Homography-Based Deep Visual Servoing Methods for Planar Grasps

Austin S. Wang*[1], Wuming Zhang*[2], Daniel Troniak[3], Jacky Liang[2], and Oliver Kroemer[2]

***Abstract*— We propose a visual servoing framework for learning to improve grasps of objects. RGB and depth images from grasp attempts are collected using an automated data collection process. The data is then used to train a Grasp Quality Network (GQN) that predicts the outcome of grasps from visual information. A grasp optimization pipeline uses homography models with the trained network to optimize the grasp success rate. We evaluate and compare several algorithms for adjusting the current gripper pose based on the current observation from a gripper-mounted camera to perform visual servoing. Evaluations in both simulated and hardware environments show considerable improvement in grasp robustness with models trained using less than 30K grasp trials. Success rates for grasping novel objects unseen during training increased from 18.5% to 81.0% in simulation, and from 17.8% to 78.0% in the real world.**

I. Introduction

Grasping in unstructured environments is an important skill that pose a number of challenging problems. Objects with different geometric and mechanical properties behave differently when being manipulated, and it is intractable to explicitly model all possible objects a manipulator might encounter. Moreover, contact dynamics between the manipulator and the objects are usually nonlinear and discontinuous. It is therefore often preferable to directly learn to select grasps using data-driven methods. Recent research in robotic grasping [1], [2], [3], [4] explored using deep Convolutional Neural Networks (CNNs), to learn a function mapping from sensory inputs and grasping actions to a metric indicating the quality of the result a grasp. Grasping strategies are then formed by querying the learned models and optimizing over the actions.

Given sensor noise and occlusions, closed-loop control methods can greatly increase performance by making fine adjustments using sensor feedback [2]. However, optimizing inputs with respect to a complex forward model is a difficult and computationally expensive problem and is usually incapable of operating in a fast control loop.

In this paper, we present a visual servoing framework for performing planar grasps using visual feedback from a gripper-mounted RGB-D camera. A Grasp Quality Network (GQN) is trained on data collected in a self-supervised manner, and used to predict a grasp success metric from camera-centric images. We relate actions in the form of x-y-θ (horizontal movement and rotation around the vertical axis) changes to image shifts using homography. The resulting images are then given to the network for inference. Eliminating action inputs to the network relieves the need to generalize over different actions, which results in a smaller and more sample-efficient network. Using the model to rapidly evaluate batches of actions in parallel, we explore and compare several sample-based optimization methods for local grasp selection.

Fig. 1: Real world table-top grasping experiment setup with a UR10 robot. The input to our grasp planner comes from a downward-facing RealSense D435 RGB-D camera mounted on the robot's end-effector. We use a homography-based deep visual servoing framework to optimize for delta gripper poses by querying a Grasp Quality Network trained via self-supervision.

We collect data and evaluate performance on simulated and real world environments. In both setups, a UR-10 manipulator attempts to grasp cluttered objects on a tabletop using feedback from a hand-mounted RGB-D camera and wrist force sensors. In the simulation, models trained on 30K data improved grasp success rates from 18.5% to up to 81.0% on novel objects not in the training set. On the hardware setup, models trained on 20K data samples achieved success rate improvements from 17.8% to up to 78.0% on novel objects.

*Authors have contributed equally and names are in alphabetical order.
[1]Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA
[2]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA
[3]National Robotics Engineering Center, Pittsburgh, PA, USA
This work was funded by the Efort Intelligent Equipment Company. It is also in part supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1745016. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

II. Related Work

Many past works have studied data-driven grasp planning [5]. The goal is for a robot to plan and execute robust grasps of objects where the exact poses, models, and properties of the objects are unknown. One approach is to synthesize grasps for a large variety of known objects offline and then match that model to new objects online. Goldfeder et al. [6] studied how to index grasps of objects in a database from partial point cloud observations of novel objects, while Mahler et al. [7] used a multi-view CNN to leverage multiple observations of the same object. However, object matching and pose estimation can be slow and prone to errors. As a result, these methods' ability to generalize between objects is limited.

Instead of explicitly reasoning about object geometries, many recent approaches utilize CNNs to directly learn a model that predicts an abstract grasp quality metric from data via supervised learning [1], [3]. Training CNNs requires a lot of data owing to the complexities of these models and the high-dimensionality of the input space of images. For training from real-world data and using high-level actions, Pinto et al. [1] used a Baxter robot to collect 50k trials of executing random grasps with 700 robot hours. This data is then used to train a CNN that takes image patches cropped around an object and predicts grasp success at different grasp angles, if the grasp occurred at the center of those patches. Mahler et al. [3] collected a large dataset consisting of millions of synthetic depth images of grasps and analytic grasp metric labels. The synthetic dataset was used to train a CNN to predict grasp quality from depth image crops, which was then used to plan grasps on manipulators in the real world. The authors recently improved both the speed and reliability of their grasping system by extending the method to plan with both parallel jaw grippers and suction cups [8].

