segments unknown objects in RGB-D images. With this method, the point cloud data is pre-segmented into patches using surface normals. The patch is represented either by a plane or non-uniform rational B-splines depending on the shape of the patch. The next step is to determine which patches belong to the same objects by calculating relations between patches. These relations are represented by feature vectors that include features such as mean curvature, color, and distance between patches. A graph is constructed with the surface patches as nodes and the relations between patches as edges. The edges have weights learned from the relation feature vectors by Support Vector Machines (SVM). Then a graph cut algorithm is used to decide globally which sets of surface patches should be grouped together to form objects. For a detailed description, refer to [1].

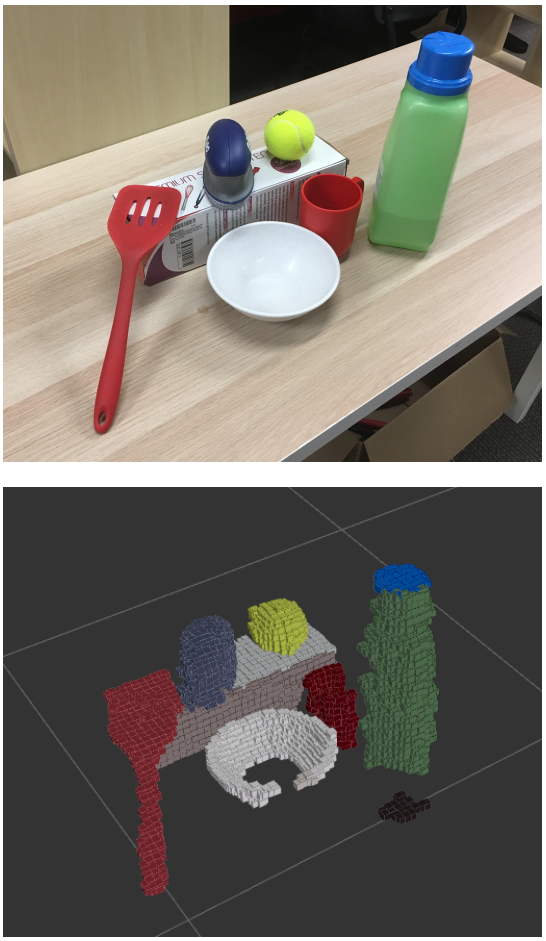

Fig. 1: Cluttered object segmentation. (Top) RGB image of objects in color. (Bottom) segmented point cloud data. Different colors represent individual segmented objects.

This approach was chosen because it performs generalizable scene segmentation in cluttered environments, which are similar to our target environments for robot grasping. It generalizes segmentation by using the shape properties of objects instead of using model information, and as such it generalizes to novel objects. However, this approach has limitations when occlusions are introduced in the scene.

Thus, we extended this approach by training on the Modified Object Segmentation Database (MOSD) [40] which accounts for occlusion. MOSD is a revised version of the Object Segmentation Database (OSD) [2]. It differs from OSD in a way such that when an object is severely occluded, it considers the object as separate disjoint segments, since model-free algorithms cannot be expected to know when significantly disjoint segments belong to the same object [40]. LIBSVM [41] is used to train a SVM.

Figure 1 shows a segmentation result using the above approach in a cluttered scene. Different colors represent different objects as classified by the segmenter.

The segmenter was developed using the Robot Operating System (ROS) [42] and connected with the grasping system of our Autonomous Grasping Pipeline described below. Testing as part of that pipeline demonstrated that the segmenter can seamlessly work with existing grasping systems. The segmenter also includes functionality for training the SVM using an arbitrary data set.

B. Grasp Calculation

In keeping with autonomous grasp calculation, the goal of our grasp calculation algorithm is to calculate and rank a set of grasps by their likely success rate from input depth data. Our implementation extends the point-and-click approach from [27] for pick-and-place, and ranks grasps with a novel pairwise ranking formulation. The algorithm has three steps: candidate grasp sampling, grasp metric calculation, and pairwise grasp ranking under context derived from the object-of-interest. With the addition of the pairwise ranking approach, the algorithm adapts its ranking strategy to different types of objects given a small amount of grasp preference training data. As input, the algorithm takes two point clouds: an object point cloud pc_o provided by our segmentation model, and an environment point cloud pc_e constructed by cropping the full scene to a padded bounding box containing pc_o.

