6DOF Grasp Planning by Optimizing a Deep Learning Scoring Function

Yilun Zhou and Kris Hauser
Department of Electrical & Computer Engineering
Duke University
Durham, North Carolina 27708
Email: yilun@mit.edu, kris.hauser@duke.edu

***Abstract*—Learning deep networks from large simulation datasets is a promising approach for robot grasping, but previous work has so far been limited to the simplified problem of overhead, parallel-jaw grasps. This paper considers learning grasps in the full 6D position and orientation pose space for non-parallel-jaw grippers. We generate a database of millions of simulated successful and unsuccessful grasps for a three-fingered underactuated gripper and thousands of objects, and then learn a modified convolutional neural network (CNN) to predict grasp quality from overhead depth images of novel objects. To generate a valid grasp from the 6D pose space, we introduce a novel optimization-based method that optimizes current suboptimal grasps using the learned grasp quality function.**

I. Introduction

Grasping is a fundamental problem in robotics, and although grasping is intuitive for humans, it is still very hard to compute reliable grasps for novel objects. There are several reasons for this difficulty. First, grasp planning is challenging due to the mathematical complexity of the problem, which involves complex geometries in close proximity, hand kinematics, actuation characteristics, and contact mechanics. Second, robots have imperfect sensing due to sensor noise and occlusion, which means even the most carefully planned grasps can fail if uncertainty is not taken into account. Finally, physics simulation of grasping is imperfect due to incompletely knowledge of the model (such as friction coefficient), and can be very slow when the hand and object models are complex.

Recent developments in machine learning methods have made data-driven approaches to grasping more popular [12, 5, 6, 7]. The data could include human labeling of good grasps [14], human teaching [1], physical experience [7], or physics simulation [11]. The vast majority of past work on grasping from a single camera image has considered learning top-down grasping with parallel-jaw grippers. This simplifies the problem because only the gripper's x-y location, angle, and width need to be learned as a function of image features. Hundreds of thousands of examples are needed to learn well even for this simplified case.

This paper uses a deep learning approach to generate grasps in a gripper's 6DOF position and orientation space, including the x-, y-, z-position, and roll-, pitch-, yaw-orientation variables of the gripper. This allows our technique to be applied to any gripper. To help generalize to new objects, we do not model the object in 3D or define any notion of object "pose," but rather learn grasps from a depth map captured by a depth camera. Our work consists of three contributions. First, we present a simulation data generation procedure to generate a massive amount of labeled data including both failed and successful grasps for a 3-fingered underactuated gripper under zero-gravity free-floating environment. Then, we present a deep learning architecture to predict a grasp quality score from a depth image and a grasp pose. Finally, we use the learned network in a grasp optimization procedure that locally optimizes a suboptimal gripper pose to have a higher score. For grasp prediction, the learned network achieves an accuracy rate of 83% on predicting grasp quality on novel objects. In addition, the method achieves a successful optimization rate of 81% on novel objects, when the initial hand pose is near a good grasp pose. As a comparison, for a naive uniform pose sampling procedure, only 2% are found to be successful.

For future work, we plan to investigate the effect of gravity on grasps obtained by the optimization procedure. We also plan to study how adaptive the control algorithm is to grasping objects that lie on a table and among other objects.

II. Related Work

Many authors have applied machine learning to robot grasping. For example, Saxena et al. [12] proposed a probabilistic model to identify candidate grasps from image features. In Jiang et al. [5], the weight for a hand-designed scoring function is learned to evaluate parallel gripper grasps represented as rectangles that are annotated on object image. Goldfeder et al. [3] devised a shape-matching method to search for grasps from the Columbia Grasp Database [2] based on SIFT features [8] of object depth images.

With the current rise of deep learning, Lenz et al. [6] used a two-stage detection process to identify grasps from RGBD image. In addition, Levine et al. [7] learns a visual-servoing control algorithm for grasping from camera image. In addition, Mahler et al. [9, 10] proposed additional datasets for parallel gripper grasping and deep learning methods to predict grasp quality.

