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 [2, 5, 6, 7]. The data could include human labeling of good grasps [14], human teaching [11], 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 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 \((x, y, z, \text{roll, pitch, raw})\) 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.
objects is 83.3%. We found that for seen objects, the network generalizes slightly better to new grasp poses, than to new depth images.

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

To illustrate the problem, we trained a neural network for the following classification problem: given a \( 20 \times 20 \) 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

\[
f(g) = x_1(g, I) + \frac{\alpha}{\epsilon + \text{distance(hand, object)}},
\]

in which \( \alpha \) controls the height of the barrier and \( \epsilon = 0 \). In practice a positive value of \( \epsilon \) is needed to prevent numerical instability when evaluating \( f(g) \) 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 \( x_1(g, I) \).

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 \((\text{roll, pitch, yaw})\), 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
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 $2\pi$, and the network does not learn the concept of “wrapping around”. In fact, we only fed the network with values between 0 and $2\pi$, 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