1) Candidate Grasp Sampling: Our system is designed for use with any algorithm that can analyze a point cloud to produce a set of candidate grasps. In this work, we use the antipodal grasp sampler included in the open source `agile_grasp` [23] package. Candidate grasp poses are sampled over pc_e to eliminate grasps in collision with the environment.

2) Grasp Metric Calculation: Extending methods from the point-and-click approach, we developed a broad set of metrics to describe the relationship of a candidate grasp to an object-of-interest and the surrounding environment. The metrics are as follows:

- m^1: difference in orientation between the grasp approach vector and the normal vector to the dominant plane, fit over pc_e. This metric represents grasps perpendicular to large planes in the environment, such as table surfaces or shelf walls.
- m^2: difference in orientation between the grasp approach vector and the normal to a plane fit over a local region of pc_o centered at the grasp point. This represents grasps perpendicular to the object-of-interest.

Fig. 2: (Left) Trained object set for classifier evaluation, model training, and grasp rate evaluation. (Right) Novel object set for generalization experiment.

- m^3: difference in orientation between the grasp orientation and the principal directions of pc_o computed by principal component analysis, i.e. grasps aligned with the object-of-interest.
- m^4: distance from the grasp point to the center of pc_o, approximating grasps near the object center of mass.
- m^5: distance from the grasp point to the nearest point in pc_o. This mainly serves to deter grasps erroneously sampled from the environment rather than the object, but also differentiates grasps centered on an object point from grasps that are offset from the object.

3) Pairwise Grasp Ranking: Taking the analogous ranking problem of information retrieval, we formulated a pairwise ranking problem by equating the search query to the object-of-interest, and the returned items to the calculated grasps. For a pair of grasps g_i and g_j, represented by grasp feature vectors $\mathbf{x}_i = [m_i^1, m_i^2, m_i^3, m_i^4, m_i^5]$ and $\mathbf{x}_j = [m_j^1, m_j^2, m_j^3, m_j^4, m_j^5]$, we define a pairwise grasp feature vector as the difference of the individual grasp feature vectors:

$$\hat{\mathbf{x}}_{ij} = \mathbf{x}_i - \mathbf{x}_j \,. \tag{1}$$

We append the result to a vector of object features $[f_0, f_1, \ldots, f_n]$ calculated from the object point cloud pc_o, resulting in an object context-enhanced pairwise feature vector

$$\mathbf{x}_{ij} = [f_0, f_1, \ldots, f_n, \hat{\mathbf{x}}_{ij}] \,. \tag{3}$$

In this work, we use $\mathbf{x}_{ij} = [l, a, b, x, y, z, \hat{\mathbf{x}}_{ij}]$ as our context-enhanced pairwise feature vector, where $[l, a, b]$ is the average color of pc_o in the CIELAB color space, and $[x, y, z]$ are the dimensions of the minimum-area bounding box containing pc_o. We chose these features as a simple feature vector that can differentiate the set of objects used in our experiments, but we note that more specific features, such as color histograms, point feature histograms, etc., or even object labels, could be used instead.

We create a binary classification problem by adding a label y_{ij} that denotes the ordering of $\mathbf{g}_i$ and $\mathbf{g}_j$:

$$(\mathbf{x}_{ij}, y_{ij}) \text{ where } y_{ij} = \begin{cases} 1 & \text{if } g_i \prec g_j \\ 0 & \text{if } g_j \prec g_i \end{cases} , \tag{4}$$

Algorithm 1 Pairwise Ranking Algorithm

Require: pc_o, pc_e
$grasps \leftarrow$ sampleAntipodalGrasps(pc_e)
$\mathbf{x} \leftarrow$ calculateMetrics($grasps$, pc_o, pc_e)
$\mathbf{f} \leftarrow$ calculateFeatures(pc_o)
for all g_i, g_j in $grasps \mid i \neq j$ **do**
  if classify($[\mathbf{f}, \mathbf{x}_i - \mathbf{x}_j]$) = 1 **then**
    incrementRank(g_i)
  else
    incrementRank(g_j)
  end if
end for
return sort($grasps$)