III. Simulation Data Generation

A. Generating Grasps

We use the Klamp't [4] simulator to generate robust grasps by a free-floating gripper (Fig. 1) on a wide range of objects

Fig. 1. The RobotiQ hand model used in this project

Fig. 2. Objects from the Princeton Shape Benchmark

(Fig. 2). The six DOFs of the gripper's base control its Cartesian position x, y, z and orientation roll, pitch, yaw. Objects are drawn from the Princeton Shape Benchmark [13], and we scale each object down by a factor of 0.3 to make them of comparable size with the hand, and graspable.

We generate both successful and unsuccessful grasps in a two step process. First, we randomly sample *qualified* grasps, which are defined to be a 6DOF pose such that the hand does not collide with the object when the fingers are open, but collides when the fingers are closed. This is a necessary condition for a successful grasp. To do so, we first choose a random hand orientation, and then choose a random hand movement direction. Then we slide the hand across the object along the sampled direction, while keeping orientation fixed. At each sliding position, collision is checked to find qualified grasps. Fig. 3 illustrates this process.

Next, we simulate each qualified grasp to label it as *robust*, *loose*, or *failed*. The simulation treats the object as a free-floating object without gravity. The closing of the hand is simulated. After some specified time, if the object is still moving, this means that it has been knocked away, and this grasp is labeled *failed*. For the remaining grasps, we simulate 10 random hand shaking operations, by randomly moving the x, y, z coordinates of the hand. If the object's center of mass leaves the hand during shaking, it is considered a *loose* grasp. The remaining grasps are considered *robust*.

B. Synthesizing Depth Images

For each labeled grasp we synthesize a top-down depth image of the object from a virtual camera. The field of view (FOV) is chosen so that all parts of the object can be captured by the camera. We augment the data by randomly orienting the object, and transforming the grasp accordingly (Fig. 4).

C. Data Generation Result

After several days of parallel simulation on a 64-core machine, for 1814 objects in the Princeton Shape Benchmark, we collected 442,769 *robust* grasps, 1,622,521 *loose* grasps, and 21,271,561 *failed* grasps (all grasps are *qualified*). For data

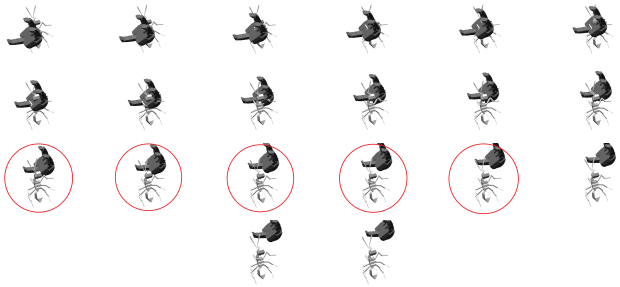

Fig. 3. Qualified grasps (circled) are determined geometrically by sliding the hand along a random direction.

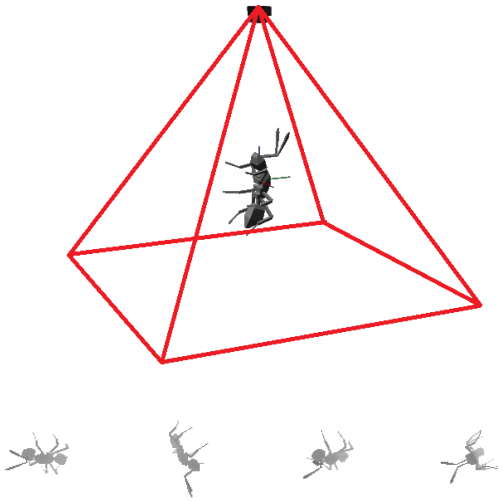

Fig. 4. Depth image synthesis. Top: camera FOV setting; bottom: depth images at random orientation

augmentation, we synthesized 1,000 images for each object at random orientations.

IV. Learning and Grasp Prediction

Grasp prediction may seem to be a regression problem at first, in which the problem is to predict $(x, y, z, roll, pitch, raw)$ grasp pose from a depth image. However, we note that the regression is actually "multi-valued", in the sense that there are multiple correct output values (grasp poses) corresponding to the same input value (depth image). Therefore, a naive regression learner that optimizes squared-loss will settle in the "average" of correct outputs.