Learning a CNN for selecting low-level actions requires a lot more data. Using 14 robots, Levine et al. [2] collected over 800k grasp attempts over a span of 2 months to train a CNN that predicts the probability of grasp success after the robot moves its end-effector pose. The Cross Entropy Method (CEM) is then used to optimize over which action to take at a given point in time. Kalashnikov et al. [9] improved the training algorithm by Levine et al. [2], and with 7 robots collecting 580k grasp attempts over a course of 4 months they were able to achieve higher success rates than the previous work. While these approaches of learning CNNs for continuous visual servoing take a long time to collect real-world data, they are also able to learn nontrivial pre-grasp and failure-recovery motions that are difficult to achieve when planning with single-grasp motion primitives.

With CNNs that predict grasp quality for one grasp at a time, CEM or other gradient-free optimization methods are typically used to select grasps with high success rates. This can be a slow process, and recent works have also explored using fully convolutional networks (FCNs) to predict grasp successes over actions applied on the entire image [10], [11], [12]. Finding the best grasp can then be achieved by taking the argument for the maximum over the network output. Inferencing using these types of networks are fast and thus enables real-time visual servoing, but training them requires high-quality labels that indicate grasp success probabilities for all actions given one image, which are hard to obtain. Viereck et al. [13] explore an alternative, where they use gradient descent on the input of the CNN to directly optimize for the best grasp without sampling.

In our work, we explore an alternative approach to planar grasp planning that leverages homography to enable real-time visual servoing without training a network that needs to explicitly reason about robot actions.

III. Grasp Success Prediction

We consider the problem of selecting an x-y-θ (horizontal position and gripper orientation around the vertical axis) grasp pose for objects in clutter lying on a planar table surface. Objects have varying shapes, sizes, and textures, and they are not known a priori by the grasp planner. The robot has access to RGB-D images from a camera attached to the robot's end-effector. The goal is to select a grasp pose such that by executing a fixed grasping policy from the grasp pose, the robot can successfully grasp and remove objects from the workspace.

A. Homography-Based Grasp Quality Prediction

A common approach to the problem of planning grasps from visual feedback is to collect a large, labeled dataset of grasp attempts, through either simulation, analytical metrics, real-world experiments, or human experts, and use the data to train a Grasp Quality Network (GQN) to predict a grasp success metric from sensory inputs and actions [3]. The trained model will then be queried by an optimization algorithm, often gradient-free, to find grasps with high grasp success scores.

In our framework, the network estimates a grasp success metric from the image taken from a gripper-mounted camera at the end-effector pose right before executing a fixed top-down grasp policy. The objective is to optimize the gripper pose from which executing the policy would have the highest chance of successfully grasping the object. Though the presented method can be applied to many grasping policies, we focus on one in the scope of this paper: vertically lowering the gripper until force feedback exceeds some threshold, then closing the gripper to perform the grasp.

Many works [3], [8], [13] have used a grasp planner that optimizes grasp poses by querying a GQN that maps an observation of the current robot and world state along with a candidate grasp pose to predict the outcome of executing a fixed grasping policy from that state. Let S be the grasp success metric and g a candidate grasp. The GQNs of these works then take the form of $S = Q(I_g)$, where I_g is a processed image that implicitly encodes the grasp location. For example, I_g could be a cropped image centered around the grasp center and rotated to align with the grasp axis. Other works [2], [9], [14], [15] query a learned Q function with candidate end-effector actions to visual servo to states