Algorithm 2 Pairwise Training Data Generation

Require: $grasps$, pc_o, pc_e
$s \leftarrow$ userSelectedGraspIndex($grasps$)
$\mathbf{x} \leftarrow$ calculateMetrics($grasps$, pc_o, pc_e)
$\mathbf{f} \leftarrow$ calculateFeatures(pc_o)
for $i = 0 :$ size($grasps$) $\mid i \neq s$ **do**
  saveTrainingInstance($[\mathbf{f}, \mathbf{x}_s - \mathbf{x}_i]$, 1)
  saveTrainingInstance($[\mathbf{f}, \mathbf{x}_i - \mathbf{x}_s]$, 0)
end for

i.e. a label of 1 denotes g_i should be preferred over g_j, and a label of 0 denotes g_j should be preferred over g_i. Given training pairs $(\mathbf{x}_{ij}, y_{ij})$ we train a binary classifier to predict y_{ij}, thus predicting a grasp ordering.

We collected a set of training data over the training object set shown in Figure 2, by having an expert user select four grasps per object using the interface, as described in the next section. We trained a random forest binary classifier over the training data set using `scikit-learn`[1], which we used for all of our evaluation. Given predictions from the model, the final grasp ranking is determined by voting. The complete algorithm is shown in Algorithm 1.

4) Data Collection: Training data collection does not require any grasp demonstration, as users instead perform selection over a list of autonomously calculated grasps. Since pairwise training data requires only relative differences in grasp quality, training examples can be collected by having the user simply click on the best grasp from the poses, without providing a score or any additional ranking information. Further, a single grasp selection generates $2(n-1)$ pairwise instances from a single click, where n is the total number of grasps, shown in Algorithm 2.

We implemented training data collection within our Programming by Demonstration interface described below and shown in Figure 3.

C. Programming by Demonstration Interface

We improve the PbD+A system from [39] by incorporating the perception and grasp calculation systems described above. The original perception system used by PbD+A was very simple. It used Euclidean Cluster Extraction from the `Point Cloud Library`[2] for segmentation and required that the objects not be touching. The new perception system allows the robot to interact with objects in cluttered environments like the one shown in Figures 1 and 3. Grasping objects is one of the more difficult tasks to program by demonstration. If you demonstrate a grasp on an object and then the object's pose changes, the PbD+A system will try to transform the grasp pose accordingly. This relies on being able to very accurately detect the object's new pose, which potentially requires detailed object modeling or large amounts of training data, and can be difficult due to occlusions. Small changes in gripper position can mean the difference between successful and failed grasps, so this system relies very heavily on the object pose estimation. As such it is not a practical solution for effectively programming grasps.

To address this issue, we incorporate our model-free object segmentation method and our autonomous grasp calculation method. To grasp an object, the user right-clicks the object in the GUI and selects the option to Generate Grasps, 3 (Top). This uses our method to generate a selection of possible grasps for the object. These are displayed as green arrows in the Grasp Selection Window in the UI, 3 (Bottom). The Grasp Selection Window is a window in the corner of the regular PbD+A GUI that zooms in on the selected object and displays the grasp suggestions as arrows. The shaft of the arrow shows the direction of the axis of rotation the gripper's wrist. The arrowhead points to the part of the object that will be grasped. The user can click on an arrow to select a grasp and a gripper marker will appear in the main GUI window to show the position of the gripper during that grasp. This allows the user to quickly view all of the generated grasps and select an appropriate grasp from many options. Most importantly, the suggestions are based on the geometry of the object, so they do not rely on accurately detecting the actual pose of the object. Each time a user executes a selected grasp, the data is collected by the grasp ranking algorithm described in II-B.4 and used to learn improved grasp rankings, which are used autonomously when reproducing the learned skills.