Fig. 5 illustrates this problem. Each point in the space, consisting of a depth image value matched with a pose value, represents a problem. Points in the colored region represents a feasible grasp. As we can see, for some depth images (vertical lines), there may be one connected component of feasible grasps, multiple disconnected components, or no grasps at all. Since we are only allowed to control the robot's pose while the image stays constant, regression methods may perform poorly.

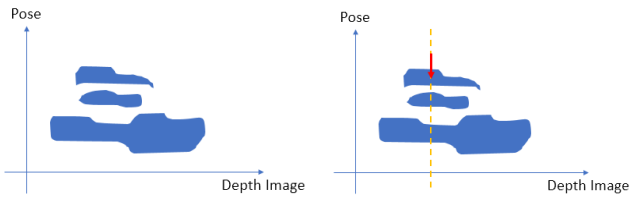

Fig. 5. Abstract illustration of the multi-valued problem space (left). We propose learning a classification score and applying a gradient-based control on the pose dimensions (right). The dashed yellow line indicates the given depth image, which the robot cannot control. Starting from an initial pose, we ascend the scoring function (red arrow).

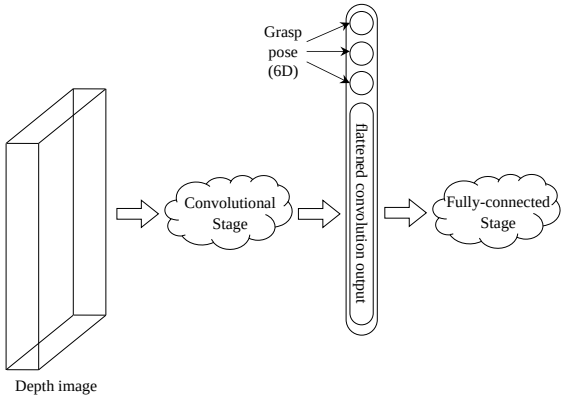

Fig. 6. The mix-in CNN architecture used for learning

Therefore, we propose an alternate formulation of this problem as a classification problem. Specifically, we train a CNN to predict a grasp score from the depth image and robot pose. Then, from a given depth image and initial suboptimal pose, we can follow the gradient of the learned scoring function with respect to pose to locally optimize the pose to a better pose.

A. Learning

To handle the multimodal input of the depth image and robot pose we use a mix-in CNN architecture [15], illustrated in Figure 6. The convolutional stage is composed of, in this order, 50 5×5 kernels, 2×2 max-pooling, 50 5×5 kernels, 2×2 max-pooling, 50 4×4 kernels, 2×2 max-pooling, 50 2×2 kernels, and 2×2 max-pooling. The input to the fully-connected stage is a vector of 1806 dimensions, and has two hidden layers of 1000 and 100 neurons respectively, before a softmax output layer. Label 0 represents failed grasps and 1 represents robust grasps.

In training, we use only robust and failed grasps, and balance the labels (thus undersampling failed grasps). We use only objects with 10 or more valid grasps. We further split these objects into a 90%/10% train/test set, which we will call them as "seen" objects and "unseen" objects. For seen objects, we did a 90%/10% split on the 1000 synthesized depth image, and use 900 images for training, and 100 for testing.

Table I shows the percentage of correct predictions on seen objects. The percentage of correct predictions on unseen objects is 83.3%. We found that for seen objects, the network generalizes slightly better to new grasp poses, than to new depth images.

Object Pose \ Grasp Pose	Train 90%	Test 10%
Train 90%	88.9%	88.7%
Test 10%	88.1%	88.1%

TABLE I
PERCENTAGE OF CORRECT PREDICTIONS FOR SEEN OBJECTS

B. Control

The control procedure finds a feasible grasp using the learned model by optimizing a scoring function. Specifically, the CNN's softmax output layer involves computing the success probability

$$\Pr(y=1) = \frac{e^{x_1}}{e^{x_0} + e^{x_1}},$$

in which $\mathbf{x} = [x_0, x_1]^T$ is output of the previous layer. During control we want to maximize $f(\mathbf{g}) = x_1(\mathbf{g}, \mathbf{I})$, in which $\mathbf{g}$ is the 6DOF grasp pose and $\mathbf{I}$ is the depth image data.