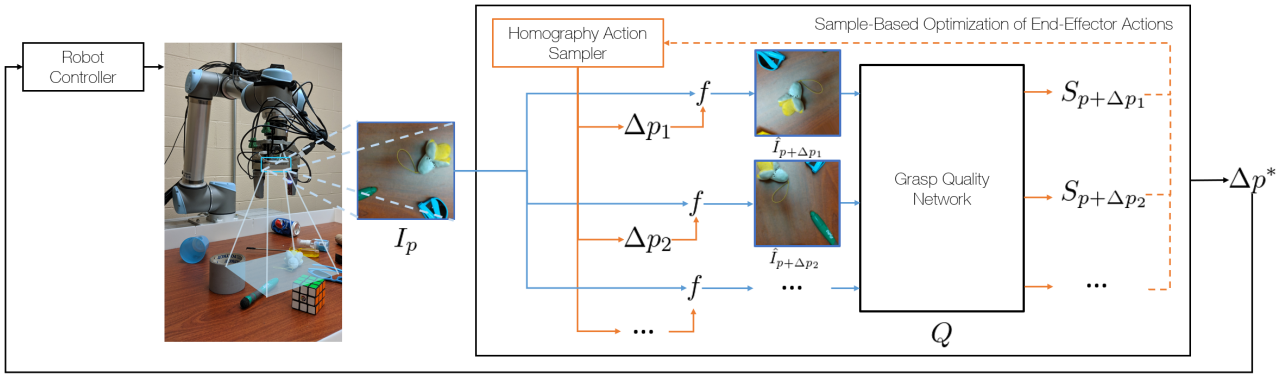

Fig. 2: Our homography-based grasp optimization pipeline for planning top-down grasps in cluttered environments. Given the current RGB-D observation I_p from the hand-mounted sensor and samples of candidate planar delta gripper poses Δp_i, we use planar affine homography f to predict what the observation image would look like if those delta poses were performed. These warped images $I_{p+\Delta p}$ are then passed to a Grasp Quality Network Q to predict their corresponding grasp success metrics $S_{p+\Delta p_i}$. The delta pose with the highest grasp success metric, Δp^*, is then performed. The robot executes a predefined grasp policy if the predicted success grasp score is greater than a fixed threshold. We evaluate the performance of our system across multiple derivative-free optimizers in both simulation and the real world. The dashed orange lines mean not all optimizers perform multiple sampling iterations.

Fig. 3: Planar affine homography is used to transform an RGB-D image taken at the current gripper pose (left) to a predicted image at another gripper pose (right) some small delta away. Pixels outside the range of the original image are extrapolated from the image edges. Using homography to approximate a forward model for images allows us to train a GQN for visual servoing that does not need to explicitly reason about robot actions and how they relate to changes in the observation space.

where executing a grasp motion or low-level policy achieves a successful grasp. The Q function of these works often take the form of $S = Q(I, p)$, where the grasp planners optimizes over the end-effector pose p to maximize S given the current observation image I.

Our method combines aspects of both approaches. We encode the candidate grasp as one executed by running a fixed grasp policy from the pose where the observation I is taken by the hand-mounted camera, but we do this in a way that enables visual servoing without explicitly incorporating an action input in our network architecture. Unlike previous works that obtain the observation image I from a static sensor placed overhead or at the robot's shoulder, our work obtains a *local* observations from a sensor attached to the end-effector [16]. This allows us to define the grasp success metric as:

$$S_p = Q(I_p) \tag{1}$$

where S_p is the success metric of executing a grasp policy from the pose p, which simultaneously corresponds to the current gripper pose and the pose of the image I_p taken by hand-mounted camera.

Our GQN has no input action. Instead, given the current image I_p and some potential action Δp, we can predict what the observation $I_{p+\Delta p}$ would be at the end of the motion Δp:

$$\hat{I}_{p+\Delta p} = f(I_p, \Delta p) \tag{2}$$

Since we use top-down grasps in our experiments, we treat p as a 2D pose (x, y, θ) at a fixed height with x, y the planar position of the robot end-effector at a fixed height, and θ the angle of the end-effector with respect to the z-axis. The delta pose is then $\Delta p = (\Delta x, \Delta y, \Delta\theta)$.

Unlike previous works, which learn a visual dynamics model via data ([17], [18]) or a Q function ([2], [9], [15], [14]), we approximate $f(I_p, \Delta p)$ for small Δp's with an analytical model using planar affine homography, which allows us to translate and rotate the image appropriately by Δp. See Figure 3 for an example.

To perform homography, we make the assumption that all pixels lie on a plane with a fixed distance orthogonal to the image plane. We compute this distance by taking the average value of all valid pixels in the depth image. Prediction errors introduced through this assumption are small since all of the objects lie on a flat table surface perpendicular to the camera and that the magnitude of Δp is constrained, and this error is also corrected in practice through feedback control. Homography can be computed entirely with matrix operations, and thus we evaluate f on the GPU to quickly generate shifted images for batches of pose shifts Δp, allowing the grasp planner to optimize Q and perform closed-loop control.