III. Initial Results

We initially tested the grasp ranking algorithm against baseline and state-of-the-art grasp calculation and ranking methods, to test both its performance on the training set objects, and its generalization to novel objects. Our methods for comparison include:

- *Random*: Select a random antipodal grasp from the list of candidates
- *AGILE*: Select a grasp returned by the full AGILE pipeline [23], which involves grasp quality classification by SVM of HOG feature grasp image encodings
- *Pointwise*: Select the highest-ranked grasp using a linear combination of our grasp metrics, similar to the point-

[1] `http://scikit-learn.org`

[2] `http://pointclouds.org`

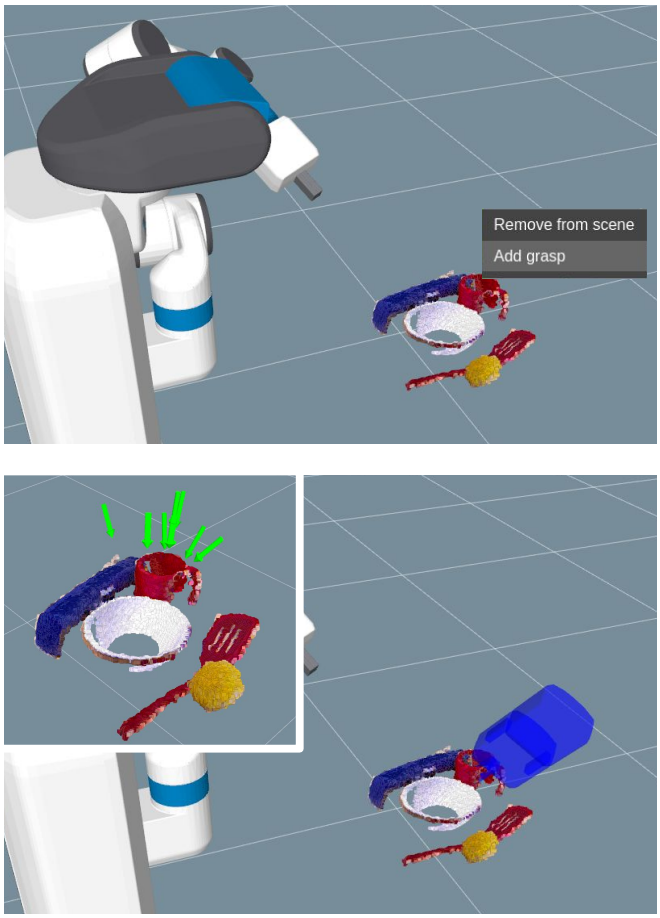

Fig. 3: Grasp calculation and preference data selection. (Top) Selecting a segmented object to grasp in the GUI. (Bottom) Grasp Selection Window with green arrows shown in the corner of the main window. The main window shows the currently selected grasp.

and-click method of [27]

- *Pairwise*: Select the highest-ranked grasp using our novel pairwise ranking algorithm

Each method uses the same antipodal grasp calculator to populate the initial set of grasp candidates, so that our comparison can focus solely on the grasp ranking algorithm. For each object, we had a fetch robot perform a controlled grasping experiment by grasping the object at 16 preselected positions and orientations, for a total of 1600 grasps (16 poses * 25 objects).

Grasp success rates for each method over the trained object set, the novel object set, and a difficult object set[3] are shown in Figure 4. We analyzed the results with a one-way repeated measures Analysis of Variance (ANOVA) for each object set, where grasp rates measured on the same object with different methods were correlated. ANOVA showed significant effects of grasp ranking method on grasp success rate in all three

[3]We defined the difficult object set as all objects for which random antipodal grasp selection had a failure rate of 50% or greater, which included the mug, food can, game controller, tongs, squirt bottle, hammer, laundry detergent bottle, scrub brush, and cooking utensil box.

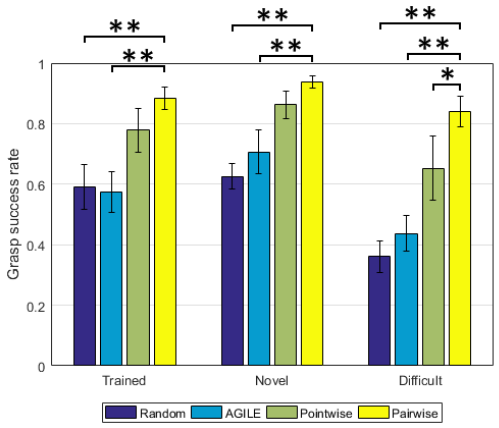

Fig. 4: Grasp rates showing statistically significant differences for the Pairwise method shown (* denotes $p < 0.05$, ** denotes $p < 0.01$). Error bars denote ± 1 standard error. Not shown for readability: the Pointwise method is a statistically significant improvement over the Random and AGILE methods for the trained ($p < 0.05$ and $p < 0.01$, respectively), novel ($p < 0.01$ and $p < 0.01$), and difficult ($p < 0.01$ and $p < 0.01$) object sets.