To illustrate the problem, we trained a neural network for the following classification problem: given a 20×20 matrix of zeroes I with some rectangular blobs of ones, and an (x, y) location, return the corresponding value in the matrix. The trained network achieves an error rate of less than 5%. We found that local optimization of the score $f(x, y)$ from an initial position (x_i, y_i) terminates at a blob for almost all starting points. Fig. 7 illustrates the trajectory and scoring function.

For the grasping problem, we observed that since colliding negative examples were omitted from training, high-scoring grasps often intersect the object. To avoid collision, we add an inverse barrier to the optimization. In theory, this would be achieved by modifying the objective function to be

$$f(\mathbf{g}) = x_1(\mathbf{g}, \mathbf{I}) + \frac{\alpha}{\epsilon + distance(hand, object)},$$

in which α controls the height of the barrier and $\epsilon = 0$. In practice a positive value of ϵ is needed to prevent numerical instability when evaluating $f(\mathbf{g})$ at a colliding pose, in which case the barrier is infinite. While this does not forbid collision in that zero or small negative distances are allowed, the penalty is quite large. We set the values so that the penalty for a pose with 0 distance will result in a penalty of about 500, compared to common values of 2 to 4 for $x_1(\mathbf{g}, \mathbf{I})$.

We also experimented with gradient descent and quasi-Newton methods for optimization. We found that due to different scaling of position (x, y, z, on the order of $\pm 0.1m$) and orientation ($roll, pitch, yaw$, in the range of $[0, 2\pi]$), the gradient with respect to the position is much greater in magnitude than that with respect to orientation. Thus, gradient descent spends most of its effort optimizing translation rather than rotation. Quasi-Newton can successfully deal with this

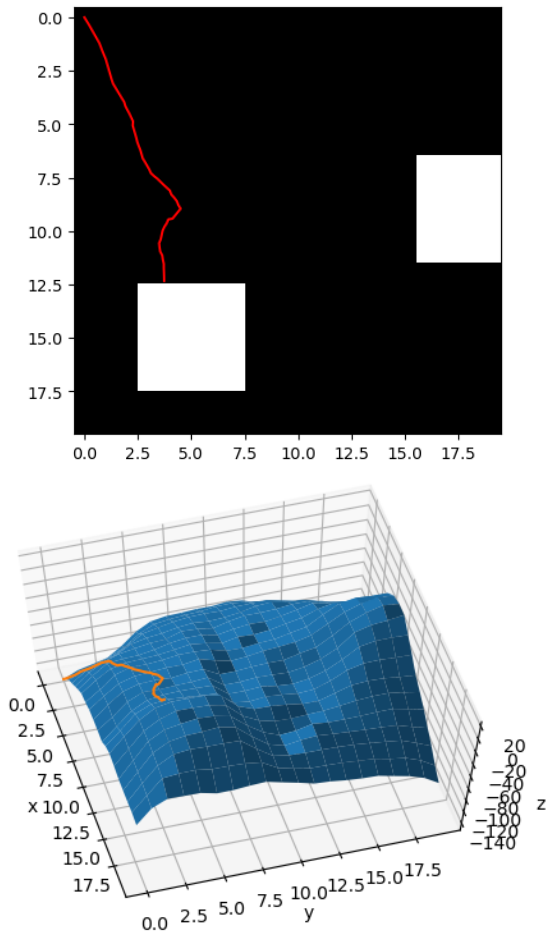

Fig. 7. Control on a toy problem, illustrating that CNNs can learn well-behaved scoring functions from classification. Top: the control problem overlayed with optimization trajectory. Bottom: scoring function is plotted on the z axis.