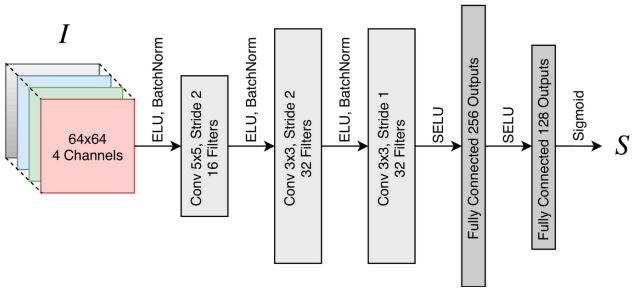

Fig. 4: Architecture of our Grasp Quality Network (GQN). The network takes in an RGB-D image and outputs a grasp success score between 0 and 1. The first two convolutional layers have stride sizes of 2 to reduce the complexity of the network while preserving spatial locality of convolution responses.

Instead of performing planar homography, an alternative approach is to deproject RGB-D images to point clouds, compute spatial transformations, then project the point clouds back into the new image. We opted not to pursue this option, due to noise and occlusions in the depth images.

Once we have an analytical model that predicts what the robot would observe from small pose shifts, we can feed its predicted images into the GQN to infer the grasp success metric at any poses within a small range of the current pose:

$$S_p = Q(I_p) \approx Q(\hat{I}_p) = Q(f(I_p, \Delta p)) \tag{3}$$

Eliminating the action input to the network helps improve sample efficiency of our GQN, as the network does not have to explicitly reason about how actions for visuomotor control affect future observations.

B. Grasp Quality Network

Our network architecture, seen in Figure 4, is relatively lightweight compared to similar works ([2], [3], [8], [9]). The raw images consist of pixel-aligned RGB-D images of size 640×480. To convert the raw images into inputs for the network we preprocess the 4-channel images through first a center-crop to 400×400 then a downsample to 64×64. We use Exponential Linear Units (ELU) and batch normalization as activations for the convolutional layers and Scaled Exponential Linear Units (SELU) for the fully connected layers. To reduce network parameters while preserving spatial locality, we use strides of 2 for convolution layers instead of max pooling.

We train the GQN via self-supervision using data autonomously collected by a robot. Grasp trials are performed with cluttered objects on a flat table surface in both simulation and the real-world using a UR-10 robot arm. The robot first moves to a predefined, fixed overhead pose, and then takes a snapshot of the workspace. A pre-grasp pose is then randomly sampled, which the robot then moves to and performs the fixed grasp policy. The label is a binary success metric S obtained by performing three shakes after executing the grasp policy and then determining if the gripper is fully closed. The input to the GQN is the RGB-D image captured at the overhead pose, which we then transform to the grasp pose using homography.

IV. Visual Servoing Methods

Once the prediction model Q is obtained, we can optimize Δp to maximize the model output S. A predefined grasp policy is executed once the predicted success metric is greater than a threshold, or when the maximum number of visual servoing iterations is reached. See Figure 2 for an illustration of the grasp optimization pipeline.

While the model can predict the success metric at any pose, due to approximation errors of using homography to estimate future observations, large changes in the pose will increase the prediction errors of the image, and consequently of the predicted success metric. Thus, visual servoing approaches that iteratively adjust the grasp pose by small amounts are desirable. Since both computing the shifted images and performing GQN inference can be parallelized with batch operations, the model benefits from optimization methods that use parallelized sampling. We implement an adaptive sampling method that scales the sampling range by $1 - S_0$ where S_0 is the success metric inferred the current center pose $S_0 = G(I_0)$. In other words, we sample across a wider region when the current metric indicates a bad grasp pose, and vice versa. This allows the controller to explore possibilities across a wider area when it's unable to find a good grasp, while optimizing locally with greater fidelity when getting close to a good grasp. We implemented and compared the following iterative approaches with our framework to find pre-grasp poses.

A. Cross-Entropy Methods

The cross-entropy method (CEM) [19] is a common derivative-free optimization method used to optimize a black box objective function, which involves iteratively sampling from a Gaussian distribution, then using the set of highest-scoring samples to fit the next Gaussian.

Full CEM: At each step of the visual servo control loop, CEM is run for 5 iterations to optimize the next pose, to which the arm would then move.

Single Iteration CEM: Performing several CEM iterations can be quite time consuming, and thus we also evaluated the performance of using only 1 iteration of CEM after each control loop. This approach allows for significantly faster visual servoing, and we tuned the maximum allowed visual servoing steps for the two CEM methods so they would require equal amounts of time to perform a grasp.

B. Finite Differences Methods

The gradient of the success metric with respect to pose change can be estimated using finite differences by sampling a fixed number of pose samples, forward propagating them through the model, and applying linear regression $g^* = \arg\min_g \sum_i (g^T p_i - S_i)^2$. The pose shift is then computed by following the direction of the gradient using descent methods.