object sets ($F(3, 42) = 14.08, p < 0.0001$, $F(3, 27) = 25.44, p < 0.0001$, and $F(3, 24) = 24.6, p < 0.0001$, respectively). Tukey's HSD post tests showed that Pairwise significantly outperformed the Random baseline and AGILE method for both the trained ($p < 0.01$, $p < 0.01$) and novel ($p < 0.01$, $p < 0.01$) object sets, showing that, despite requiring some training, the pairwise method can generalize to novel objects and maintain its higher grasp success rate. The pairwise method's advantages are most pronounced for the difficult object set, where more object-specific grasping strategies were necessary for success. Tukey's HSD post tests showed a significant improvement of Pairwise over all of the methods ($p < 0.01$, $p < 0.01$, $p < 0.05$ for Random, Agile, and Pointwise, respectively), reflecting the Pairwise method's ability to effectively adapt its approach to difficult objects.

IV. Discussion and Future Work

We have shown an end-to-end pipeline that incorporates autonomous segmentation and grasp calculation into a Programming by Demonstration interface. The pipeline includes object segmentation robust to clutter, which acts as input to a novel autonomous grasp calculation method. Grasp calculation makes use of pairwise ranking, which, in keeping with the advantages of programming by demonstration, leverages easy-to-provide user input from the interface to improve grasping performance on difficult-to-grasp objects.

Our initial results show that our pairwise grasp ranking method outperforms contemporary approaches, while generalizing to novel objects with a minimal amount of training data. We would next like to perform a user study evaluating the full end-to-end system, to determine whether and to what degree replacing the grasp demonstration step of pick-and-place programming by demonstration with robust

segmentation and autonomous grasp calculation improves the ease of the demonstration process and the effectiveness of reproducing skills from those demonstrations.

REFERENCES

[1] A. Richtsfeld, T. Mrwald, J. Prankl, M. Zillich, and M. Vincze, "Segmentation of unknown objects in indoor environments," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 4791–4796.

[2] A. Richtsfeld, "The object segmentation database (osd),," http://www.acin.tuwien.ac.at/?id=289, 2012.

[3] J. Bohg and D. Kragic, "Learning grasping points with shape context," *Robotics and Autonomous Systems*, vol. 58, no. 4, pp. 362 – 377, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889009001699

[4] M. Fenzi, R. Dragon, L. Leal-Taixé, B. Rosenhahn, and J. Ostermann, "3d object recognition and pose estimation for multiple objects using multi-prioritized ransac and model updating," in *Pattern Recognition*, A. Pinz, T. Pock, H. Bischof, and F. Leberl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 123–133.

[5] X. Li and I. Guskov, "3d object recognition from range images using pyramid matching," in *2007 IEEE 11th International Conference on Computer Vision*, Oct 2007, pp. 1–6.

[6] J. Kuehnle, A. Verl, Z. Xue, S. Ruehl, J. M. Zoellner, R. Dillmann, T. Grundmann, R. Eidenberger, and R. D. Zoellner, "6d object localization and obstacle detection for collision-free manipulation with a mobile service robot," in *2009 International Conference on Advanced Robotics*, June 2009, pp. 1–6.

[7] M. Sun, G. Bradski, B.-X. Xu, and S. Savarese, "Depth-encoded hough voting for joint object detection and shape recovery," in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 658–671.

[8] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 2155–2162.

[9] A. Collet, M. Martinez, and S. S. Srinivasa, "The moped framework: Object recognition and pose estimation for manipulation," *The International Journal of Robotics Research*, vol. 30, no. 10, pp. 1284–1306, 2011. [Online]. Available: https://doi.org/10.1177/0278364911401765

[10] U. Asif, M. Bennamoun, and F. Sohel, "Real-time pose estimation of rigid objects using rgb-d imagery," in *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, June 2013, pp. 1692–1699.