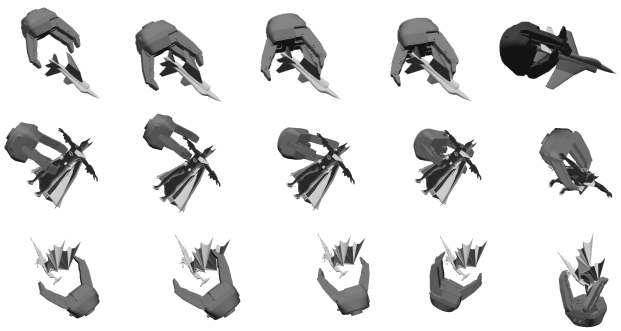

Fig. 8. Example sequences for locally optimizing a failed grasp to a robust grasp. After optimization, (top) the hand can successfully grasp the plane by the tail; (middle) the hand moves closer to the Batman and grasps from the side; (bottom) the hand changes to the opposite side of the dragon to better grasp it.

problem but leads to trajectories that are less smooth. Fig. 8 shows a successful optimization sequence on an *unseen* object.

For a more systematic testing, we generated 100 problems, on unseen objects, from robust poses by pulling the hand away from the object and perturbing the hand orientation. We made sure that the resulting pose is *qualified*. Using BFGS optimization (a form of quasi-Newton method) implemented in SciPy, 93% of the grasps are robust under one shake, and 81% are robust under 10 shakes (the training definition of robust grasp).

V. Conclusion and Future Work

This paper presented a method for generating large amount of grasping data from simulation, proposed an architecture to learn to predict grasp quality based on gripper pose and depth image, and showed how such a model can be used to do grasp planning and control.

There are several directions for future work. First, our previous work [15] presented several extensions of the CNN architecture for fusing grasp pose and depth image, and we have not yet explored the performance of other architectures.

In addition, the roll-pitch-yaw representation may be problematic because it is not unique for a given rotation, and the network does not explicitly learn the concept of equivalence. Moreover, they are periodic with period of 2π, and the network does not learn the concept of "wrapping around". In fact, we only fed the network with values between 0 and 2π, and thus it may extrapolate (unreliably) to values outside of this range during optimization. A different rotation representation, such as quaternions, might improve performance.

Finally, although we are using the depth camera for object sensing, we do assume that we know the complete model during optimization (to calculate the distance function). This can be hard to achieve, especially for novel objects. One idea would be to train the network with obviously bad (or non-*qualifying*) examples, that include explicit colliding poses as negative examples. A preliminary investigation in this direction shows that after this training, the highest scoring poses tend to be those that do not collide with the object, rather than marginally colliding. Therefore, we may even be able to use the optimization without inverse-barrier penalty to achieve "blind" model-free control.

References

[1] Staffan Ekvall and Danica Kragic. Interactive grasp learning based on human demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3519–3524. IEEE, 2004.

[2] Corey Goldfeder, Matei Ciocarlie, Hao Dang, and Peter K Allen. The columbia grasp database. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1710–1716. IEEE, 2009.

[3] Corey Goldfeder, Matei Ciocarlie, Jaime Peretzman, Hao Dang, and Peter K Allen. Data-driven grasping with partial sensor data. In *IEEE/RSJ International Conference*

on Intelligent Robots and Systems (IROS), pages 1278–1283. IEEE, 2009.

[4] Kris Hauser. Robust contact generation for robot simulation with unstructured meshes. In *Robotics Research*, pages 357–373. Springer, 2016.

[5] Yun Jiang, Stephen Moseson, and Ashutosh Saxena. Efficient grasping from rgbd images: Learning using a new rectangle representation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3304–3311. IEEE, 2011.

[6] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research (IJRR)*, 34(4-5):705–724, 2015.

[7] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *arXiv preprint arXiv:1603.02199*, 2016.

[8] David G Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157. IEEE, 1999.

[9] Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-Net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1957–1964. IEEE, 2016.

[10] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken 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.

[11] Andrew T Miller and Peter K Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004.

[12] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research (IJRR)*, 27 (2):157–173, 2008.

[13] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Shape Modeling Applications*, pages 167–178. IEEE, 2004.

[14] Jaeyong Sung, Seok H Jin, and Ashutosh Saxena. Robobarista: Object part based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds. In *International Symposium on Robotics Research (ISRR)*, 2015.

[15] Yilun Zhou and Kris Hauser. Incorporating side-channel information into convolutional neural networks for robotic tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.