First-Order Ascent: The most straightforward method is to move Δp in the direction of the estimated gradient using gradient descent algorithms. We sample 64 pose changes to estimate the gradient in this method. We found that in practice, RMSprop, which weighs learning rates of each element in Δp separately by their past gradient magnitudes, works well for choosing pose updates in most situations.

Second-Order Ascent: Second-order methods offer faster convergence and are invariant to the scaling of the problem, but come at the cost of being computationally expensive to execute. With enough sampled input poses, we can estimate the Hessian using polynomial regression:

$$(g^*, H^*) \approx \arg\min_{g,H} \sum_i (S_0 + g^T p_i + p_i^T H p_i - S_i)^2 \quad (4)$$

Newton's method $p_{t+1} = p_t + \alpha H^{-1} g^T$ can then be performed after obtaining the gradient and Hessian updates, where we use a learning rate $\alpha = 0.05$. 128 pose changes are sampled to estimate the gradient and the Hessian in this method.

Gauss-Newton Ascent: A large number of samples are needed to acquire an accurate Hessian estimate, which makes second-order ascent slow to execute. We therefore also experimented with the Gauss-Newton approximation to obtain a Hessian estimation from the gradient: $H \approx g^T g$, then use the approximated Hessian to perform the same update. 96 pose changes are sampled to estimate the gradient in this method.

V. Experiments

To evaluate the proposed homography-based visual servoing approach, we train the GQN via self-supervision and evaluate multiple action selection optimization algorithms in both simulation and the real world.

A. Network Training

Due to the lightweight design and sample efficiency of our model, it does not take long to train the network. Separate models were trained on 30K simulated data in V-REP [20] and 20K data collected from the hardware setup. Due to the symmetry of the parallel gripper and the fact that the shifted image is centered around the grasp pose, we can augment each data entry by mirroring the input image both horizontally and vertically. We further augmented data collected on hardware by varying the brightness of the RGB channels, and by applying slight Gaussian noise to individual pixels in all four channels. The models were trained on 20 epochs with a batchsize of 200. Both training and inference of GQN are performed on a desktop computer running Ubuntu 16.04 with a 3.7 GHz Intel Core i7-8700K and NVIDIA GTX 1080Ti GPU.

B. Experiments in Simulation

1) Data Collection: We use V-REP with Bullet v2.83 for our simulated experiments. See Figure 5. For the Bullet physics engine, we set the friction coefficient to 1.3 with sticky contacts on, and use 0.05s as the time step. For training, we collected $30,000$ grasp attempts on 30 household objects from 3DNnet [21] and the KIT object database [22]. For each grasp attempt, we chose 5 objects randomly from the training set and placed them 0.15m above the table at a random pose within the workspace of the robot arm. The objects were then dropped onto the table. The robot end-effectors start at a pre-set pose 0.5m above the table. We then randomly choose 1 object as the target, move the robot end-effectors to a random pre-grasp pose at 0.4m above the table and within $[-0.125, 0.125]$m from the center of the object in both x and y directions. The angle of the gripper is also uniformly sampled for the initial data collection. We then move the end-effector directly down to the table height, or until contact with the object, and then close the grippers. Once the grippers are closed, the robot performs a lift motion to move the end-effectors to 0.25m above the table. A grasp is considered successful if the gripper is not completely closed after the lift motion. The entire data collection process took 139 hrs in a single simulation running on a single desktop computer with Ubuntu 16.04 with a 3.7 GHz Intel Core i7-8700K.

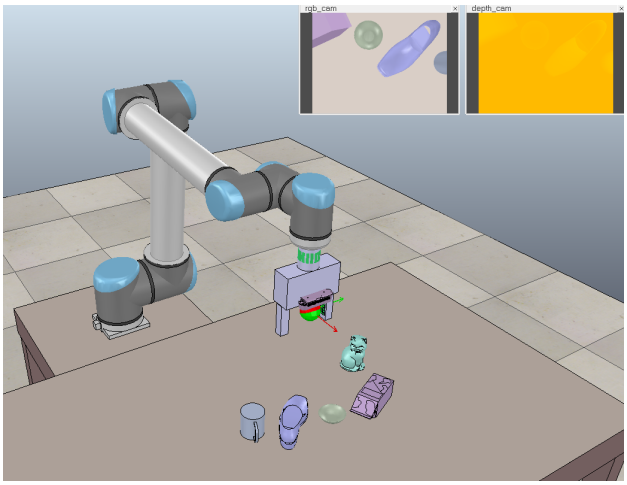

Fig. 5: Simulation table-top grasping environment with a UR10 robot in V-REP using a Bullet backend. During data collection, for each simulation grasp attempt 5 objects are randomly dropped onto the table surface, and the robot initiates a top-down grasp from a random pose above the objects.