[11] ——, "A model-free approach for the segmentation of unknown objects," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 4914–4921.

[12] Z. Xue, A. Kasper, J. M. Zoellner, and R. Dillmann, "An automatic grasp planning system for service robots," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, 2009, pp. 1–6.

[13] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 1817–1824.

[14] A. Kasper, Z. Xue, and R. Dillmann, "The kit object models database: An object model database for object recognition, localization and manipulation in service robotics," *The International Journal of Robotics Research*, vol. 31, no. 8, pp. 927–934, 2012.

[15] D. Kent, M. Behrooz, and S. Chernova, "Construction of a 3d object recognition and manipulation database from grasp demonstrations," *Autonomous Robots*, vol. 40, no. 1, pp. 175–192, 2016.

[16] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The columbia grasp database," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 2009, pp. 1710–1716.

[17] J. Aleotti and S. Caselli, "Part-based robot grasp planning from human demonstration," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 4554–4560.

[18] A. Morales, T. Asfour, P. Azad, S. Knoop, and R. Dillmann, "Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006, pp. 5663–5668.

[19] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[20] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, vol. 79, 2008.

[21] J. Oberlin and S. Tellex, "Autonomously acquiring instance-based object models from experience," in *Robotics Research*. Springer, 2018, pp. 73–90.

[22] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from rgbd images: Learning using a new rectangle representation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 3304–3311.

[23] A. ten Pas and R. Platt, "Using geometry to detect grasp poses in 3d point clouds," in *Intl Symp. on Robotics Research*, 2015.

[24] D. Fischinger, A. Weiss, and M. Vincze, "Learning grasps with topographic features," *The International Journal of Robotics Research*, vol. 34, no. 9, pp. 1167–1194, 2015.

[25] A. Ten Pas and R. Platt, "Localizing handle-like grasp affordances in 3d point clouds," in *Experimental Robotics*. Springer, 2016, pp. 623–638.

[26] K. Hsiao, S. Chitta, M. Ciocarlie, and E. G. Jones, "Contact-reactive grasping of objects with partial shape information," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 1228–1235.

[27] D. Kent, C. Saldanha, and S. Chernova, "A comparison of remote robot teleoperation interfaces for general object manipulation," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, 2017, pp. 371–379.

[28] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, 2016, pp. 3406–3413.

[29] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Robotics: Science and Systems (RSS)*, 2017.

[30] J. Fürnkranz and E. Hüllermeier, "Preference learning: An introduction," in *Preference learning*. Springer-Verlag Berlin Heidelberg, 2011, pp. 1–17.

[31] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 133–142.

[32] M. Richardson, A. Prakash, and E. Brill, "Beyond pagerank: machine learning for static ranking," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 707–715.

[33] T. Pahikkala, E. Tsivtsivadze, A. Airola, J. Boberg, and T. Salakoski, "Learning to rank with pairwise regularized least-squares," in *SIGIR 2007 workshop on learning to rank for information retrieval*, vol. 80, 2007, pp. 27–33.

[34] R. Jin, H. Valizadegan, and H. Li, "Ranking refinement and its application to information retrieval," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 397–406.

[35] O. Chapelle and S. S. Keerthi, "Efficient algorithms for ranking with svms," *Information Retrieval*, vol. 13, no. 3, pp. 201–215, 2010.

[36] D. C. G. Pedronette and R. D. S. Torres, "Exploiting pairwise recommendation and clustering strategies for image re-ranking," *Information Sciences*, vol. 207, pp. 19–34, 2012.

[37] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *Journal of machine learning research*, vol. 4, no. Nov, pp. 933–969, 2003.

[38] M. C. Gombolay, R. Jensen, J. Stigile, S.-H. Son, and J. A. Shah, "Apprenticeship scheduling: Learning to schedule from human experts." in *IJCAI*, 2016, pp. 826–833.

[39] S. Elliott, R. Toris, and M. Cakmak, "Efficient programming of manipulation tasks by demonstration and adaptation," *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017.

[40] S. Alexandrov, "Modified object segmentation database (mosd)," https://github.com/PointCloudLibrary/data/tree/master/segmentation/mOSD.

[41] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1961189.1961199

[42] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.