2) Performance Evaluation: For testing, we use 10 novel objects from 3DNnet [21] and the KIT object database [22]. For each grasp attempt, we choose 5 objects randomly from the testing set and put them on the table randomly. Similar to data collection, the robot end-effectors are at the same pre-set pose at the beginning of each attempt. We then move the end-effectors to a random pre-grasp pose as the initial start for visual servoing. Once our grasp planner finds a grasp with a predicted success metric greater than 0.95, the robot executes a predefined top-down grasp policy, where the robot lowers its end-effectors until it reaches the table surface or makes contact with an object. Once a contact is achieved, the robot closes its grippers and performs a lift motion by

Visual servoing method	Success rate		Inference
	Training set	Testing set	time (ms)
Random grasps	20.0%	18.5%	
1st-order ascent	81.0%	81.0%	47
2nd-order ascent	64.5%	62.0%	89
Gauss-Newton ascent	77.0%	74.5%	62
Single iteration CEM	77.5%	75.0%	204
Full CEM	72.5%	74.5%	985

TABLE I: Performance of various visual servoing methods in simulation. All methods except for full CEM only require network inference on a single batch. The difference in inference time is due to different number of samples required for good performance and thus different batchsizes.

0.25m. We label each attempt similarly to our data collection stage.

We benchmarked the query times for different visual servoing methods. The results can be seen in TABLE I.

C. Experiments in Real World

1) Data Collection: For training on the real-world robot, we collected $20,000$ grasp attempts of 53 common office and household objects using a UR-10 robot. See Figure 1 for the grasping experiment setup. We inpaint and threshold the depth images to reduce noise from the real-world depth sensor. Collecting all the training data took about a week. All objects were put randomly on a wooden table top within a range that is reachable by the robot end-effectors. See Figure 6 for the objects used during training.

For each grasp attempt, the robot end-effector started at a pre-set pose 0.5m above the table. Since there is no ground-truth poses for the objects, we uniformly sample an end-effector pose p across the reachable region on the table top, and set the height to be 0.2m above the table. The robot moves its end-effectors to that pre-grasp pose and lowers the end-effectors vertically until reaching table top or making contact with an object. It then closes the grippers and moves up by 0.25m. The label for each attempt is determined by whether the grippers are fully closed. If the robot successfully grasps an object, it moves the the object to a random location above the workspace and drops the object.

2) Performance Evaluation: Similar to during data collection, the robot starts at a pre-set pose and moves to a random pre-grasp pose 0.2m above the table, where it then starts visual servoing. The visual servoing stops once the predicted success metric exceeds a threshold of 0.85 or it hits a maximum number of iterations. During visual servoing the predicted grasp success metric and the corresponding delta poses are stored. If the maximum number of visual servo steps is reached without discovering a grasp above the threshold, the robot executes the grasp from the pose that had the highest predicted success metric. During execution of our grasp policy for real world experiments, we detect contact with objects or the table top using the robot's force feedback, and we set the corresponding upward force threshold to 40N. The results can be seen in TABLE II.

Fig. 6: Objects used to collect training data for real world experiments. We used 56 household and office objects of various shapes, colors, and deformation properties. To make the table top background more realistic the objects are placed on a textured wooden table with faint diffuse reflection.

Visual servoing method	Success rate	
	Training set	Testing set
Random grasps	20.0%	17.8%
1st-order ascent	71.0%	65.0%
Single iteration CEM	84.5%	78.0%

TABLE II: Performance of various visual servoing methods on real world experiments.

VI. Results and Discussion

The evaluation results showed a significant increase in grasp robustness for all the methods. In addition to the performance evaluation, additional studies on the input features, network outputs, and successful/failure cases also gave insight into what was being learned by the network.

A. Performance Evaluation

In TABLE I and TABLE II, we observe that our methods using homography outperform the random grasp baseline by a large margin in both simulation and real world test. The 1st-order ascent method achieves the best performance with the least inference time in simulation with similar success rates across train and test sets. This is not the case in the real world experiments where there is a large gap in performance between train and test. This difference may be explained by the fact that the GQN has more approximation errors for real world objects in the test set, so approximated gradients are much noisier for first-order optimization methods in the real world, especially for the test set. The number of samples used by each method on each iteration are tuned in simulation and on the real world setup. The two CEM methods require a lot more samples to achieve good results. In simulation the single iteration CEM performed better than its full counterpart, which shows that for the CEM methods numerous suboptimal steps are more efficient than fewer fully-optimized steps.

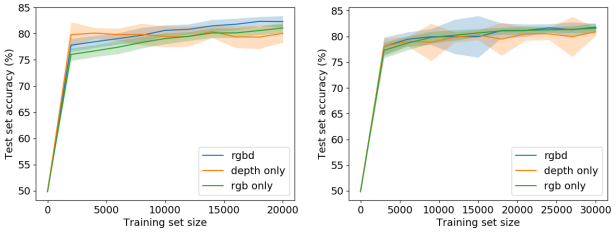

Fig. 7: Ablation study of training our GQN with varying types of channels for both simulation (left) and the real world setup (right). Note that this is not a training curve, but testing results from models fully trained on different amounts of data. We cross-validate over 30 trials by sampling a test set at the beginning of each trial and excluding the test set from the dataset to form the training set. Then, we train models for multiple epochs on different amounts of data sampled from the training set. Performance is evaluated with balanced datasets sampled from the test set of that trial. To evaluate accuracy we set the output of the GQN to be 1 if the predicted success metric is over 0.5.

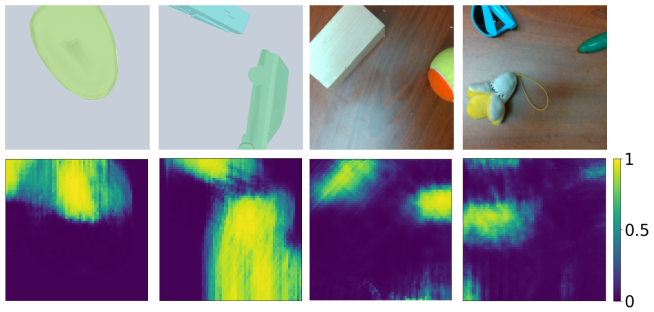

Fig. 8: Images from hand-mounted cameras (top row) and corresponding heatmaps of success metrics (bottom row) of GQN predicted grasp success metrics for pure translations. The left two examples are from simulation, the right two the real world. We compute the value of a pixel in the heatmap by first performing a homography with only pure translation that shifts the image to be centered around the location of the pixel and getting its predicted metric from the GQN.

B. Ablation Study of Using RGB vs Depth

We evaluated with using just the RGB channels or just the depth channel. We evaluate the resulting models using cross-validation as seen in Figure 7. For models trained on real world data, using depth only is more sample efficient when only a small amount of data is available. However, due to noise in the depth images, the network is able to perform better using information from RGB channels once there is enough data to generalize over different colors. In addition, the simulated objects have uniform texture, so the addition of RGB information do not significantly increase the amount of information seen by the network. Thus, while RGB-D performs the best in the real world, their differences are much less apparent in the simulation.

C. Grasp Heatmap

For each RGB-D image, we can generate a heatmap (see Figure 8) by querying the network with different shifts (translation only) of the same image. Each pixel in the heatmap refers to the predicted success score of the warped image centered at that pixel location. By looking at the resulting heatmaps, we observe several characteristics of graspable regions that agree with physical intuition. Firstly, since the gripper is located below the camera from the camera perspective, the graspable region corresponding to and object will have a slight vertical bias, which is equal to the distance between the camera and the center of the gripper. Narrow, vertical objects in the image result in a large graspable region since there is a large range of poses the gripper can be in while still having the two fingers on either side of the object. Conversely, wide objects usually have a thin graspable region in the center. Also, graspable regions of round objects take on a rectangular shape, corresponding to the range of grasp points close enough to the center such that squeezing the round object would not cause it to roll out between the fingers. Moreover, we can usually observe regions with slightly lower metric scores on either side of wider objects. This is because when a finger presses into the edge of the object from above, there is a probability, depending on the properties of the object, that the object will be pushed into the area between the fingers.

D. Visual Servoing Duration

At each iteration of visual servoing, the robot chooses whether or not to continue executing the policy by querying the GQN with the current image feedback. We group the number of visual servoing steps throughout different grasp attempts in bins and evaluate the grasp success rate in each bin individually, as shown in Figure 9. The last bar in each plot indicates the maximum iteration limit of each method, at which the robot would execute the grasp policy regardless of the current sensory inputs. We observe that it is more likely for CEM to choose a good quality grasp in a few iterations if it is able to find one, while the 1st-order ascent method needs varying numbers of steps to find a favorable grasp. Most failures in the 1st-order ascent method can be attributed to not being able to find a confident grasp within the maximum iteration limit.

E. Failure Cases

The approximation errors of using homography to predict images taken at different camera poses arise from perspective changes, occluded objects coming in and out of view, and the extrapolated pixels around the borders of the shifted image. In addition, we observed that while depth data can improve grasp success rate in simulation, this is not necessarily true in the real world due to noise in the RealSense camera. The noise can make flat surfaces appear bumpy and sharp edges appear round, which affect the apparent robustness of a grasp. In addition, unlike many previous work using uniform-colored background, we use a table top with a wooden texture. More data and a better depth sensor can potentially allow the GQN to generalize over these noises and variations.

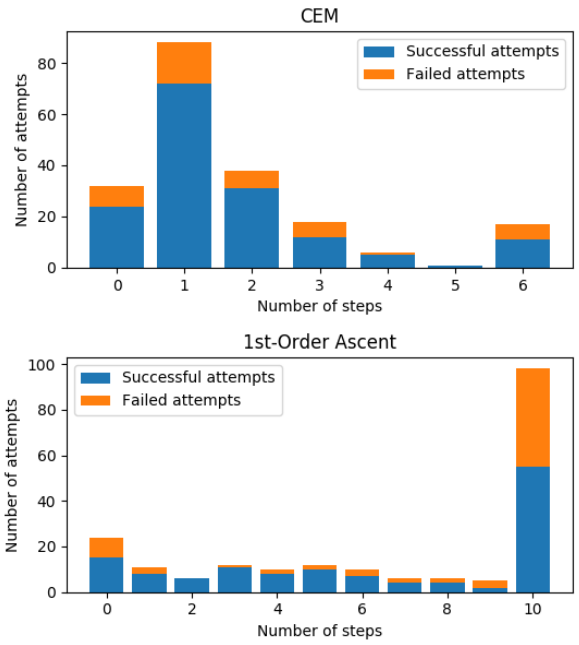

Fig. 9: We record and bin the number of visual servoing steps the robot performs before deciding to execute a grasp policy during evaluation on the test set. We also separately denote the number of grasps that are successful in each bin. This provides insight into how many steps it takes for each method to find a good grasp pose and how good the chosen poses actually are.

VII. Conclusion and Future Works

In this work, we propose a deep visual servoing method for grasps that use planar affine homography to approximate changes in observations from a gripper-mounted camera and a Grasp Quality Network trained via self-supervision. Because the proposed GQN does not have to explicitly reason about the effects of actions on observations, which are modeled via homography, the GQN can be expressed by a lightweight CNN and is sample efficient. Additionally, predicted grasp metrics can be computed quickly on the lightweight CNN, which allows for real-time visual servoing.

We evaluated our method in both simulation and the real world with multiple action optimization algorithms. With models trained using less than 30K grasp trials, we achieve significant improvement on the accuracy of grasping novel objects.

Acknowledgment

The authors would like to thank Jialiang Zhao for his help on setting up the simulation environment, and Samuel Clarke for his work on designing the camera mount. The authors would also like to thank Joseph Giampapa, Brad Lisien, Eric Foote, and Eric Schmidt for their work on engineering the experiment platform.

References

[1] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," *CoRR*, 2015.

[2] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *IJRR*, vol. 37, no. 4-5, pp. 421–436, 2018.

[3] 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," 2017.

[4] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 4304–4311.

[5] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesisa survey," *TRo*, vol. 30, no. 2, pp. 289–309, 2014.

[6] C. Goldfeder and P. K. Allen, "Data-driven grasping," *AuRo*, vol. 31, no. 1, pp. 1–20, 2011.

[7] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. 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 *ICRA*, 2016, pp. 1957–1964.

[8] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, 2019.

[9] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *CoRL*, 2018.

[10] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," 2018.

[11] V. Satish, J. Mahler, and K. Goldberg, "On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks," *RA-L*, 2019.

[12] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," in *ICRA*, 2018, pp. 1–8.

[13] U. Viereck, A. t. Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using simulated depth images," *CoRL*, 2017.

[14] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *ICRA*, 2018, pp. 4243–4250.

[15] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, A. Ray, J. Schneider, P. Welinder, *et al.*, "Domain randomization and generative models for robotic grasping," in *IROS*. IEEE, 2018, pp. 3482–3489.

[16] M. Gualtieri and R. Platt, "Learning 6-dof grasping and pick-place using attention focus," *CoRL*, 2018.

[17] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *ICRA*, 2017, pp. 2786–2793.

[18] F. Ebert, C. Finn, S. Dasari, A. Xie, A. X. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," *CoRR*, 2018.

[19] R. Y. Rubinstein and D. P. Kroese, *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-carlo Simulation (Information Science and Statistics)*. Springer-Verlag, 2004.

[20] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *IROS*, 2013, pp. 1321–1326.

[21] W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze, "3dnet: Large-scale object class recognition from cad models," in *ICRA*, 2012, pp. 5384–5391.

[22] 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," *IJRR*, vol. 31, no. 8, pp. 927–934, 2012.